Probabilistic Protein Homology Modeling

Armin Jonathan Olaf Meier

München 2014

Dissertation zur Erlangung des Doktorgrades
der Fakultät für Chemie und Pharmazie
der Ludwig-Maximilians-Universität München

Probabilistic Protein Homology Modeling

Armin Jonathan Olaf Meier

aus

Memmingen, Deutschland

2014

Erklärung:

Diese Dissertation wurde im Sinne von §7 der Promotionsordnung vom 28. November 2011 von Herrn Dr. Johannes Söding betreut.

Eidesstattliche Versicherung:

Diese Dissertation wurde eigenständig und ohne unerlaubte Hilfe erarbeitet.

München, am 17. Juni 2014

Armin Meier

Dissertation eingereicht am

1. Gutachter: Dr. Johannes Söding
2. Gutachter: PD Dr. Dietmar Martin

Mündliche Prüfung am 27. Juni 2014

Acknowledgments

The completion of this thesis could not have been accomplished without the help of several people.

First and foremost, I would like to thank Dr. Johannes Söding for having given me the opportunity to work in his group. With his passion and enthusiasm for science he was an outstanding mentor for me, and I am very happy to have profited from his wealth of knowledge and his approach to tackle scientific problems.
I would like to thank PD Dr. Dietmar Martin for agreeing to be my second PhD examiner. I am also very grateful to Prof. Dr. Patrick Cramer, Prof. Dr. Klaus Förstemann, Prof. Dr. Mario Halic and Prof. Dr. Ulrike Gaul for offering their time as members of my committee.
Next, I owe a debt of gratitude to Jessica Andreani for having critically read parts of this thesis. Also thanks to Stefan Seemayer who has made valuable remarks on paper drafts and was very helpful with all kinds of technical questions.
Furthermore, I would like to thank all members of the Söding and Gagneur group for the pleasant working atmosphere in the office.
Finally, I would like to thank my parents for their support during my studies.

Summary

Searching sequence databases and building 3D models for proteins are important tasks for biologists. When the structure of a query protein is given, its function can be inferred. However, experimental methods for structure prediction are both expensive and time consuming. Fully automatic homology modeling refers to building a 3D model for a query sequence from an alignment to related homologous proteins with known structure (templates) by a computer. Current prediction servers can provide accurate models within a few hours to days. Our group has developed HHpred, which is one of the top performing structure prediction servers in the field.

In general, homology based structure modeling consists of four steps: (1) finding homologous templates in a database, (2) selecting and (3) aligning templates to the query, (4) building a 3D model based on the alignment.

In part one of this thesis, we will present improvements of step (2) and (4). Specifically, homology modeling has been shown to work best when multiple templates are selected instead of only a single one. Yet, current servers are using rather ad-hoc approaches to combine information from multiple templates. We provide a rigorous statistical framework for multi-template homology modeling. Given an alignment, we employ Modeller to calculate the most probable structure for a query. The 3D model is obtained by optimally satisfying spatial restraints derived from the alignment and expressed as probability density functions. We find that the query's atomic distance restraints can be accurately described by two-component Gaussian mixtures. Moreover, we derive statistical weights to quantify the redundancy among related templates. This allows us to apply the standard rules of probability theory to combine restraints from several templates. Together with a heuristic template selection strategy, we have implemented this approach within HHpred and could significantly improve model quality. Furthermore, we took part in CASP, a community wide competition for structure prediction, where we were ranked first in template based modeling and, at the same time, were more than 450 times faster than all other top servers.

Homology modeling heavily relies on detecting and correctly aligning templates to the

query sequence (step (1) and (3) from above). But remote homologies are difficult to detect and hard to align on a pure sequence level. Hence, modern tools are based on profiles instead of sequences. A profile summarizes the evolutionary history of a given sequence and consists of position specific amino acid probabilities for each residue. In addition to the similarity score between profile columns, most methods use extra terms that compare 1D structural properties such as secondary structure or solvent accessibility. These can be predicted from local profile windows.

In the second part of this thesis, we develop a new score that is independent of any predefined structural property. For this purpose, we learn a library of 32 profile patterns that are most conserved in alignments of remotely homologous, structurally aligned proteins. Each so called "context state" in the library consists of a 13-residue sequence profile. We integrate the new *context score* into our HMM-HMM alignment tool HHSEARCH and improve especially the sensitivity and precision of difficult pairwise alignments significantly.

Taken together, we introduced probabilistic methods to improve all four main steps in homology based structure prediction.

Contents

List of Figures

List of Tables

Abbreviations

AA	Amino Acid
BLAST	Basic Local Alignment Search Tool
CASP	Critical Assessment of Structure Prediction
CTX	ConTeXt
EM	Expectation Maximization
GDT	Global Distance Test
HMM	Hidden Markov Model
LL	Log Likelihood
LSO	Log-Sum-of-Odds
MAC	Maximum ACcuracy
MSA	Multiple Sequence Alignment
PDB	Protein DataBase
PSI-BLAST	Position Specific Iterated BLAST
ROC	Rank Order Characteristic
SCOP	Structural Classification Of Proteins
SOM	Self Organizing Map
SS	Secondary Structure
TM	Template Modeling

1. Introduction to homology detection and modeling

1.1. Pairwise sequence alignment

A central concept in biology is that new organisms such as plants or animals evolve from already existing ones. This also holds on a molecular level, where proteins can be traced back to their ancestors. In the course of time, however, proteins had to adapt to changing environmental conditions. Thus their amino acid sequences were subject to mutations. Depending on the time scale and selective pressures, these mutations accumulated. In the end, proteins for which a common ancestor can be inferred are said to be homologous. Often, homologous proteins vary considerably in sequence but share most of their characteristic properties such as function and structure. Consequently, sequence homology is of major interest for biologists, because it allows the investigation of functions and structures of new and uncharacterized proteins.
In order to detect homologous proteins, computational methods have been developed. A so-called alignment is a way to identify similar regions in two sequences that originate from evolutionary relationships between the proteins. Each residue in one sequence is paired with either a similar residue in the other sequence or with a gap. Since numerous such pairings are possible, a best one with respect to some quality score has to be found. It is assumed that a sequence gets mutated into any of its successor sequences by a series of substitutions. To assess their score, substitution matrices have been generated. They describe rates of mutation of each amino acid into another. Such a matrix reflects that in homologous proteins mostly physico-chemically similar amino acids are substituted. Computational biology has come up with algorithms that compute the highest scoring alignment given a substitution matrix.
More formally, given two sequences $q = q_1 \dots q_n$ and $t = t_1 \dots t_m$ with $q_i, t_j \in \Sigma$ and $\bar{\Sigma} = \Sigma \cup \{-\}$, a global alignment between q and t is a pair $(\bar{q}, \bar{t}) \in \bar{\Sigma} \times \bar{\Sigma}$ with $|\bar{q}| = |\bar{t}|$ and $(\bar{q}_i, \bar{t}_i) \neq (-,-)$ for all $i = 1, \dots, |\bar{q}|$. For instance, if Σ is the alphabet of amino acids, Figure 1.1 depicts an alignment for $q = AGTPDR$ and $t = APERL$.

$$\begin{array}{cccccccc} A & G & T & P & D & R & - \\ A & - & - & P & E & R & L \end{array}$$

Figure 1.1.: Pairwise alignment of $q = AGTPDR$ and $t = APERL$.

Given a substitution matrix S, each entry $S(a, b)$ estimates the rate of change from a to b, where $a, b \in \Sigma$. Furthermore, pairs containing a gap are scored with a gap penalty function $g(a, -)$ or $g(-, a)$. Mostly, it is defined so that opening a gap is quite expensive whereas extending an existing gap increases the cost linearly in its length. In the example above, the first gap in column 2 is a 'gap opening' whereas the second gap in column 3 is an extension of the previous one.
The Needleman-Wunsch algorithm (Needleman and Wunsch, 1970) computes a global alignment between q and t with maximal score by dynamic programming as follows: a $n \times m$ matrix D is filled successively where $D(i, j)$ holds the optimal score for an alignment between $q_{1\ldots i}$ and $t_{1\ldots j}$. Extending this sub-alignment is possible in three ways: either q_{i+1} matches t_{j+1} or q_{i+1} is aligned to a gap in t or t_{j+1} is aligned to a gap in q; see Figure 1.2 for the match case.

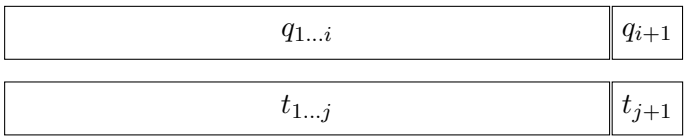


Figure 1.2.: Extending the sub-alignment between $q_{1\ldots i}$ and $t_{1\ldots j}$ with a match.

The optimal extended alignment is then found by maximizing the scores over all three possibilities:

$$D(i+1, j+1) = \max \begin{cases} D(i, j) + S(q_{i+1}, t_{j+1}), \\ D(i, j+1) + g(q_{i+1}, -), \\ D(i+1, j) + g(-, t_{j+1}). \end{cases} \tag{1.1}$$

Starting in the upper left corner of D, all cells are filled according to equation (1.1), see also Figure 1.3. In the end, $D(n, m)$ contains the score of the optimal global alignment. To find the actual alignment, a backtrace is carried out. Here, one starts at the cell in the lower right, $D(n, m)$, and determines the cell that maximized $D(n, m)$ by reversing equation (1.1). This procedure is continued until $D(1, 1)$ is reached.

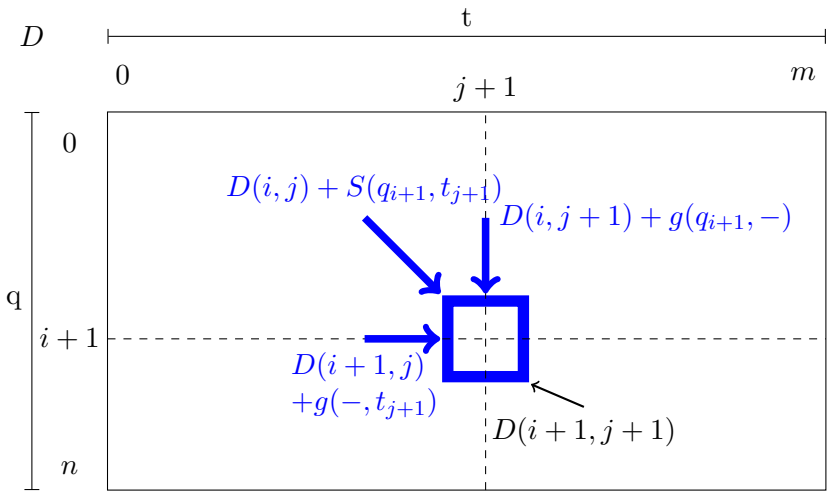


Figure 1.3.: Filling the dynamic programming matrix D: this is a graphical representation of eq. (1.1) and shows the three different options to fill in $D(i+1, j+1)$.

Global alignments assume that q and t are similar along their whole sequences. Yet, when for instance q is a multiple domain protein whereas t is not, aligning these sequences globally seems counter-intuitive. Likewise, remote homologs that share only parts of their sequences are not supposed to be completely alignable. In these cases, a local alignment makes more sense. Smith and Waterman (Smith and Waterman, 1981) found that by including zero as a fourth option into equation (1.1), local alignments can be calculated analogously:

$$D(i+1, j+1) = \max \begin{cases} 0, \\ D(i, j) + S(q_{i+1}, t_{j+1}), \\ D(i, j+1) + g(q_{i+1}, -), \\ D(i+1, j) + g(-, t_{j+1}). \end{cases} \tag{1.2}$$

Now, the backtrace no longer starts at $D(n, m)$ but at the cell with the highest value in D.

Concerning the complexity of both the global and local sequence-sequence alignment algorithms, the entire D matrix has to be filled. Backtracing can be prepared along the way resulting in a total number of steps of $O(nm)$.

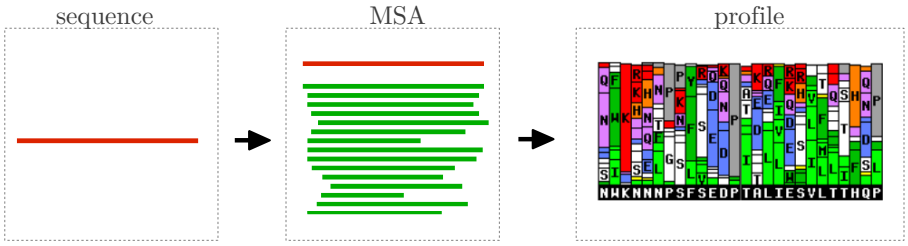


Figure 1.4.: A single sequence is extended to a multiple sequence alignment (MSA) by searching homologs in a database. The MSA is then converted into a sequence profile (right).

1.2. Profile alignments

As outlined in section 1.1, any protein sequence has numerous homologs which share characteristic properties and which, taken together, form a family of proteins. In addition to pure sequence information, such a family summarizes evolutionary information. Specifically, when certain amino acids are of crucial importance for a protein, they will be strongly conserved in all members of the family. Moreover, several consecutive positions may exhibit a characteristic composition of amino acids.

In order to exploit the extra information contained in protein families, large sequence databases are searched. All hits are combined into a multiple sequence alignment (MSA), i.e. an extension of the pairwise sequence alignment described in the previous chapter for more than two proteins. Subsequently, the MSA is converted into a more compact sequence profile by calculating the frequencies of each amino acid at each position, see Figure 1.4. As indicated in the profile on the right in Figure 1.4, it provides for each sequence residue a position specific distribution over all 20 amino acids. A single sequence can be regarded as a special profile where each position has frequency one for its amino acid. Obviously, this constitutes a strong constraint and the extended profile is much more expressive. Methods based solely on sequence suffer from the inability to detect and correctly align remote homologs because there is not enough information in the bare sequences. Here, profiles help out and the most sensitive and accurate alignment tools rely on comparing profiles instead of single sequences. Probably the most prominent example for the advantage of profiles is the upgrade of BLAST (Basic Local Alignment Search Tool, Altschul et al. (1990)) to PSI-BLAST (Position-Specific Iterative BLAST, Altschul et al. (1997)). Whereas the first one is a heuristic to speed up the

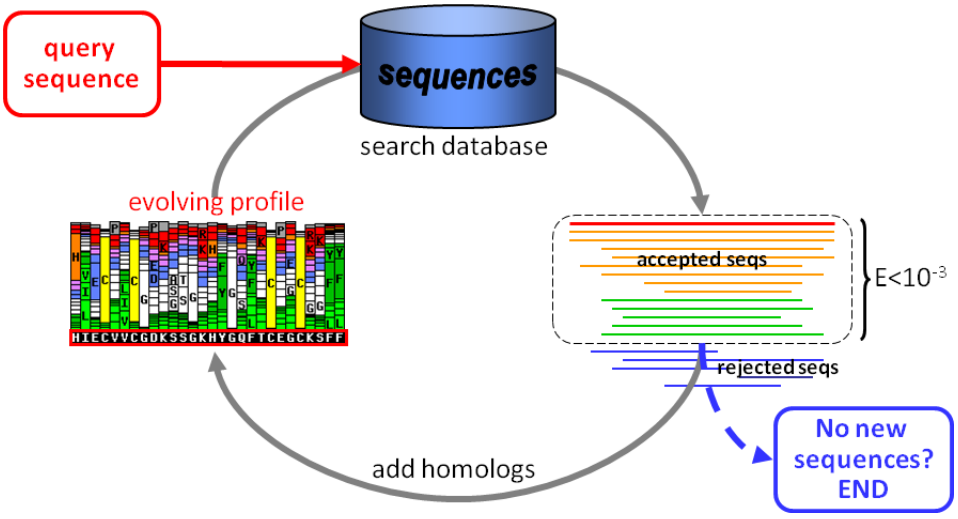


Figure 1.5.: Workflow of Psi-Blast: first, a query sequence is searched against a database by performing pairwise alignments with a BLOSUM substitution matrix. Next, all accepted hits are merged into a profile. This profile is then used in the next iteration to refine the substitution matrix. That way, it becomes more likely to detect remote homologs.

calculation of pairwise sequence-sequence alignments, the latter outperforms Blast in both sensitivity and precision by iteratively building up a query profile, see Figure 1.5. In particular, a given query sequence is searched against a database using a BLOSUM substitution matrix. All accepted hits are then combined into a profile. In the next round, this profile is used to refine the substitution matrix, which makes it more likely to detect more remote homologs. In contrast to a substitution matrix such as BLOSUM that has been trained on a large number of alignments from various families, a sequence profile is much more specific as it summarizes information from previous rounds, i.e. it involves mainly sequences from the query family.
Even if Psi-Blast increased the power of homology detection tools considerably, there is still room for improvements. A natural extension is to replace sequences with profiles also on the database side. Today, virtually all of the most sensitive and accurate remote homology detection methods are fundamentally based on profile-profile comparison.

1.3. HMM-HMM alignment

Enriching a single sequence to a sequence profile is one of the main sources of success in remote homology detection. The extension of Blast (Boratyn et al. (2013)) to Psi-

BLAST (Altschul et al. (1997)) marked the beginning of the common use of profiles. A next step led to profile-profile alignment tools (Ohsen et al. (2003); Wang and Dunbrack (2004); Wu and Zhang (2008)), which were shown to outperform single sequence based methods. Finally, a sequence profile can be encoded as a Hidden Markov Model (Profile-HMM, Eddy (1998)), which provides a statistical framework for the amino acid frequencies in the columns of the profile and additionally contains position specific insertion and deletion probabilities (Eddy (1998)). Söding (2005) introduced HMM-HMM comparison and thereby raised the power of remote homology detection to the current state of the art. The next section briefly describes how to compare two HMMs and introduces some of the alignment features used throughout this thesis.

1.3.1. Log-sum-of-odds score

Given two HMMs q and t of length L_q and L_t we follow Mückstein et al. (2002), and express the probability for an alignment $\mathcal{A}$ as:

$$P(\mathcal{A}) = \frac{e^{\beta S(\mathcal{A})}}{Z}, \tag{1.3}$$

where $S(\mathcal{A})$ is the score for the alignment $\mathcal{A}$ between q and t, $\beta = 1/kT$ (T = temperature, k = Boltzmann constant) is assumed to be one, and Z is the partition function:

$$Z = \sum_{\mathcal{A}} e^{\beta S(\mathcal{A})}. \tag{1.4}$$

Z serves as a normalization factor and thus runs over all alignments. $S(\mathcal{A})$ is the score of the alignment. Ideally, comparing homologous sequences results in consistently higher scores than for non-homologs. Given an HMM, a log-odds score can be defined which assesses how much more likely a sequence $x_1, \ldots, x_L$ is emitted by the HMM than by a simple null model:

$$S_{\mathrm{LO}} = \log \frac{P(x_1, \ldots, x_L|\text{emission on path})}{P(x_1, \ldots, x_L|\text{nullmodel})}.$$

Söding proposed a generalization for HMM-HMM comparison:

$$S_{\mathrm{LSO}} = \log \sum_{x_1,\ldots,x_L} \frac{P(x_1, \ldots, x_L|\text{co-emission on path})}{P(x_1, \ldots, x_L|\text{nullmodel})}. \tag{1.5}$$

This Log-Sum-of-Odds (LSO) score is a natural extension of the Log-Odds score as can be seen when one HMM encodes only a single sequence and thus the sum reduces to a single term. Writing out equation (1.5) more concretely results in:

$$S_{\text{LSO}} = \sum_{k:X_kY_k=MM} S_{\text{aa}}(q_{i(k)}, t_{j(k)}) + \log \mathcal{P}_{tr}, \tag{1.6}$$

where $\mathcal{P}_{tr}$ is the product of all transition probabilities for the path through q and t and the sum runs over all match-match states. Herein, $S_{aa}(q_i, t_j)$ is the column score:

$$S_{\text{aa}}(q_i, t_j) = \log \sum_{a=1}^{20} \frac{q_i(a)\ t_j(a)}{f(a)}. \tag{1.7}$$

$q_i(a)$ and $t_j(a)$ denote the probability of amino acid a in column i and j in the HMMs q and t, respectively. $f(a)$ is the background probability of amino acid a which serves as a weighting factor so that rare amino acids get a higher weight and columns with only background frequencies will cancel out giving a score of $\log(1) = 0$.

1.3.2. Posterior probabilities

In order to assess the local alignment quality between residue i in q and j in t, we aim to calculate the posterior probability for the pair state match-match (M_i^q, M_j^t) to be part of an alignment between q and t:

$$P(M_i^q \diamond M_j^t | q, t). \tag{1.8}$$

Therefore, all alignments having pair state (M_i^q, M_j^t) must be considered. In this regard, two auxiliary functions are defined:

$$F_{MM}(i, j) = \sum_{\mathcal{A} \in \mathcal{F}_{i,j}} e^{\beta S(\mathcal{A})} \tag{1.9}$$

and

$$B_{MM}(i, j) = \sum_{\mathcal{A} \in \mathcal{B}_{i,j}} e^{\beta S(\mathcal{A})}. \tag{1.10}$$

$\mathcal{F}_{i,j}$ is the set of all alignments between $q_{1\ldots i}$ and $t_{1\ldots j}$ ending in the pair state (M_i^q, M_j^t) and similarly, $\mathcal{B}_{i,j}$ is the set of all alignments between $q_{i+1\ldots L_q}$ and $t_{j+1\ldots L_t}$ starting after the pair state (M_i^q, M_j^t). Typically, F_{MM} is called the Forward and B_{MM} the Backward

partition function.
Combining all (local) alignments that end in (M_i^q, M_j^t) with all that start in (M_i^q, M_j^t) yields the posterior:

$$P(M_i^q \diamond M_j^t | q, t) = \frac{F_{MM}(i,j) \times B_{MM}(i,j)}{Z}. \tag{1.11}$$

The partition function for local HMM-HMM comparison, Z, is expressible as:

$$Z = 1 + \sum_{i,j} F_{MM}(i,j), \tag{1.12}$$

where the number one at the beginning comes from the empty alignment.

1.3.3. Alignment between two pairs

The posterior probability as described in section 1.3.2 considers a single pair of residues i and j. A further extension will be needed in section 3.2.1, where two pairs (i,j) and (k,l), $i<k$, $j<l$, are taken into account (see also Figure 1.6):

$$P(M_i^q \diamond M_j^t,\ M_k^q \diamond M_l^t | q, t). \tag{1.13}$$

Due to the computational complexity of calculating eq. (1.13) exactly, we approximate

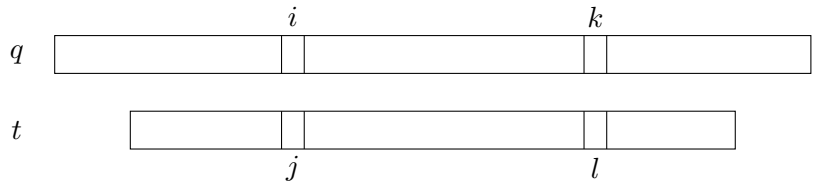


Figure 1.6.: Aligning two pairs of residues: (i,j) and (k,l). Calculating posterior probabilities for all pairs is time-consuming, so we use the approximation in eq. (1.14)

it as:

$$P(M_i^q \diamond M_j^t,\ M_k^q \diamond M_l^t | q, t) \approx \begin{cases} \min\{P(M_i^q \diamond M_j^t), P(M_k^q \diamond M_l^t | q, t)\} & \text{if } k-i = l-j \\ P(M_i^q \diamond M_j^t) \cdot P(M_k^q \diamond M_l^t | q, t) & \text{otherwise.} \end{cases} \tag{1.14}$$

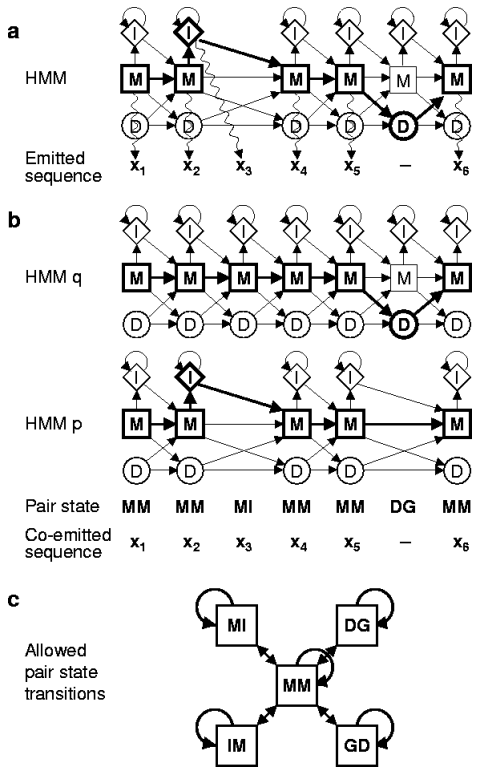


Figure 1.7.: (a) The bold arrows represent an alignment of a sequence to a profile HMM. Each column in the HMM contains a match state M, a delete state D and an insert state I. (b) Alignment of two HMMs with a path through the HMMs (bold arrows) corresponding to a sequence co-emitted by both HMMs. (c) Allowed transitions between pair states. (Figure taken from Söding (2005))

I.e. when (i, j) and (k, l) are on the same diagonal in the dynamic programming matrix, we take the minimum, and otherwise the positions are assumed to be independent and can be multiplied.

1.3.4. Pairwise HMM alignment

The structure of a profile HMM is depicted in Figure 1.7. Each column in the profile gets associated with a match state M, a delete state D and an insert state I. Both match states and insert states can emit amino acids whereas delete states can not. Consequently, there are five state pairs: MM, MI, IM, DG and GD. Gaps are treated as in sequence-sequence alignments. Finding the highest scoring path through two HMMs can again be accomplished by dynamic programming similar to Needleman-Wunsch. However, the number of state pairs is increased to five and therefore it is necessary to

fill up five matrices as follows:

$$S_{MM}(i,j) = S_{\text{aa}}(q_i,t_j) + \max \begin{cases} S_{MM}(i-1,j-1) + \log(q_{i-1}(MM)t_{j-1}(MM)) \\ S_{MI}(i-1,j-1) + \log(q_{i-1}(MM)t_{j-1}(IM)) \\ S_{IM}(i-1,j-1) + \log(q_{i-1}(IM)t_{j-1}(MM)) \\ S_{DG}(i-1,j-1) + \log(q_{i-1}(DM)t_{j-1}(MM)) \\ S_{GD}(i-1,j-1) + \log(q_{i-1}(MM)t_{j-1}(DM)) \end{cases} \tag{1.15}$$

$$S_{MI}(i,j) = \max \begin{cases} S_{MM}(i-1,j) + \log(q_{i-1}(MM)t_j(MI)) \\ S_{MI}(i-1,j) + \log(q_{i-1}(MM)t_j(II)) \end{cases} \tag{1.16}$$

$$S_{DG}(i,j) = \max \begin{cases} S_{MM}(i-1,j) + \log(q_{i-1}(MD)) \\ S_{DB}(i-1,j) + \log(q_{i-1}(DD)). \end{cases} \tag{1.17}$$

Here, $q_i(XX')$ and $t_j(YY')$ denote the transition probabilities to go from state X or $Y \in \{M, I, D\}$ in column i or j to a state X' or $Y' \in \{M, I, D\}$. As for the Smith-Waterman algorithm, by adding a zero as a sixth option in equation (1.15), a local alignment can be calculated. Again, the optimal alignment can be constructed by backtracing from the cell with the highest score.

1.3.5. HHsearch

Söding (2005) has implemented a pairwise HMM-HMM alignment tool called HHSEARCH. Both the query and all database templates are encoded by Hidden Markov Models and their alignment is calculated as outlined in the previous section. In addition to comparing amino acid frequencies by the column score (equation (1.7)), HHSEARCH also scores secondary structure. If DSSP states are missing, PSIPRED is employed to predict secondary structure.

To further refine the alignment quality, HHSEARCH exploits the finding of Pei and Grishin (2001). They observed that homologous sequences often share clusters of similar regions, i.e. ranges of consecutive columns. Having determined an optimal scoring alignment path, HHSEARCH corrects for these clusters by calculating the correlation of consecutive match-match states. Specifically, if the l-th state is MM and $i(l)$, $j(l)$ are

the corresponding columns in q and t, then the autocorrelation function is defined as:

$$g(d) = \sum_{l=1}^{L-d} S_l \cdot S_{l+d}, \tag{1.18}$$

where S_l is the column score for state l, and d is a fixed sequence separation value. Homologous sequences are expected to exhibit positive values of $g(d)$ for low ds. Consequently, the final alignment score is adjusted by the correlation score:

$$S_{\text{corr}} = w_{\text{corr}} \sum_{d=1}^{4} g(d). \tag{1.19}$$

w_{corr} has been trained on a small test set of 317×317 pairwise alignments.

Like most alignment tools, HHsearch includes an offset parameter into its scoring function. It is added to the column score S_{aa} and was found to be slightly negative (-0.1). Presumably, it prevents long but non-homologous alignments from accumulating small scores.

When a profile is quite sparse, some columns might get assigned a frequency of zero for certain amino acids. Obviously, such extreme values originate from an inadequate sampling of sequences in the Msa. For instance, the database might be biased towards certain protein families. To compensate for such effects, HHsearch adds pseudocounts to the profiles. Depending on the profile's diversity, the amount of pseudocounts gets adapted. See Angermüller et al. (2012) for more details.

1.4. Alignment features

To assess the quality of an alignment, HHsearch provides the user with several output values. Table 1.1 gives an overview of some important alignment features that will be used throughout this thesis.

Table 1.1.: Alignment features: the probability, raw score, ss score and similarity are calculated by HHsearch.

Feature	Description
Probability	The Probability of a template to be a true positive: For the probability of being a true positive, the secondary structure score in column SS is taken into account, together with the raw score. True positives are defined to be either globally homologous or they are at least homologous in parts, and thereby locally similar in structure. More precisely, the latter criterion demands that the MAXSUB score between query and hit is at least 0.1. In almost all cases the structural similarity will we be due to a global or local homology between query and template.
Sum of posteriors	The sum of all posterior probabilities along the alignment A between query q and template t, i.e. $$\text{SoP} = \sum_{(i,j)\in A} P(q_i \diamond t_j | q, t),$$ where the posterior is calculated as in section 1.3.2. Since SoP is heavily length dependent, it is usually divided by the query length $\|q\|$.
Raw score	The raw score is what comes out of the (Viterbi) Hmm-Hmm alignment excluding the secondary structure score. Informally speaking, it is the sum over the similarities of aligned profile columns minus the gap penalties.
SS score	The secondary structure score. This score tells how well the PsiPred-predicted (3-state) or actual DSSP-determined (8-state) secondary structure sequences agree with each other. PsiPred confidence values are used in the scoring, low confidences get less statistical weight.
Similarity	The Similarity is the arithmetic mean of the substitution scores between the aligned residue pairs from the query and template.

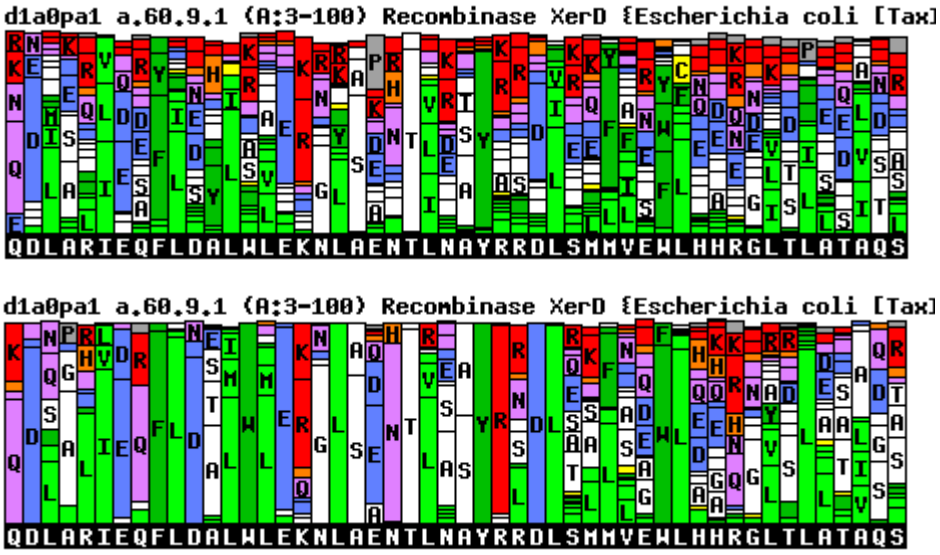


Figure 1.8.: Effective number of sequences: the lower profile is generated from the upper profile by reducing the effective number of sequences from 7.9 to 3. After the filtering, the lower profile is much more sparse and many columns are dominated by a single amino acid.

1.4.1. Effective number of sequences

The effective number of sequences at column i of a multiple alignment is calculated on the subalignment M_i formed by all sequences with a residue in column i and by all columns with at most 10% terminal gaps in these sequences. A terminal gap is a gap that lies either to the left or to the right of the entire sequence. For each column j of M_i we calculate amino acid frequencies $p(j, a)$, using the Henikoff sequence weighting scheme. Then the number of effective sequences is:

$$N(i) = \exp\left(-\frac{1}{L_i}\sum_{j\in M_i}\sum_{a=1}^{20} p(j,a)\cdot\log p(j,a)\right). \tag{1.20}$$

Here, L_i is the number of columns in M_i.
We measure the amount of homology information in an alignment by its effective number of sequences. Higher numbers mean more diverse alignments and profiles, see Figure 1.8.

1.4.2. TMscore

The wealth of structural data available today requires automatic processing pipelines. In structure prediction, models can be produced easily by fully automated servers. Evaluating the quality of these models by hand is quite labour-intensive and thus a range of scores has been devised to carry out the task. Simply calculating the root mean square deviation is disadvantageous because outliers are penalized heavily even if the overall fold is correct. Ideally, a score should be independent of the query length and differentiate between a random and a statistically significant prediction. Zhang and Skolnick (2005b) have proposed the template modeling score (TMscore) which has been designed to be both length independent and strongly correlated to full length models. Based on the Levitt-Gerstein score (Levitt and Gerstein, 1998), the TMscore is defined as:

$$\text{TMscore} = \max\left[\frac{1}{L_{\text{target}}}\sum_{i=1}^{L_{\text{ali}}}\frac{1}{1+\left(\frac{d_i}{d_0(L_{\text{target}})}\right)^2}\right]. \tag{1.21}$$

Here, L_{target} is the length of the target protein, L_{ali} is the number of aligned residues, d_i is the distance between the i-th pair of aligned residues and

$$d_0(L_{\text{target}}) = 1.24\sqrt[3]{L_{\text{target}}-15} - 1.8$$

is a length dependent normalization factor and can be interpreted as the mean distance between an aligned pair of residues in randomly related proteins of length L_{target}. The maximization is with respect to all superpositions of the target and template protein.

TMalign (Zhang and Skolnick, 2005b) is a tool that finds an alignment between two structures such that their TMscore is maximized. In subsequent sections we will take TMalign alignments as a gold-standard reference.

1.5. Homology modeling

It is known that protein structure is more conserved than its sequence. Especially in the twilight zone and below, homologous protein sequences have accumulated so many substitutions that it is often difficult to detect their similarity on sequence level while their structures retain their fold. Due to physical constraints in the structure space (Laskowski et al., 1993), there are fewer distinct folds than sequences, i.e. the sequence space is much larger than the structure space.

Illergard et al. (2009) investigated the mapping between sequence and structure in a set of structurally aligned protein domains. They found that structure is three to ten times more conserved than sequence in protein cores irrespective of the measure of structural change.

Homology modeling makes use of this structural conservation of homologous proteins. It refers to constructing a 3D model for a query sequence (target) from an alignment of this query to templates with known structures. The size of databases like PDB and SCOP is steadily increasing and most of the fold space is already covered (Zhang and Skolnick, 2005a). As a consequence, homology modeling is by far the most successful approach to structure prediction. Its success is reflected by the fact that all top-performing servers in the Critical Assessment of protein Structure Prediction (CASP, Moult et al. (2014)) rely on homology modeling and outperform all ab-initio methods.

Part I.

Homology modeling

2. HHpred and structure prediction

2.1. Overview

As homology modeling has proven to return reliable models and the number of solved protein structures steadily increases, numerous groups have developed their own prediction pipeline. Even if the basic workflow is often similar and consists of (1) building a database of proteins with known structures, (2) searching the database for homologous templates, (3) aligning the query to some selected templates and (4) making a 3D model from this alignment, the details in each step differ considerably from group to group. Depending on the group's background knowledge and field of expertise, they focus on the alignment step, refinement, template selection or other topics like model selection. Alternatively, meta-servers have become popular. They consist of a number of individual servers and combine their predictions into a consensus model.
In our group, we have set up a fully automatic structure prediction server called HHpred. Its core components consist of our sensitive and accurate alignment tools HHblits and HHsearch (Remmert et al., 2012; Söding, 2005). In particular, we build database MSAs with HHblits and search against this database with HHsearch (steps (1) and (2) from above). Template selection will be described in section 3.2.3 in detail. Finally, a 3D model is calculated from the alignment (step (4)) with Modeller. In the next sections, we will give a short introduction into Modeller and outline the HHpred pipeline.

2.2. Modeller

Modeller (Sali and Blundell (1993)) is a protein structure modeling program that takes as input an alignment between a query sequence and a set of templates with known structure, and automatically calculates a 3D model for the query.
To build such a model, Modeller first extracts spatial restraints according to the alignment, second renders each restraint into a mathematical form and third runs an optimization schedule to optimally satisfy all restraints (see Figure 2.1). Even if homol-

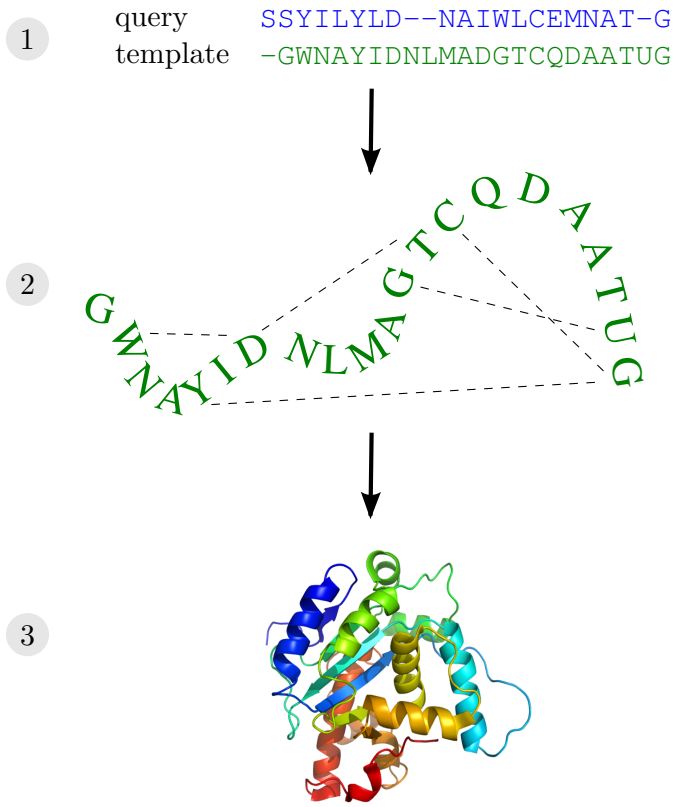


Figure 2.1.: MODELLER flowchart: 1) MODELLER needs as input an alignment between the query and one (or several) templates with known structure, 2) Based on the alignment and the template structures, a list of spatial restraints is extracted, 3) The final model is generated by satisfying all spatial restraints as well as possible, i.e. by calculating the most probable model.

ogous proteins often have similar structure (section 1.5), the quality and reliability of an alignment can vary significantly. To account for these uncertainties, each restraint is specified by a probability density function (pdf). For example, the distance between two given C_α atoms in the query is supposed to be similar to the corresponding distance in the templates and will be modeled by Gaussian distributions (see Figure 2.2). If a sufficient number of distance restraints is specified, the 3D structure of a protein is well determined. MODELLER assumes all constraints to be independent of each other and the joint probability function for a query is thus approximated by multiplying all individual pdfs:

$$P(\text{query structure} \mid \text{templates, alignment}) = \prod_i \text{pdf}_i \tag{2.1}$$

This overall product involves besides template derived distance restraints also bond length-, dihedral- and angle potentials to ensure chemically valid structures.
In an optimization step, MODELLER determines the best query structure by maximizing the joint probability in eq. (2.1), or, equivalently, minimizing its negative logarithm.

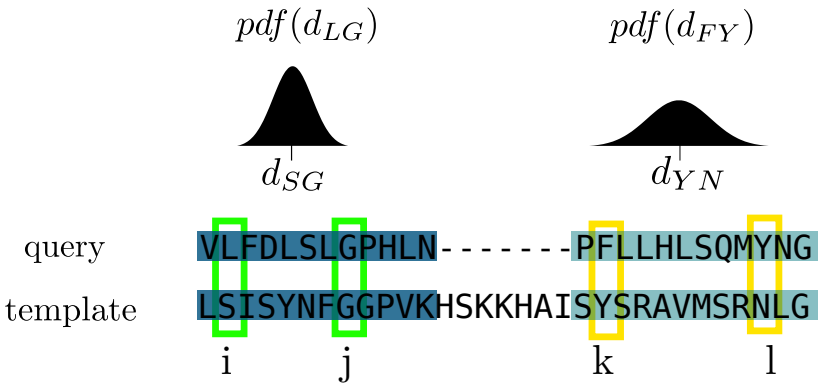


Figure 2.2.: Probability density functions (pdfs) for distance restraints: the alignment at query positions i and j is quite confident (dark turquois) whereas at k and l it is uncertain (light turquois). Therefore, the pdf for the distance between the query residues L and G gets peaked an has its maximum at the template distance d_{SG} and the pdf for the distance between F and Y gets flat and centered at d_{YN}.

By default, MODELLER implements the following flowchart of comparative modeling: In a first step, an initial structure is generated by averaging the coordinates of all template structures. Then, a long list of restraints, e.g. C_α-C_α distances, is extracted from the alignment and the template structures. Each restraint is specified by its corresponding pdf as mentioned above. Stereochemical restraints ensure a valid chemical conformation of the model (e.g. valid bond-lengths and angle potentials) and are added to the list. Finally, an optimization schedule is executed to find a model structure that maximizes the objective function or, equivalently, that satisfies the list of restraints as well as possible.

In contrast to template derived distance restraints, bond-length distributions are very sharply peaked to ensure they are strictly kept. Distance restraints, in turn, are subdivided into four classes according to their atom types: C_α-C_α backbone, N-O backbone, side chain-main chain, side chain-side chain.

Thanks to its modular setup, it is straightforward to extend MODELLER or replace any default restraints by customized ones.

Whereas template independent restraints are indispensable for fine tuning the model conformation, template derived distance restraints make up the majority of restraints and mostly define the structural characteristics of a target protein structure, i.e. its overall fold. Consequently, their accurate modeling is crucial for model quality.

2.3. HHpred pipeline

The flowchart in Figure 2.3 gives a schematic overview of the HHpred pipeline as implemented for the CASP9 and CASP10 competitions.
In the following, we present only a summary of the pipeline without going into all details. We will provide references to sections later in this thesis whenever appropriate.
Starting with an amino acid sequence, HHblits builds a multiple sequence alignment for the query. HHsearch (Söding, 2005) then scans the PDB (filtered to 70% pairwise sequence identity) for homologous sequences (templates) and returns a list of templates together with their alignments to the query. A pre-selection of these is then iteratively filtered to identify the ones which are most similar to the query (see Hildebrand et al. (2009) for details). We use the heuristic template selection scheme presented in section 3.2.3 to identify a set of hits for modeling. Next, Modeller's default distance restraints are replaced by the two-component mixtures of Gaussians from section 3.2.1. Finally, we start Modeller to build three models in parallel. Each model gets an energy value by Modeller and the structure with minimal energy is finally submitted.
Apart from generating very accurate models, we intended to provide a server which – in contrast to most of the top-performing competitors – had a preferably low median running time. Consequently, we omitted any time consuming additional de novo or loop modeling protocols and refinement steps.

2.4. CASP

Protein structure prediction has a long tradition in bioinformatics. Thus, over the years, a large number of servers for building three-dimensional (3D) models has been developed. In order to objectively test and assess their performance, a community-wide, double-blind experiment takes place every second year.

2.4.1. Overview

The idea is to provide all participating groups with amino acid sequences (targets) whose structures have been solved experimentally but have not yet been published. All groups are then challenged to predict their 3D structures by either running their automatic prediction pipeline or by manual editing. Subsequently, all final models are sent back to the organizers. In the end, all predictions are assessed by comparing them to the gold standard, i.e. the experimental structures. In the end, rankings (e.g. by group or best overall model) can be generated by assessing all predictions according to different

	TBM	FM	total
CASP9	118	29	147
CASP10	111	15	126

Table 2.1.: CASP9/CASP10 number of targets in Template Based Modeling (TBM) category versus Free Modeling (FM).

quality measures.

All targets are subdivided into two categories: *template based* (TBM) and *free modeling* (FM), depending on whether there existed a suitable template in current databases before the target structures became public, or not. Typically, the number of template free targets is considerably lower than template based targets. As present-day databases are continuously growing, this trend will probably continue. Yet, sensitive and accurate alignment tools are becoming increasingly important to identify the best homologous templates in the vast set of candidates.
In the next section, we briefly summarize HHpred's performance in the last two CASP experiments.

2.4.2. Results and Discussion

During CASP9 (CASP10), structures for targets with a total of 147 (126) domains had to be predicted. Most of them fell into the TBM category, see Table 2.1. Tables 2.2 and 2.3 show the top 10 performers in CASP9 (79 servers in total) and CASP10 (69 servers in total) as officially ranked on the CASP website. HHpredA, HHpredB and HHpredC ran the same pipeline except of a small bug in HHpredC. We had intended to integrate extra structural scores into HHpred but were hindered by time constraints. Yet, in CASP10 these new scores were implemented in HHpred-thread by our then Master student Markus Meier.
In both CASP9 and CASP10 HHpred was ranked among the top ten servers. A closer look at the mean runtime reveals that HHpred was more than 450 times faster than all other high scored servers.
HHpred's main field of application is in TBM because it lacks any de novo modeling functionality and is completely relying on template information. When removing all FM targets from Tables 2.2 and 2.3, we get the results in Tables 2.4 and 2.5. In CASP9, HHpred could even win the TBM category. However, most of the top performing servers are quite on par with each other and no single server could clearly outperform all com-

petitors. The only exception is in terms of runtime: here, HHpred is unrivaled.

Table 2.2.: Official CASP9 results for the top 10 servers (TBM and FM targets). The table is sorted with respect to the sum Z-score column. Runtime is given in mean minutes per target.

CASP9 TBM + FM results						
rank	**server**	**dom**	**sum Z-score**	**avg Z-score**	**avg GDT-TS**	**time [min]**
1	QUARK	147	115.788	0.788	62.675	3358.736
2	Zhang-Server	147	113.242	0.77	62.765	3347.378
3	RaptorX-MSA	147	103.27	0.703	61.774	3586.239
4	RaptorX	147	103.01	0.701	61.731	3587.406
5	RaptorX-Boost	147	99.845	0.679	61.453	3587.241
6	HHpredB	147	93.104	0.633	59.528	4.334
7	HHpredA	147	93.104	0.633	59.528	4.405
8	HHpredC	147	91.821	0.625	59.361	4.398
9	Seok-server	147	89.542	0.609	60.158	3735.85
10	MULTICOM-CLUSTER	147	88.944	0.605	59.987	1030.446

Table 2.3.: Official CASP10 results for the top 10 servers (TBM and FM targets). The table is sorted with respect to the sum Z-score column. Runtime is given in mean minutes per target.

CASP10 TBM + FM results						
rank	**server**	**dom**	**sum Z-score**	**avg Z-score**	**avg GDT-TS**	**time [min]**
1	Zhang-Server	126	111.874	0.888	60.601	2457.093
2	QUARK	126	105.531	0.838	60.204	2462.948
3	BAKER-ROSETTA	126	87.787	0.697	57.542	2977.735
4	RaptorX-ZY	126	85.964	0.682	58.43	4250.788
5	RaptorX	126	82.911	0.658	58.055	3606.894
6	TASSER-VMT	126	82.016	0.651	57.382	3307.054
7	PMS	126	78.113	0.62	57.559	4378.698
8	HHpred-thread	124	77.339	0.624	58.402	11.766
9	HHpredA	126	76.748	0.609	57.563	6.486
10	HHpredAQ	126	75.904	0.602	57.295	6.635

Table 2.4.: Official CASP9 results for the top 10 servers in the TBM category. The table is sorted with respect to the sum Z-score column. Runtime is given in mean minutes per target.

CASP9 TBM results						
rank	**server**	**dom**	**sum Z-score**	**avg Z-score**	**avg GDT-TS**	**time [min]**
1	HHpredA	118	87.577	0.742	69.164	4.706
2	HHpredB	118	87.577	0.742	69.164	4.854
3	Zhang-Server	118	86.737	0.735	70.684	3304.536
4	HHpredC	118	86.294	0.731	68.957	4.949
5	QUARK	118	84.157	0.713	70.251	3324.657
6	RaptorX-MSA	118	83.927	0.711	70.199	3553.134
7	RaptorX	118	83.256	0.706	70.087	3549.116
8	RaptorX-Boost	118	81.559	0.691	69.822	3552.941
9	Seok-server	118	76.990	0.652	68.857	3718.563
10	MULTICOM-CLUSTER	118	68.990	0.585	849.689	849.689

Table 2.5.: Official CASP10 results for the top 10 servers in the TBM category. The table is sorted with respect to the sum Z-score column. Runtime is given in mean minutes per target.

CASP10 TBM results						
rank	**server**	**dom**	**sum Z-score**	**avg Z-score**	**avg GDT-TS**	**time [min]**
1	Zhang-Server	111	92.057	0.829	65.536	2393.143
2	QUARK	111	89.207	0.804	65.171	2393.881
3	BAKER-ROSETTASERVER	111	80.144	0.722	62.482	2976.484
4	RaptorX-ZY	111	78.842	0.710	63.562	4225.730
5	RaptorX	111	75.422	0.679	63.120	3531.615
6	TASSER-VMT	111	69.048	0.622	62.168	3236.028
7	HHpredA	111	68.757	0.619	62.799	6.498
8	HHpredAQ	111	67.513	0.608	62.466	7.079
9	PMS	111	67.218	0.606	62.422	4378.698
10	HHpred-thread	110	66.699	0.606	63.272	11.766

Figure 2.4 summarizes the performance of all modeling servers in both CASP9 and CASP10. The HHpred servers are colored in red and all other top performers using

HHSEARCH in green. Employing the new distance restraints of section 3.2.1 in CASP10 increased the cumulative GDT-TS score by 3% (default: 57.50, new restraints: 59.33).

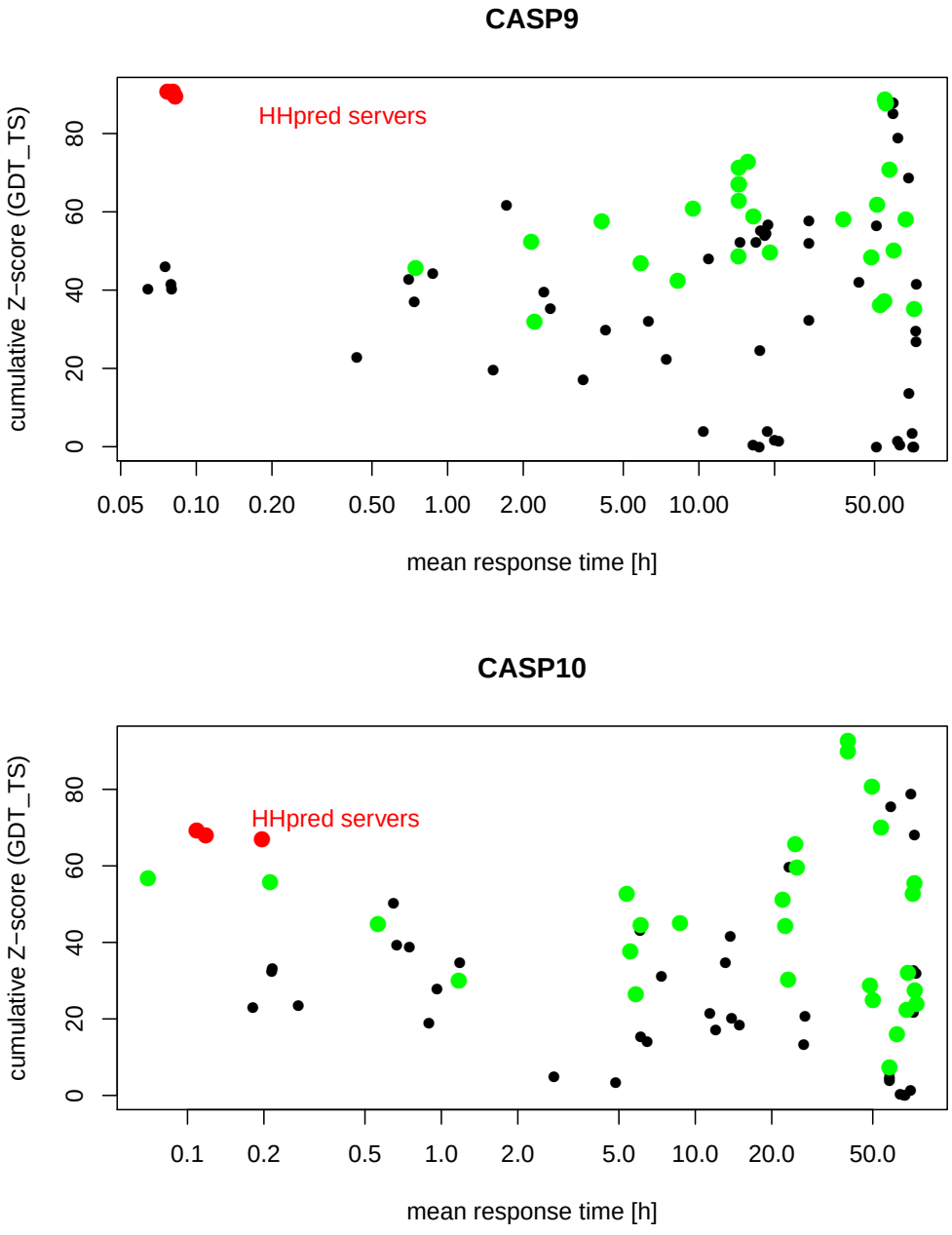


Figure 2.4.: CASP9 and CASP10 results for all servers in the TBM category. HHPRED is in red, in green are all other servers that use HHSEARCH.

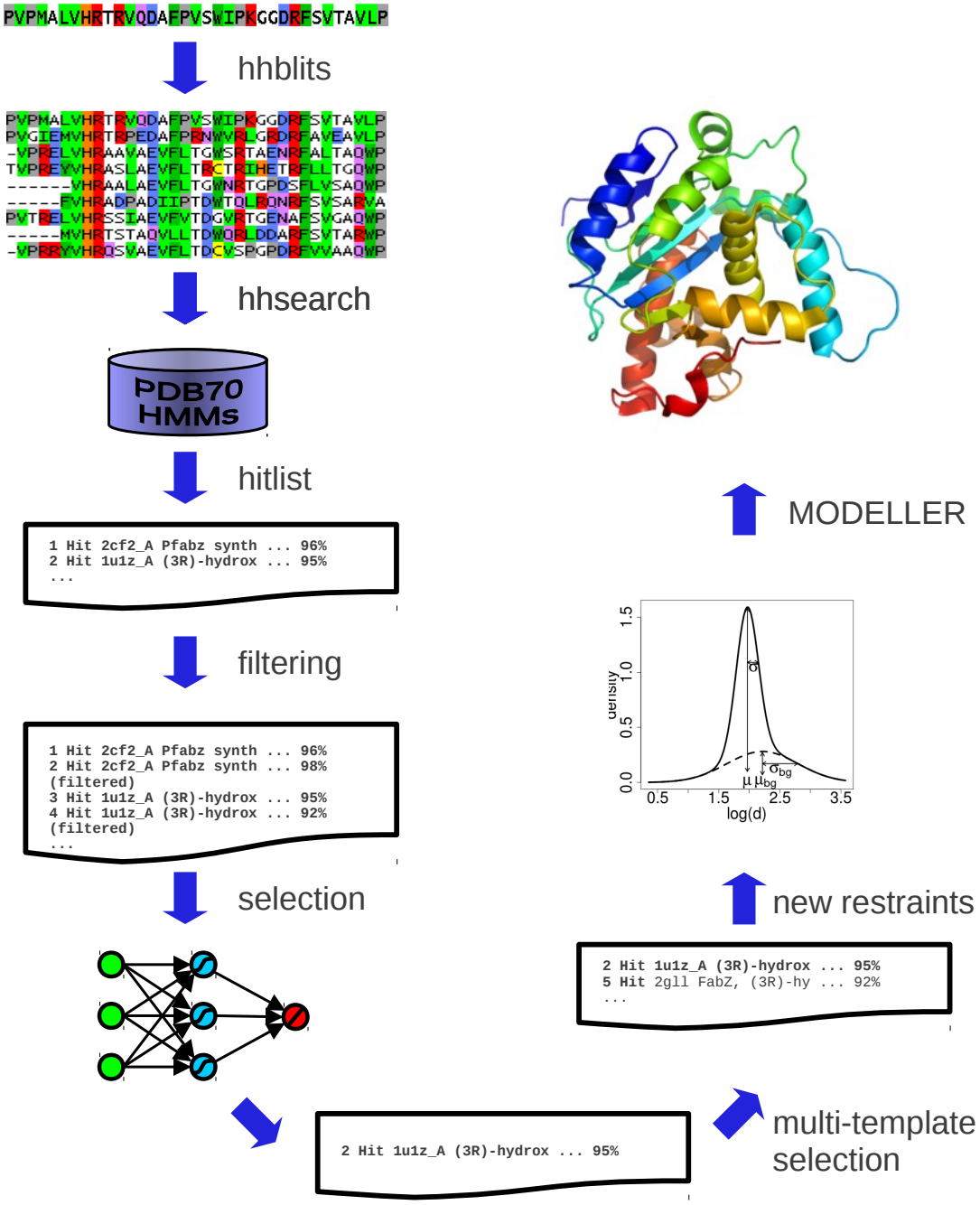


Figure 2.3.: HHPRED flowchart: starting from an amino acid sequence, a profile is built with HHBLITS. Next, HHSEARCH scans the PDB70 for templates. After filtering and template selection we generate the final alignment between the query and a selection of templates. Based on this alignment, we compute our new distance restraints. Finally, we start MODELLER – extended by our new restraints – and calculate a 3D model.

3. Mixture Gaussian distance restraints

3.1. Introduction

Homology modeling methods build a three-dimensional ($3D$) model for a query sequence based on an alignment to template proteins of known structure. A wide range of applications such as function determination, interaction prediction and drug screening is opened up when a suitable $3D$ structure is available.
By taking advantage of the fact that evolutionarily related proteins share similar structure, homology modeling is the most accurate approach to structure prediction. De novo methods still cannot provide reliable models, especially for larger proteins. Even though there is some remarkable progress in the field (Marks et al. (2012); Nugent and Jones (2012)), all top performers in the Critical Assessment of Structure Prediction (CASP) heavily rely on template information (Kinch et al. (2011); Mariani et al. (2011)). Nowadays databases such as PDB or SCOP contain a multitude of solved protein structures (Berman, 2008; Lo Conte et al., 2000) and cover most of the fold space (Zhang and Skolnick (2005a)). But it is still challenging to identify and align templates correctly (Peng and Xu (2011)).
Xu (2005) proposed a neural network (NN) for picking a suitable template. Most of the successful prediction servers, however, perform multi-template modeling. Larsson et al. (2008) provided a study in which they assessed the effect of multiple templates on model quality and concluded that most of the gains are due to increased coverage.
To our knowledge, no theoretically well founded strategy for multi-template protein homology modeling has been developed so far, which contrasts its wide spread use in virtually every successful prediction pipeline. Contrary to single template selection, picking multiple templates is fundamentally complicated by complex dependencies between all selected structures. Accordingly, current methods are mostly based on heuristics (Cheng, 2008; Fernandez-Fuentes et al., 2007; Peng and Xu, 2011). Some methods (Wang et al. (2010); Zhang (2008)) rather build a set of models based on several different template lists and then post-select a final model according to some quality measure (Zhang and Skolnick (2004)). Consequently, multi-template selection is still a problem to be solved

by more sophisticated methods. Template selection is, however, less critical when the structure modeling tool can properly deal with multiple templates.
Different approaches to combine several templates of been implemented: ModSeg/ENCAD (Levitt, 1996) copies template coordinates and bridges gaps by short fragments that match the framework of the target structure. SWISS-MODEL (Schwede et al., 2003) generates a core model by averaging template backbone atom positions and uses constraint space programming to fill gaps. Both methods decrease deviations in protein structure geometry by steepest descent energy minimization. MODELLER was the first program to dispense with crude heuristics by resorting to a statistical approach. In this context, MODELLER maximizes a probability density function such that a list of (spatial) restraints is satisfied as well as possible (section 2.2). Even if it achieved similar results in benchmarks (Dalton and Jackson (2007); Wallner and Elofsson (2005)), MODELLER performed marginally better than the others, and furthermore, its coherent underlying statistical model is flexible and easily extensible.
Several extensions of the default MODELLER pipeline exist already. Most of them are concerned with refined energy functions (Joo et al. (2009)) or loop modeling (Fiser et al. (2000)). One of its principal elements, the template dependent distance restraints, has been unchanged so far and we are not aware of any further developments. This is in contrast to the fact that nearly two thirds of the top 20 CASP performers are relying on MODELLER (CAS (2010)). All these would profit from a more accurate model building tool.
In this work, we provide a solid statistical framework for multi-template modeling. First, we examine MODELLER's distance restraints and replace them by mixtures of two Gaussians that allow a comprehensive treatment of multiple templates. Second, we introduce statistical template weights to correct for redundancies within the templates and combine them with our new restraints. Finally, we introduce a new strategy for multiple template selection.
We assessed all these steps on a benchmark set and refer to section 2.4, where we report about how these methods (implemented within our webserver HHPRED) performed in two recent CASP competitions.

3.2. Methods

3.2.1. Alignment features

When a given query q is searched with HHsearch against a database of templates t, each pairwise alignment $A(q,t)$ is characterized by a number of features to assess its quality. Every aligned pair of residues (i,j) in A gets a posterior probability $P(i \diamond j|q,t)$ ($P(i \diamond j)$ for short) of being correctly aligned. Furthermore, we define

$$\text{sumProbs} := \sum_{(i,j)\in A(q,t)} P(i \diamond j|q,t). \tag{3.1}$$

The posterior can also be extended to two pairs $(i,j),(k,l)$: $P(i \diamond j, k \diamond l)$ (see section 1.3.3). In addition, there are global features, such as the probability of homology $P_{\text{hom}}(t)$, the raw score, similarity and secondary structure score (ss-score) (see Table 1.1). Alignment greediness is controlled by the 'mact' parameter which is a posterior probability threshold for the maximum accuracy re-alignment (Durbin et al., 1998).

Structure Modeling

Modeller is used for comparative modeling of 3D protein structures by satisfaction of spatial restraints. It takes as input an alignment between a query sequence and a number of templates with known structure. In a next step, it generates a list of spatial restraints based on the alignment. Each restraint is expressed as a probability density function (pdf) and acts on e.g. C_α-C_α distances or dihedral angles. To satisfy all spatial restraints as well as possible, Modeller maximizes the product of all restraint pdfs and thereby calculates the most probable model. See section 2.2 for more details.

Default distance restraints

Most restraints act on distances between two atoms (e.g. the distance between any two backbone C_α atoms, C_α-C_α). We denote by d the distance between two atoms in the query and by d_t the corresponding distance in template t according to the alignment. Sali and Blundell (1993) built 3D models based on a set of pairwise training alignments and then calculated their distance differences $d - d_t$. Here, d is taken from the query's PDB entry. To find the distribution of $d - d_t$, they inspected histograms of $d - d_t$. These histograms demonstrated that Gaussians with a mean of zero and a standard deviation that is dependent on the alignment quality can approximate the distributions. The functional form of the standard deviation had four variables: 1) the distance in the

template, 2) the fractional sequence identity of the two aligned sequences, 3) the average solvent accessibility of the two aligned residues and 4) the average distance from a gap.

New distance restraints

Sali and Blundell (1993) aligned homologous proteins based on their 3D structure in the training set to find the distribution of $d - d_t$. In practice, however, the query structure is unknown. Hence, we used sequence profile alignments to generate distance difference histograms and omitted any structural information on the query side. Moreover, since spatial distances are restricted to be positive, we modeled logarithmized distances $\log(d) - \log(d_t)$. Figure 3.1 shows four histograms of $\log(d) - \log(d_t)$ together with their fits. Here, all models were built based on pairwise HMM-HMM alignments, i.e. only sequence and no structural information is used.
Figures 3.1 A to C all depict C_α-C_α distance differences. However, the quality of the alignments underlying the 3D models differs in each case: in A, it is low as reflected by both a low similarity and posterior probability. B and C are based on medium and high quality alignments, respectively. D differs from A only in the atom types involved and has a similar shape.
Taken together, these histograms suggest that distance distributions have not only one but two components: one that describes the main peak and a second that becomes more pronounced as the alignment quality drops. Consequently, we model distance distributions as mixtures of two Gaussians with feature dependent parameters (Figure 3.2 and eq. (3.2)). In the following, we refer to the second, newly introduced component as the background.

$$\underbrace{w(\Theta)\; N(\log(d)|\mu(\Theta),\sigma(\Theta))}_{\text{correct}} + \underbrace{(1-w(\Theta))\; N(\log(d)|\mu_{bg}(\Lambda),\sigma_{bg}(\Lambda))}_{\text{background}} \tag{3.2}$$

All parameters of the mixtures are assumed to be functions of variables that describe the alignment quality: the means (μ, μ_{bg}), standard deviations (σ, σ_{bg}) and the weight (w) depend on $\Theta = (d_t,\ \text{pp}(i \diamond j, k \diamond l),\ \text{similarity})$ and $\Lambda = (\text{pp}(i \diamond j, k \diamond l),\ \text{similarity})$, respectively. See section 3.2.1 for a detailed explanation of the features.

The weight w can be regarded as a prior probability that a specific atomic distance in a template will also be adopted in the query structure. Such a weighting ensures that locally unreliable alignments will have an increased background component leading to a softer distance restraint. Badly aligned residues are unlikely to provide confident distance information for the query, so the restraint should become flat, i.e. dominated

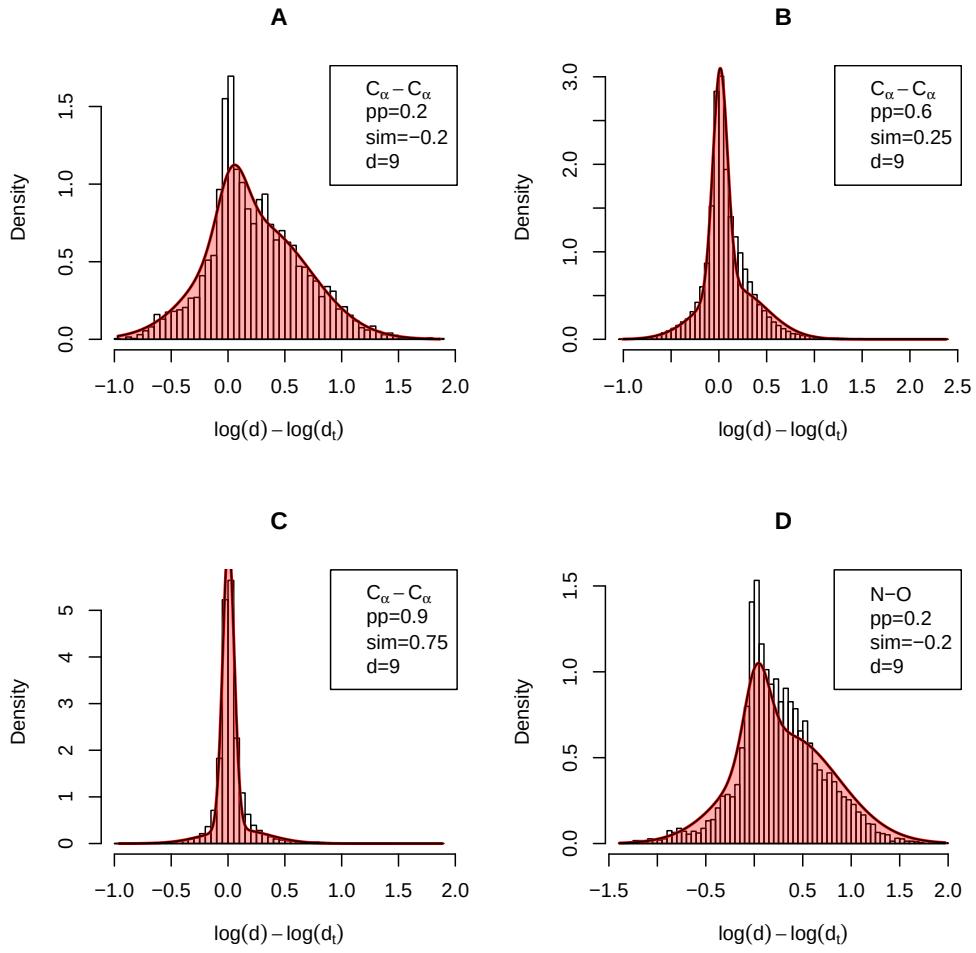


Figure 3.1.: Empirical distance distributions and their MDN fits: in A,B and C, distance differences between two C_α atoms around $9\mathring{A}$ are plotted. The alignment quality – measured by the posterior probability and similarity – increases from A to C. When it is low, a single Gaussian can no longer fit the histograms accurately since a second (background) component becomes visible (sub-figures A and D). The same effect can be observed for other types of atoms, e.g. N-O as shown in figure D.

by the background component. Similarly, a larger standard deviation $\sigma(\Theta)$ comes in hand with a low alignment quality, making the distribution wider and vice versa.

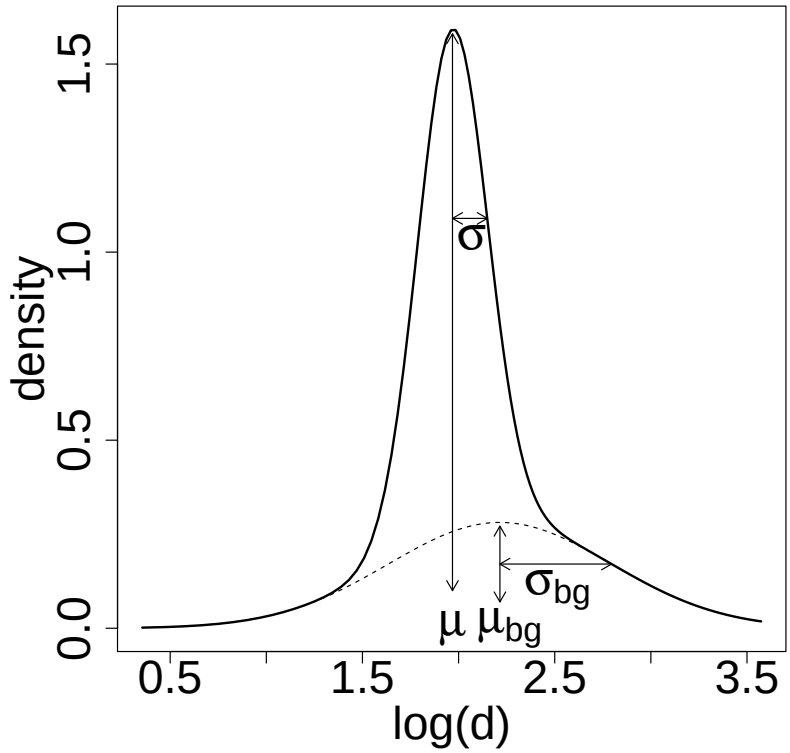


Figure 3.2.: A new distance restraint is modeled as mixture of two Gaussians. The first component with mean μ and standard deviation σ describes distances derived from correct alignments whereas the second component with mean μ_{bg} and standard deviation σ_{bg} models the background distribution resulting from erroneous alignments. Both components are weighted by w and $1-w = w_{bg}$, respectively.

Mixture density networks

To fit the feature dependent mixtures of Gaussians in eq.(3.2) we resort to mixture density networks (MDN) (Bishop (1994)). A MDN is a special kind of neural network that determines the parameters in a mixture of Gaussians as a function of given input features by a maximum likelihood approach. Figure 3.3 illustrates the MDN setup for the new distance restraints in eq.(3.2). As input features we use the spatial distance in the template d_t, the alignment posterior probability of two pairs of residues being aligned

correctly and the sequence similarity. To make the background parameters independent of d_t, connections between these nodes are removed.

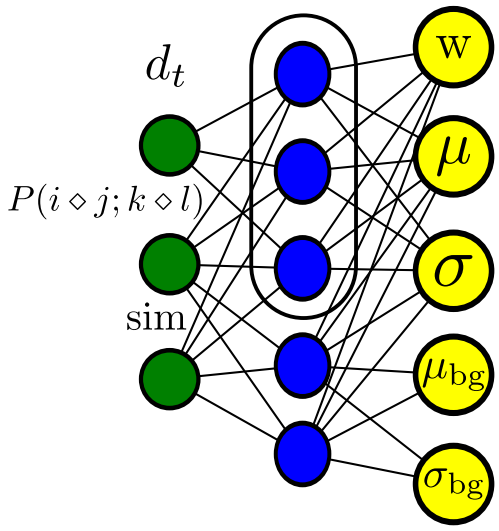


Figure 3.3.: Mixture density network for distance restraints: the network gets as input the template distance, the posterior probability for the aligned residue pairs and the sequence similarity, and calculates means, standard deviations and weights for mixture of Gaussians. The background parameters (μ_{bg}, σ_{bg}) are not connected to the input feature d_t (no path from the hidden layers in the black box to background outputs) to make the background independent of the template distance.

MODELLER fits its single component distance distributions by specifying feature dependent standard deviations with a multi-parametric polynomial and subsequently doing a least-squares minimization.

Combination of multiple templates

In multi-template modeling, several restraints can act on a given distance. To construct a combined distance restraint $P(d|\cdot)$ from two known structures t_1, t_2 with distances d_1 and d_2 between equivalent atoms (i.e. in aligned residues), MODELLER adds up both single restraints $P(d|d_1)$ and $P(d|d_2)$ linearly:

$$P(d|d_1, d_2, s_1, s_2) = \alpha(s_1)\ P(d|d_1) + \alpha(s_2)\ P(d|d_2), \tag{3.3}$$

where s_1, s_2 are the average residue neighborhood differences in the first and second alignment, respectively, and $\alpha(s_1)$, $\alpha(s_2)$ are the corresponding weights. Such a weighted sum can be extended to any number of templates. A simple parametric model is employed for $\alpha(s_1), \alpha(s_2)$ and it is fit based on a procedure described in (Sali and Blundell (1993)). Yet, such a sum corresponds to a one-hot encoding of a combined restraint, where only a single template can generate a distance.

The two component distance restraints allow for a different and consistent way to combine multiple templates. Here, the background component will turn out to be of major importance. Specifically, a combined distance restraint for a distance d in the query given two templates t_1 and t_2 with their alignments A_1, A_2 to the query and the corresponding distances d_1, d_2 in t_1 and t_2, respectively, follows from applying Bayes' formula twice:

$$\begin{aligned} P(d|d_1, A_1; d_2, A_2) &= \frac{P(d_1, d_2|d, A_1, A_2)P(d|A_1, A_2)}{P(d_1, d_2|A_1, A_2)} \\ \frac{P(d|d_1, A_1; d_2, A_2)}{P(d|A_1, A_2)} &\approx \frac{P(d_1|d, A_1)}{P(d_1|A_1)} \cdot \frac{P(d_2|d, A_2)}{P(d_2|A_2)} \\ &= \frac{P(d|d_1, A_1)}{P(d|A_1)} \cdot \frac{P(d|d_2, A_2)}{P(d|A_2)} \end{aligned} \tag{3.4}$$

Here, the second equation is only an approximation since it requires independence of the templates given the query. Equation (3.4) can easily be extended to more than two templates.

In chapter 3.2.2, the assumption of independence will be corrected for by introducing template specific weights.

$P(d_k|d, A_k)$, $k \in \{1, \ldots, |\mathcal{T}|\}$ in the numerator of eq. (3.4) are modeled by the two component mixtures given in eq. (3.2). $P(d_k|A_k)$ corresponds to the background component in $P(d_k|d, A_k)$. Dividing by the background has two effects: first, it prevents the background to become dominant when the individual background components of all $P(d|d_k, A_k)$ are multiplied. Second, the negative logarithm of MODELLER's default distance restraints is quadratic in d. i.e. unsatisfiable restraints can lead to arbitrarily high values during optimization. Dividing by the background avoids this quadratic increase as the logarithm of $\frac{P(d|d_k, A_k)}{P(d|A_k)}$ has flat tails. Similarly, when a distance in the query drastically differs from the one in the template, there is no information about query distance. This lack of information is also reasonably reflected by flat tails.

Combining two component distance restraints as in eq.3.4 dissolves contradictory restraints and reinforces consistent restraints. Let us assume we are given two contradic-

tory distance restraints for the same pair of atoms but from two different templates: the first is supposed to be derived from badly aligned residues and has a pronounced background component, whereas the second is more confident. Figure 3.4 illustrates this situation for both MODELLER's default restraints (A and B) and the two component Gaussians (C and D). The background component of the unconfident restraint (Figure 3.4 C, green) is more pronounced, which makes the confident restraint (blue) dominating the combined restraint as given by eq. 3.4 (Figure 3.4 D).

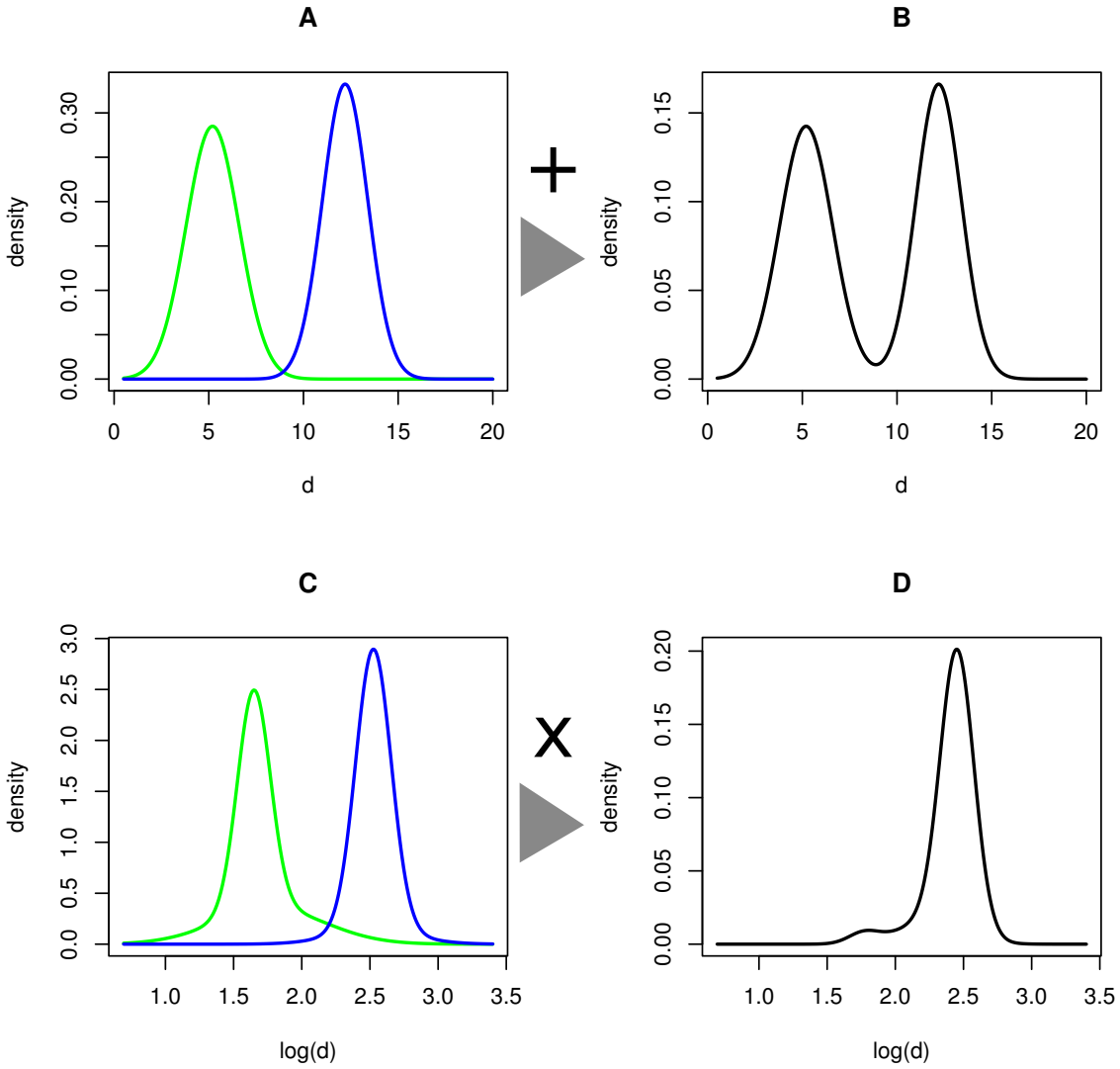


Figure 3.4.: Combining distance restraints: given two restraints (in green and blue) for the same query distance but derived from two different templates, MODELLER combines them by a weighted sum ($A \rightarrow B$) whereas the new restraints are multiplied ($C \rightarrow D$). Due to the more pronounced background component in green, sub-figure C, the more confident blue distribution will dominate the resulting restraint in sub-figure D.

On the other hand, when both restraints peak at the same distance, multiplication will reinforce the resulting restraint by making it more narrow and leading to a sharper distribution. Such a sharper restraint conforms with the distance information provided by the templates.

3.2.2. Template weighting

As already mentioned in section 3.2.1, the new distance restraints framework assumes templates to be independent of each other. In a simple case with three templates where two of them are closely related to each other whereas the third is more distantly related, this independence assumption is already violated (see Figure 3.5). Giving all three

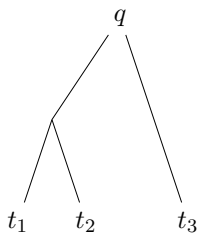


Figure 3.5.: Evolutionary relationship between a query q and three templates t_1, t_2, t_3. t_1 and t_2 are closely related to each other and should be down-weighted with respect to t_3.

templates the same weight ignores the dependencies given by the tree (Altschul et al., 1989). We correct for the violation of the independence assumption by giving each template an individual weight. We first construct a tree and root it at the query. The goal is to connect all leave nodes directly to the query and let the weights assess their direct influence on the query.
Therefore, we define a basic reconstruction step that transforms a given subtree with an internal node into an equivalent one where all leaves are directly dependent on the root by finding appropriate weights, see Figure 3.6. This basic transformation is then applied iteratively as shown in Figure 3.7.

To construct the initial tree, we use a neural network similar to the one in section 3.2.3 and predict the structural similarity between two sequences t_k and t_l (or q) and then fill the pairwise distance matrix:

$$\mathrm{dist}(t_k, t_l) = -\log(\text{pred. TMscore}(t_k, t_l)) \tag{3.5}$$

UPGMA clustering based on this matrix returns a tree T which is subsequently rearranged so that the query q becomes the root. T encodes both the relation of the templates to the query and the relations among the templates. Let d_0 be the (spatial) distance between a given pair of residues in the query and $d_1, \ldots, d_n$ the corresponding distances in the templates. Each leaf gets a weight $w_k, k = 1, \ldots, n$, with $w_0 = 1$ for the root. The probability of $d_k \to d_l$ (that a specific distance in t_k, d_k, becomes d_l in

time τ_l) along an edge in T is assumed to undergo diffusive behaviour and is therefore modeled as:

$$P(d_k \to d_l|T) \propto \exp\left(-\frac{(d_k - d_l)^2}{\tau_l}\right)^{w_l}, \tag{3.6}$$

where the time τ_l is proportional to the edge length between k and l in the UPGMA tree. The weight w_l controls the peakedness of the distribution and will represent the influence of the template on the root of the current tree.
In the basic transformation step in Figure 3.6, all leaf nodes (templates) become directly dependent on q and the new weights approximate their direct influence on the query (see Figure 3.6).

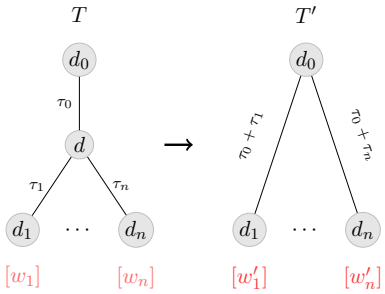


Figure 3.6.: Illustration of restructuring a given tree T with an unknown internal node d (left) into one where $d_1, \ldots, d_n$ directly depend on d_0 (right). This is done by integrating over d and finding appropriate weight $w'_1, \ldots, w'_n$ so that both trees describe the same distribution. $\tau_1, \ldots \tau_n$ correspond to distances coming from UPGMA clustering.

Formally, we want to get rid of the inner node d by integrating over all values of d and defining new weights $w'_1, \ldots, w'_n$ such that the distribution $P(d_0|d_1, \ldots, d_n; w_1, \ldots, w_n)$ represented by T (Figure 3.6, left) is the same as the one represented by T':
$P(d_0|d_1, \ldots, d_n; w'_1, \ldots, w'_n)$ (Figure 3.6, right). When the conditional distributions (given by the tree edges) are Gaussian, an analytical solution exists for the new template weights $w'_1, \ldots, w'_n$, which ensures that the tree before and after the transformation model the same distribution (see section 3.2.2).
Starting from an arbitrary UPGMA tree, this basic transformation is applied iteratively $n-1$-times on sub-trees as shown in Figure 3.7. At each step, one inner node is removed and the procedure continues until all template leaves are directly connected to the query. Since the final weights are supposed to represent the influence of t_k on q, we adapt distance distributions from the corresponding template t_k as follows: $P(d|d_k, A_k)^{w_k^{\text{final}}}$.

Intuitively, weights less than 1 widen the distribution and vice versa.

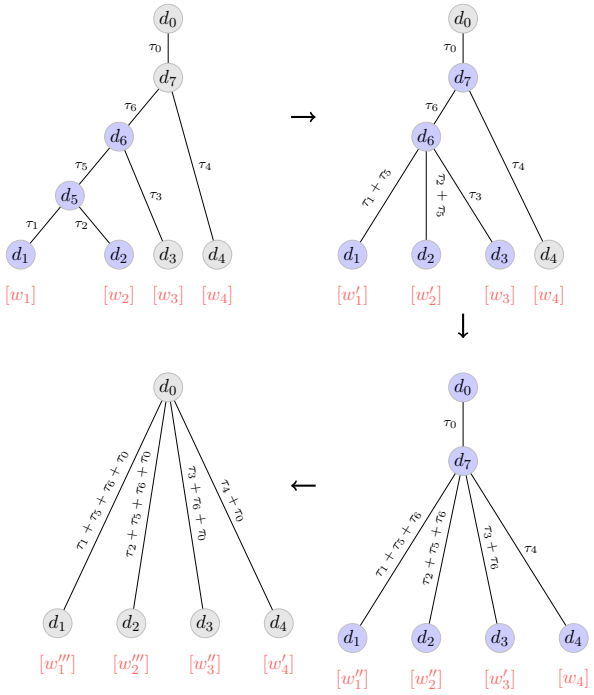


Figure 3.7.: Iterative restructuring of a tree. In each step, the basic transformation from Figure 3.6 is applied to an appropriate subtree (colored in blue). Weights and edge labels get updated until all templates are directly connected to the root.

Details of template weight calculation

In the following, we describe how to calculate template weights given a tree that specifies the evolutionary relations between a query and templates and among all templates. We are interested in the distance between a given pair residues in the query, d_0, given the corresponding template distances $d_1, \ldots, d_n$. We assign a weight w_i, $i = 1, \ldots, n$ ($w_0 = 1$ for the query) to each template. This weight will represent the influence of the template on the query. We model the distribution of d_0 given in Figure 3.6 on the left as follows:

$$\frac{P(d_0|d_1, \ldots, d_n, w_1, \ldots, w_n)}{P(d_0)} = \int \frac{P(d_0|d, w_0)}{P(d_0)} P(d|d_1, \ldots, d_n, w_1, \ldots, w_n) \mathrm{d}d, \quad (3.7)$$

where

$$\begin{aligned} P(d|d_1,\ldots,d_n,w_1,\ldots,w_n) &= \frac{P(d_1,\ldots,d_n|d,w_1,\ldots,w_n)P(d)}{P(d|d_1,\ldots,d_n)} \\ &= \frac{P(d_1|d,w_1)}{P(d_1)} \cdot \ldots \cdot \frac{P(d_n|d,w_n)}{P(d_n)} P(d). \end{aligned} \tag{3.8}$$

As mentioned in section 3.2.2, we assume a diffusive behaviour with variance proportional to time τ_i:

$$\frac{P(d_i|d,w_i)}{P(d_i)} \underset{d,d_i}{\propto} \exp\left(-\frac{w_i(d-d_i)^2}{\tau_i}\right) \quad \forall\, i = 0,\ldots,n. \tag{3.9}$$

The times τ_i are given by the UPGMA clustering. Then (3.7) becomes with respect to d_0:

$$\frac{P(d_0|d_1,\ldots,d_n,w_1,\ldots,w_n)}{P(d_0)} \propto \int \exp\left(-\sum_{i=0}^{n} \frac{w_i}{\tau_i}(d-d_i)^2\right) \mathrm{d}d. \tag{3.10}$$

The argument in the exponent can be rewritten as a quadratic expression of d:

$$-\underbrace{\left(\sum_{i=0}^{n}\frac{w_i}{\tau_i}\right)}_{\frac{1}{\tau_{\min}}} d^2 + 2\left(\sum_{i=0}^{n}\frac{w_i}{\tau_i}d_i\right)d - \sum_{i=0}^{n}\frac{w_i}{\tau_i}d_i^2 = -\frac{1}{\tau_{\min}}\left(d^2 - 2\left(\sum_{i=0}^{n}u_i d_i\right)d + \sum_{i=0}^{n}u_i d_i^2\right), \tag{3.11}$$

where we defined:

$$u_i := \frac{w_i/\tau_i}{\sum_{i'=0}^{n}\frac{w_{i'}}{\tau_{i'}}} = \frac{\tau_{\min}}{\tau_i}w_i. \tag{3.12}$$

Completing the square in (3.11) gives:

$$-\frac{1}{\tau_{\min}}\left(\left(d - \sum_{i=0}^{n}u_i d_i\right)^2 - \left(\sum_{i=0}^{n}u_i d_i\right)^2 + \sum_{i=0}^{n}u_i d_i^2\right). \tag{3.13}$$

When integrating over d (eq.(3.10)), the factor

$$\exp\left[-\frac{1}{\tau_{\min}}\left(\sum_{i=0}^{n}u_i d_i^2 - \left(\sum_{i=0}^{n}u_i d_i\right)^2\right)\right] \tag{3.14}$$

can be pulled out of the integral (since it is independent of d). Therefore:

$$\frac{P(d_0|d_1,\ldots,d_n,w_1,\ldots,w_n)}{P(d_0)} \underset{d_0}{\propto} \exp\left(-\frac{1}{\tau_{\min}}\left(\sum_{i=0}^{n} u_i d_i^2 - \left(\sum_{i=1}^{n} u_i d_i\right)^2\right)\right) \quad (3.15)$$

$$\underset{d_0}{\overset{!}{\propto}} \exp\left(-\sum_{i=1}^{n} \frac{w_i'}{\tau_0+\tau_i}(d_0-d_i)^2\right) \quad (3.16)$$

Here, the last step introduced new weights w_i' so that the template distances $d_1,\ldots,d_n$ become directly dependent on the query distance d_0. Now, an expression for w_i', $i = 1,\ldots,n$ must be found:

$$\sum_{i=0}^{n} u_i d_i^2 - \left(\sum_{i=0}^{n} u_i d_i\right)^2 = \tau_{\min} \sum_{i=1}^{n} \frac{w_i'}{\tau_0+\tau_i}(d_0-d_i)^2 + \text{const}(d_0). \quad (3.17)$$

We collect terms with equal powers of d_0:

$$(u_0-u_0^2)d_0^2 - \left(2u_0 \sum_{i=1}^{n} u_i d_i\right) d_0 = \left(\tau_{\min} \sum_{i=1}^{n} \frac{w_i'}{\tau_0+\tau_i}\right) d_0^2 - \left(2\tau_{\min} \sum_{i=1}^{n} \frac{w_i'}{\tau_0+\tau_i} d_i\right) d_0 + \text{const}(d_0). \quad (3.18)$$

Equating coefficients leads to:

$$\begin{aligned} u_0(1-u_0) &= \sum_{i=1}^{n} v_i \\ \sum_{i=1}^{n} u_i d_i &= \frac{1}{u_0} \sum_{i=1}^{n} v_i d_i, \end{aligned} \quad (3.19)$$

where $v_i := \frac{\tau_{\min}}{\tau_0+\tau_i} w_i'$. Solving for w_i' for all $i = 1,\ldots,n$ leads to:

$$w_i' = \frac{\frac{1}{\tau_0}+\frac{1}{\tau_i}}{\frac{1}{\tau_{\min}}} w_i w_0. \quad (3.20)$$

3.2.3. Template selection

Single template selection

By default, HHsearch returns a list of query-template alignments which is sorted with respect to P_{hom} (see HHsuite user guide, Söding et al. (2013)). However, further features may help to better characterize the best template for modeling. We trained

a neural network to predict the model quality measured by TMSCORE (Zhang and Skolnick, 2005b) based on three alignment features: score, sumProbs/L, ss-score. Given these predictions, the top ranked template is selected as in (Hildebrand et al., 2009).

Multiple templates selection: CASP9 strategy

The complex interdependencies between several templates have prevented a theoretically solid approach to solve the problem of finding the best template selection. We propose a heuristic selection strategy which combines coverage and local alignment quality. Moreover, we also take into account previously selected templates.
Starting with a set of accepted templates $\mathcal{T}$ (containing only the top ranked hit from section 3.2.3 at the beginning), we calculate for each template t in the hit-list $\mathcal{L}$ the score $S(t)$ (eq. 3.21), see Figure 3.8. We assess the local alignment quality at query position i by multiplying $P(i \diamond j)$ with P_{hom}, and calculate the score of t based on the alignment to the query q, $A(q,t)$ and all preselected templates in $\mathcal{T}$ as follows:

$$S(t) = \sum_{(i,j)\in A(q,t)} [\, \exp(\alpha(\underbrace{P_{\text{hom}}(t)\, P(i \diamond j)}_{\text{local quality of t at i}} - \underbrace{P_{\max}(i)}_{\text{coverage at i}}\,)) - \beta\,], \tag{3.21}$$

where $P_{\max}(i)$ is the maximum of $P_{\text{hom}}(t)\, P(i \diamond j)$ over all templates in $\mathcal{T}$:

$$P_{\max}(i) = \max_{t\in\mathcal{T}}\{\, P_{\text{hom}}(t) P(i \diamond j)\,\}. \tag{3.22}$$

The $P_{\max}(i)$ quantify the current coverage of residue i as given by all templates in $\mathcal{T}$. Thus, the exponent in 3.21 measures t's additional contribution at position i compared to all templates that are already in $\mathcal{T}$. α and β are tuning parameters which influence the greediness.

Among all templates with $S(t) > 0$, we select the one with the highest $S(t)$, update P_{max} and add it to set of selected templates, $\mathcal{T}$. This process is iterated until no template in $\mathcal{L}$ has a score $S(t) > 0$. By considering both alignment quality and coverage, we seek to ensure the approach works for both single- and multi-domain proteins.
Integrating structural information by filtering out templates with a pairwise TMSCORE below a given threshold could not improve the model quality.

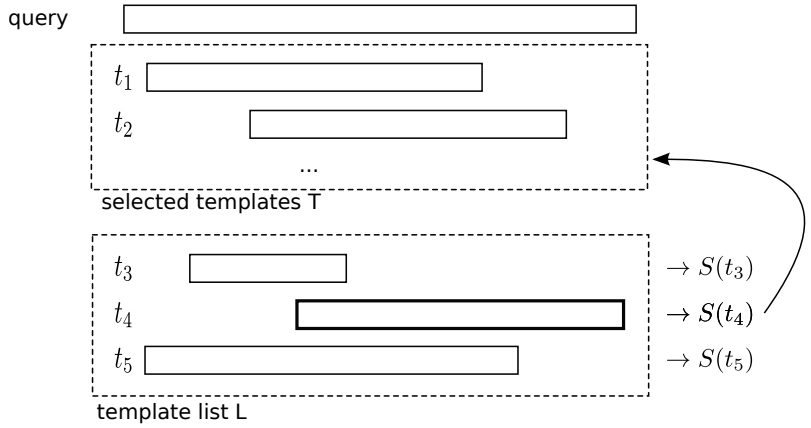


Figure 3.8.: Selection of multiple templates: the first template t_1 is selected by the NN in section 3.2.3. For each template in the template list $\mathcal{L}$ (lower dashed box) its score $S(t)$ is calculated (formula 3.21). Then the maximal scoring template (here: t_4) is accepted and added to $\mathcal{T}$ if its score is positive. This process is iterated until there is no more template with a score $S(t) > 0$.

3.3. Results

3.3.1. Benchmark sets

For benchmarking, we generated three different sets: a test-, training- and optimizing-set. To simulate conditions as given during CASP, we composed the sets such that they resembled CASP queries in difficulty as measured by their sequence identity to the best template. Analysing 108 CASP7 queries, we got the sequence identity bins in Figure 3.9, which served as a reference distribution in the three sets.

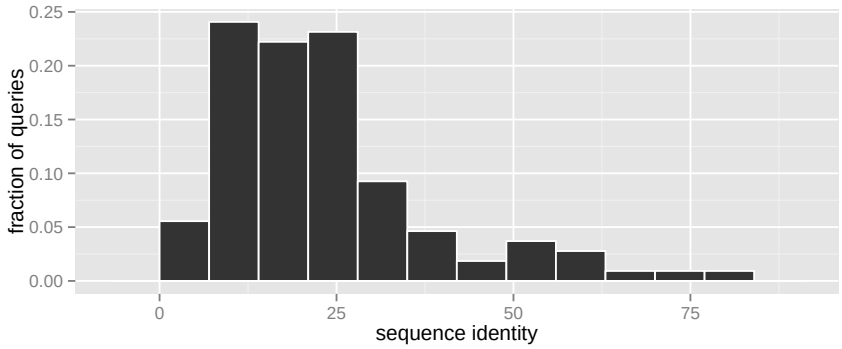


Figure 3.9.: Sequence identity of 108 CASP7 targets to their best template in the PDB.

We filtered the PDB (May 2010) down to 20% sequence identity and a minimal E-Value

of 0.1. For all sequences in the resulting PDB20E01, we built sequence profiles with our sensitive, iterative multiple sequence alignment tool HHBLITS (Remmert et al. (2012)) using standard parameters (but with three search iterations against the uniprot to get sufficiently diverse profiles as necessary for CASP to detect even remote homologs). Next, we trained Hidden Markov Models (HMMs) from these profiles, which served as input to HHSEARCH. Each query q was searched against the PDB70 resulting in a template list of database hits, tlist(q). A test-, training- and optimization-set were extracted based on the queries in PDB20E01 and their corresponding template lists as follows: for each set (test/training/optimization), we defined sequence identity bins according to the CASP sequence identity distribution (Figure 3.9). Each bin had a size equal to its fraction of queries times the set size. We randomly sampled a query q from PDB20E01 and determined its structurally most similar template t in tlist(q) according to TMALIGN. Next, we checked t's sequence identity $\mathrm{id}_{\mathrm{seq}}$ to the query. q was then randomly put into one of the three sets (test/training/optimization) if the following two conditions were met: first, it has not been sampled before and second, the $\mathrm{id}_{\mathrm{seq}}$ bin in the respective set was not yet filled up. In that way we generated a test- and training-set each of size 1000 and an optimization-set with 500 queries.

3.3.2. Template selection

Single template neural network

To select the first template based on multiple alignment features, we used the network in section 3.2.3. For training, we built up to ten 3D models for every query in the training set based the top ranked pairwise alignments in its tlist(q). This left us with 9.000 models, generated with MODELLER. We superposed models with their native PDB structures via TMSCORE. To learn the network parameters we ran a standard back-propagation procedure. In order to avoid local optima, training was started from several random initializations, which all turned out to be nearly equivalent in terms of their performance on the test-set. The correlation between the network predictions and true TMSCOREs was 0.89. In summary, this led to an increase of the mean TMSCORE compared with selecting the best ranked hit in the template list of 1.43% (TMSCORE: 0.684 versus 0.694, GDT-TS: 0.612 versus 0.621), Figure 3.10.

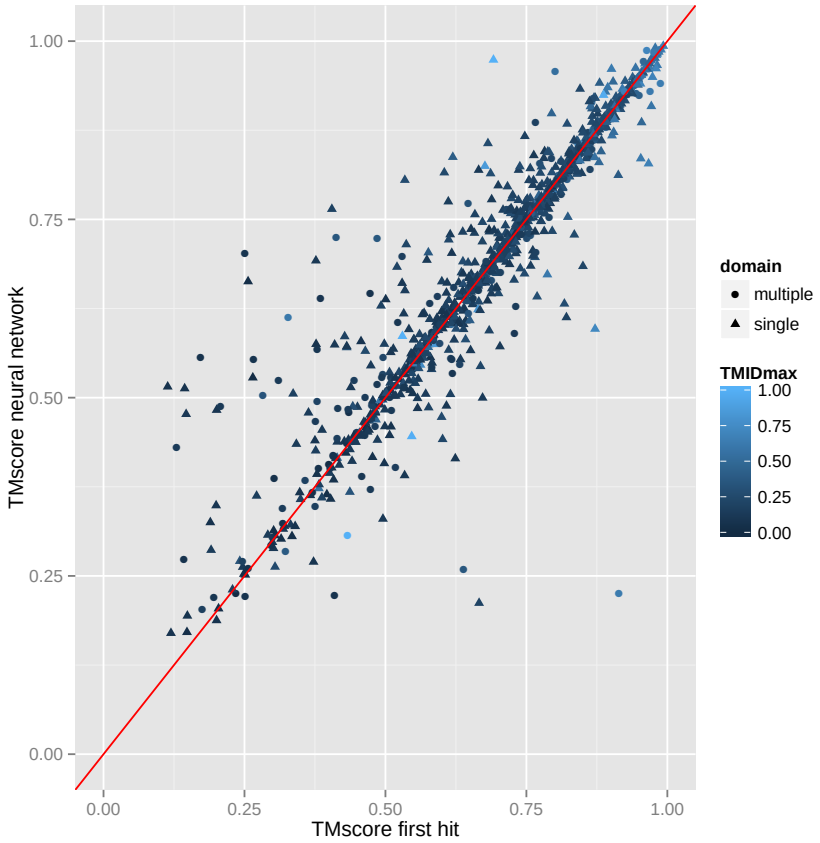
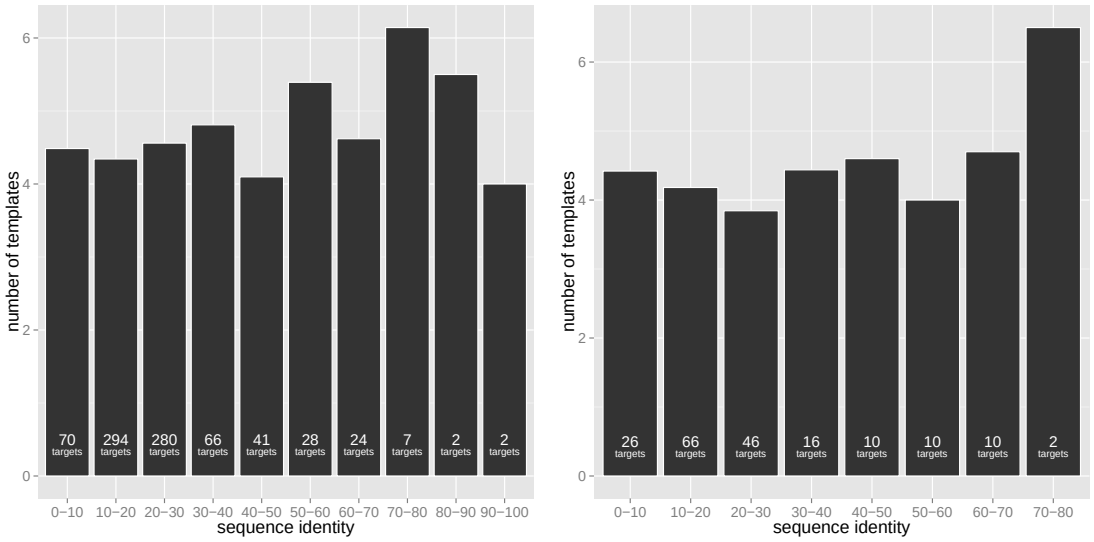


Figure 3.10.: Single template selection neural network: compared with using the best ranked hit in the hit-list, the single template NN increases the overall model quality (points above the diagonal). TMIDmax is the sequence identity of the query to the best template in its hit-list.

Multiple templates selection

Choosing multiple templates increases both the coverage and the probability to detect a correct template. However, a higher number of templates leads to accumulation of noise and wrong templates which decreases the model quality. As described in section 3.2.3, the CASP9 template selection heuristic has two tuning parameters: α and β. They were optimized on a grid $\alpha \in \{0.9, 0.95, 1, 1.05, 1.1\}, \beta \in \{0.8, 0.9, 1, 1.1, 1.2\}$ using all sequences in the optimization set as queries. For each parameter combination, templates were selected according to the score in formula (3.21). The alignments between the query and all templates were then supplied to MODELLER and $3D$ structures were generated. We found $\alpha = 0.95$ and $\beta = 1$ to maximize the cumulative TMSCORE of all models.

In the test-set, this strategy selected on average 4.6 templates per query, resulting in a mean coverage of 94% of all residues. Figure 3.11 depicts histograms which show the dependency of the number of templates on the sequence identity. In both single and multiple domain targets, the number of selected templates was quite comparable across all ranges.



(a) single domain (b) multiple domains

Figure 3.11.: Mean number of templates selected by the multi-template selection heuristic (y-axis) with a sequence identity as given on the x-axis. The number of targets in each sequence identity bin is denoted in white at its bottom.

3.3.3. Mixture density network

As training data for the mixture density networks for two-component distance restraints (section 3.2.1), we used the 3D models generated for the single template selection network in section 3.3.2. All network input feature values (i.e. similarity, posterior probabilities and distances) were extracted from the alignments and template structures. We fitted distributions of $\log(d) - \log(d_t)$ with mixtures of two Gaussians. MODELLER includes four different classes of distances depending on the atom types involved: between two C_α atoms (C_α–C_α), N-O atoms, side chain – main chain and side chain – side chain. So we generated four sets of training data with 3 million (C_α–C_α and N–O), 1 million (SC–MC) and $300k$ (SC–SC) training cases.

Optimizing the log-likelihood of the MDN was done by conjugate gradient ascent until

convergence was reached. Bad local minima were avoided by starting from different random initializations. Figure 3.1 depicts representative binned histograms of $\log(d) - \log(d_t)$ with overlaid MDN fits and demonstrates the feature dependency of the main and background component.

3.3.4. Distance restraints

In the following, we compare the effect of the new distance restraints on the model quality. To make this comparison independent of the input alignments, we selected and aligned templates with the heuristic in section 3.2.3 and used these fixed alignments to generate the different kinds of restraints to be tested in MODELLER.

Modeller's default distance restraints

We followed MODELLER's (version 9.10) default schedule and constructed distance restraints as in equation (3.3). Averaging the TMSCOREs of all models in the test set resulted in a mean values of 0.714. Thus, MODELLER profited from multi-template information and could increase the single template neural network score by 2.79%.

Two-component distance restraints

We removed all of MODELLER's default template based distance restraints and plugged in our new two component mixtures of Gaussians instead. The optimization schedule was kept unchanged. In sum, we got a slight improvement in model quality compared with MODELLER's default setting in section 3.3.4: the mean TMSCORE of all models was increased from 0.714 to 0.7195 (GDT-TS: 0.648 to 0.653).

Two-component distance restraints with template weights

By introducing template weights by the tree based method in section 3.2.2 the mean TMSCORE could be increased from 0.7195 to 0.725 (GDT-TS: 0.653 to 0.660). For calculating the weights, it is important to have an accurate estimation of pairwise distances between the query and all templates (and among the templates). Since the neural network for single template selection in section 3.2.3 has proven to provide solid estimates of the TMSCORE (correlation of 0.89), we again employed a very similar network which was trained on the same data (some additional features were included). Figure 3.12 shows a scatter plot to compare our new and weighted distance restraints with MODELLER's default ones.

Taken together, single template selection can be improved by neural network based TMSCORE predictions. Selecting more than one template increases model quality further. Replacing MODELLER's distance restraints by two component mixtures of Gaussian in combination with template specific weights provides a probabilistic approach to multi-template modeling that helps to build better models. For a summary of all results, see Table 3.1.

	single-template		multi-template		
	first hit	NN	MODELLER default (MD)	New restraints	New restraints + weights
TMSCORE	0.684	0.695	0.714	0.719	0.725
Improvement wrt 'first hit'	0%	1.43%	4.25%	5.06%	5.87%
P-value		1.67e-4 wrt first	<2.2e-16 wrt NN	7.58e-4 wrt MD	4.21e-12 wrt MD

Table 3.1.: Model quality (mean TMSCORE) for single- and multi-template homology modeling with MODELLER in different settings. In total, the mean TMSCORE can be improved by 5.87%. Using two component mixtures of Gaussians instead of MODELLER's default restraints yields 1.5%. P-values are calculated based on paired t-tests.

3.4. Discussion

MODELLER is one of the most popular tools for homology modeling (nearly 7.000 citations). Whereas there have been publications to improve MODELLER's energy function or loop modeling protocol (Fiser et al., 2000; Joo et al., 2009), to our knowledge, multi-template modeling by satisfaction of spatial restraints has not been advanced since its introduction. Modeling distance restraints with two-component mixtures of Gaussians is a statistically solid approach that fits well into MODELLER's probabilistic framework. However, the actual performance gain was quite modest on our benchmark. Since the modeling pipeline is quite complex and it is difficult to follow what is happening inside MODELLER's internals (e.g. the optimization schedule), we might have integrated the new restraints in a suboptimal way. Irrespective of that, the baseline seems to be quite high. A lot of effort has already been put in for years to advance homology modeling and all top CASP performers are scoring close to each other.

Obviously, distance restraints are highly dependent on the input alignments: as they get more reliable better models will emerge. Profile or HMM alignment tools such as HH-

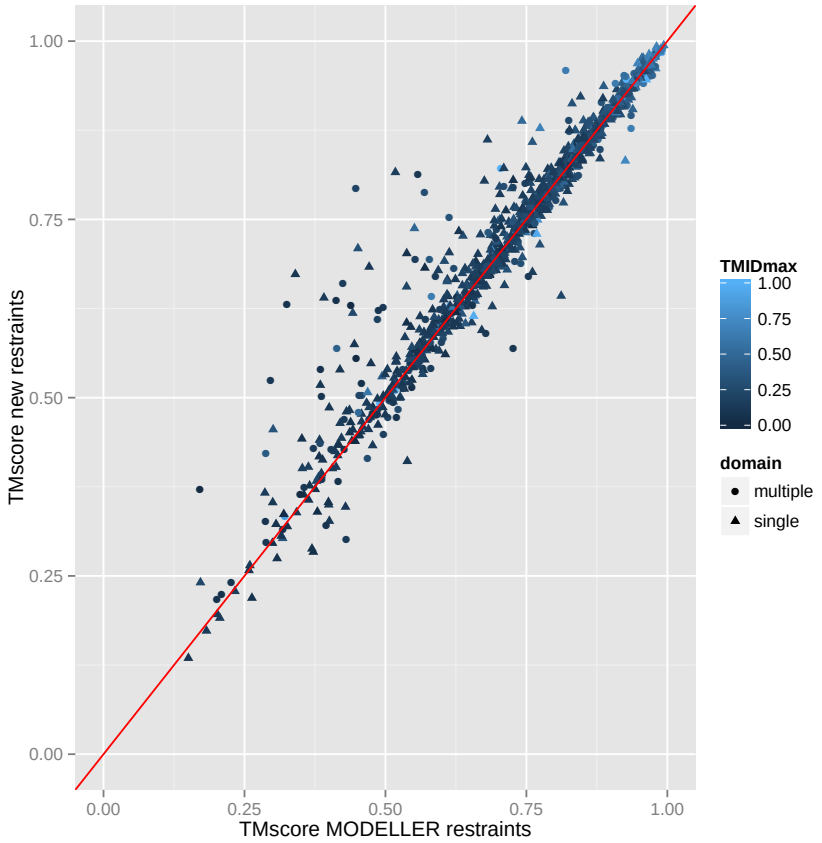


Figure 3.12.: TMscores of Modeller versus new restraint + template weight models: easy targets (measured in terms of sequence identity) mainly appear in the upper right corner. Most points are located around the diagonal with some advantage for models built based on the new distance restraints. TMIDmax is the sequence identity of the query to the best template in its hit-list.

search provide very sensitive and precise sequence alignments. Correctly identifying and aligning the structurally most similar template in a database and copying its coordinates onto the query is in most cases better than using multiple templates (Zhang, 2009). However, without structural information on the query side, it is currently impossible to produce such alignments. Multi-template modeling instead aims at building consensus restraints by extracting information from different homologous structures without having to know which single one is the best. Our new restraints were constructed such that contradictory information from several templates can be resolved in a coherent way. However, the selected templates might rarely cause severe contradictions that have to be resolved: either because all templates agree on the distance or because one obviously

best template outvotes the others. In order to test that hypothesis, we modified the template selection as follows. For each query in the test set, we constructed three different template selections which consisted of (a) two good templates which were supplemented by (b) one or (c) two additional bad templates as follows: the template lists for each query were sorted with respect to the predicted TMscore as calculated by the neural network in section 3.2.3. Then the top ranked template was selected. Next, we went down the sorted list and added the first hit which had a TMscore > 0.5 as the second good template (a). Conversely, for the bad templates, we went up the sorted list starting from the bottom and added one (b) and two templates (c) which had a TMscore < 0.3. Thus we ended up with three different template selections for each query q in test set. (b) and (c) were the ones including bad templates to test the ability of the restraints to correctly cope with harsh inconsistencies. Model quality measurements are shown in Table 3.2: In summary, the new restraints are especially advantageous when outliers

good templates	bad templates	old restraints	new restraints
2	0	0.713	0.723
2	1	0.705	0.721
2	2	0.695	0.716

Table 3.2.: Mean TMscores of 1000 models built with different templates: two good templates selected with the neural network and zero, one or two additional bad templates. The new restraints are less negatively affected by bad templates.

are present. In practice, however, the effect is considerably less pronounced due to the reasons mentioned above.

A future project might interweave alignment and model generation by sampling them in turn from appropriate conditional distributions and thus combining sequence and structure information in a more consistent way.

Part II.

Homology detection

4. Structural properties

4.1. Overview

When tracing the history of alignment tools, one of the most significant improvements came from the extension of a single sequence to a profile. Such a profile contains information not only about a single sequence but about the protein family this sequence belongs to, i.e. its evolutionary history. Thus, evolutionary related proteins are grouped together. During evolution, all residues in a protein are exposed to various selective pressures. It is obviously difficult to deduce these effects from a single sequence, whereas a profile encodes them in form of a pattern. For instance, Figure 4.1 shows both a sequence and its corresponding profile on top. The column marked by the dark blue arrow on the left encodes a well conserved alanine suggesting its functional importance. On the other side, the alanine below the light blue arrow on the right has often been replaced by other amino acids during evolution and is much less conserved.

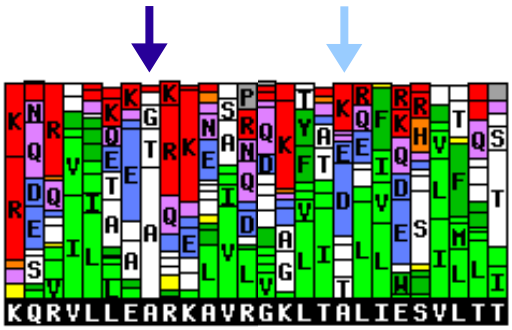


Figure 4.1.: A sequence with its profile on top. Two alanine residues are marked by arrows. The left one (dark blue) is well conserved whereas the right one has often been substituted by other amino acids. This extra information can not be inferred from the bare sequence.

Including more information also comes at a price, however: first, whereas the statistical significance of ungapped pairwise sequence-sequence alignments can be calculated analytically, this is no longer possible for profile alignments. Second, the computational burden of profile alignments is higher. Yet, both of these drawbacks can be neglected in most practical applications: Hidden Markov Models (HMMs), trained from MSAs, provide a theoretically solid framework and steady increase in computational power enables low running times. Due to the success of replacing single sequences with profiles, all current highly sensitive pairwise alignment tools heavily rely on them. In CASP, where it is of fundamental importance to have a sensitive and accurate alignment, all top performers employ profile based methods (Casp, 2012). Yet, there are different approaches to exploit profile information. Many different scores have been proposed to measure the similarity between two profile columns (see Ye et al. (2011) for an overview). In order to increase their performance, most tools extend their scoring function to include additional 1D structural properties (Söding and Remmert, 2011). Definitely the most prominent among these extra scores is secondary structure. It has been recognized to improve both alignment quality and precision.

4.1.1. General approaches to extend pairwise profile alignments

Even if the sequence profile is the most important input for any highly sensitive pairwise alignment algorithm, especially low homology detection profits from additional orthogonal information. For instance, a sequence profile can be sparse due to the lack of homologous sequences in a database or the inability of a multiple sequence alignment tool to detect remote (super-) family members. Apart from defining and refining existing profile-profile scores (Edgar and Sjölander, 2004) and experimenting with schemes to enrich the profile by pseudocounts (Biegert and Söding, 2009), some groups have started to integrate a bunch of additional terms into their scoring function. The SP, SP-4, SP-5 suite and SPARK-X come up with angle and fragment based potential functions (Faraggi et al., 2011; Song et al., 2007; Zhang et al., 2008). Xu and Xu (2000) designed a contact potential function and MUSTER (Zhang and Skolnick, 2005a) includes structure fragment profiles. Because of highly complex interactions between (pure) sequence scores and profile-derived structure scores, they must be reasonably combined. In the simplest case, the structural scores are orthogonal to both the profile score and each other, and can simply be added up. To allow, however, for more flexibility, Ma et al. (2012) made use of conditional neural fields. Ohlson et al. (2006) resort to neural networks to construct a nonlinear scoring function.

Like profile-profile scores, structural scores are local, meaning that different alignment positions are scored independently of each other. Pairwise interactions, however, capture non-local interactions but are computationally more intensive. In Ma et al. (2013), the authors integrate a global potential term and find a slight improvement in alignment quality.

4.1.2. Secondary structure

Biochemical analysis of naturally occurring proteins has revealed a set of recurring spatial patterns in their three-dimensional structures. Even though there is no unique definition of these patterns, the most popular one comes up with three distinct classes:

- alpha helix (H)
- beta sheet (E)
- random coil (C)

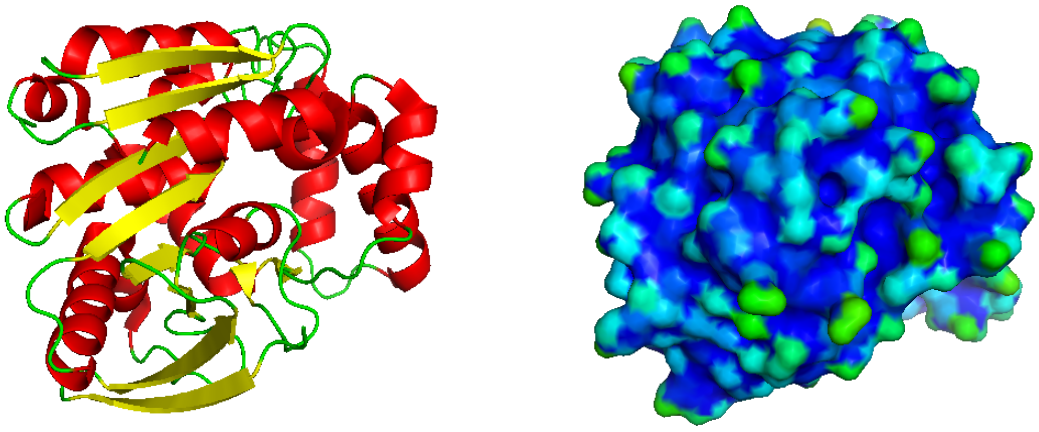


Figure 4.2.: X-ray structure of a hydrolase from Bacillus anthracis (pdb code 3OOS). Left: secondary structure coloring (helix (red), sheet (yellow), coiled (green)), right: solvent accessibility coloring (accessible (green), buried (blue)).

Figure 4.2 visualizes each of them. As the complexity of spatial structures might suggest, this three class partitioning is only a very crude representation of reality. Consequently, various more refined alphabets have been devised. Amongst them, the most prominent is the DSSP (Kabsch and Sander, 1983) which is an eight class extension of the classical

three states. Having a more detailed alphabet has pros and cons: they provide a more fine-grained description of secondary structure, but their prediction get more complicated.
Secondary structure is a spatial, function generating characteristic of a protein and thus it is generally more conserved than the amino acid sequence. Different amino acids have a natural propensity to come up in a given secondary structure element. For instance, methionine, alanine, leucine, glutamate and lysine (MALEK) prefer to adopt helical conformations. Even more distinct, each profile column favours another secondary structure state depending on the composition of amino acids at the current position. Beyond that, considering context information extends predictability of secondary structure because neighboring profile columns provide important information about the current position.

4.1.3. Further 1D structural properties

In the structure prediction community, secondary structure is considered as one of the most important feature apart from sequence (or profile). Nevertheless, a couple of other 1D structural properties has been defined and investigated. Among others, solvent accessibility, torsion angle potentials, contact density, structural environment and information from chemical shifts were scored (e.g. SP5, SPARK-X, RAPTOR, MUSTER). Furthermore, three states secondary structure alphabets have been refined to better describe backbone conformations and burial states (Karchin et al., 2004). All of these 1D properties aim at capturing a specific structural characteristic of a protein. However, actually all of them are calculated based on sequence (profile) information, mostly via machine learning techniques such as support vector machines and neural networks. Thus, these features are all derived from sequence profiles differing only in the structural property that they have been associated with during training. In most cases, (local) profile windows of a specific width D serve as input and a structural feature of interest is taken as output for some supervised learning algorithm. Neural networks, for instance, are flexible enough to establish a mapping between the input and output and can thereby provide a mapping from profiles to structures (if over-training has been taken care of).
When structural features strongly correlate with each other, as is the case for three and eight class secondary structure prediction, these interdependencies should be taken into account by the alignment scoring function. For this reason, to optimally integrate a new (structural) property into an alignment tool, the scoring function must be designed carefully.

4.2. Prediction Tools

As already indicated in the previous chapter, the performance of virtually every prediction tool is depending on the input sequence profile. As the relation between profiles and a specific 1D structural property is often quite complex, the corresponding mapping function will usually be complex as well. Neural networks have proven to be sufficiently versatile and powerful. In addition, conditional neural fields (Wang et al., 2011), support vector machines (Hua and Sun, 2001) and regression techniques have been implemented (Pan, 2001).
In the next sections we will shortly explain three approaches to profile based 1D structural property prediction. The first one, the well-known PsiPred, is a neural network for three class secondary structure prediction. In a next step, Predict-2nd extends PsiPred by learning various other structural features. Yet, all training data is required to be labeled, i.e. the structural 1D states have to be predefined (by human expert knowledge) based on solved structures. In order to overcome this limitation, we describe a more general approach by Ohlson et al. (2006) who stick to an unsupervised learning procedure called self organizing maps (SOMs) to derive an alignment score.

4.2.1. PSIPRED

PsiPred (Jones, 1999) is a well known secondary structure prediction tool and is one of the first programs to employ Psi-Blast-generated position specific scoring matrices as input. As outlined in sections 4.1.2, 4.1.3 and 4.2, it heavily relies on input from MSAs. In short, the workflow is as follows:

1. generation of a sequence profile by Psi-Blast

2. prediction of secondary structure with a neural network that takes the sequence profile as input

3. filtering of the predictions of step 2 with a second neural network

4. output of final secondary structure prediction together with position specific confidence

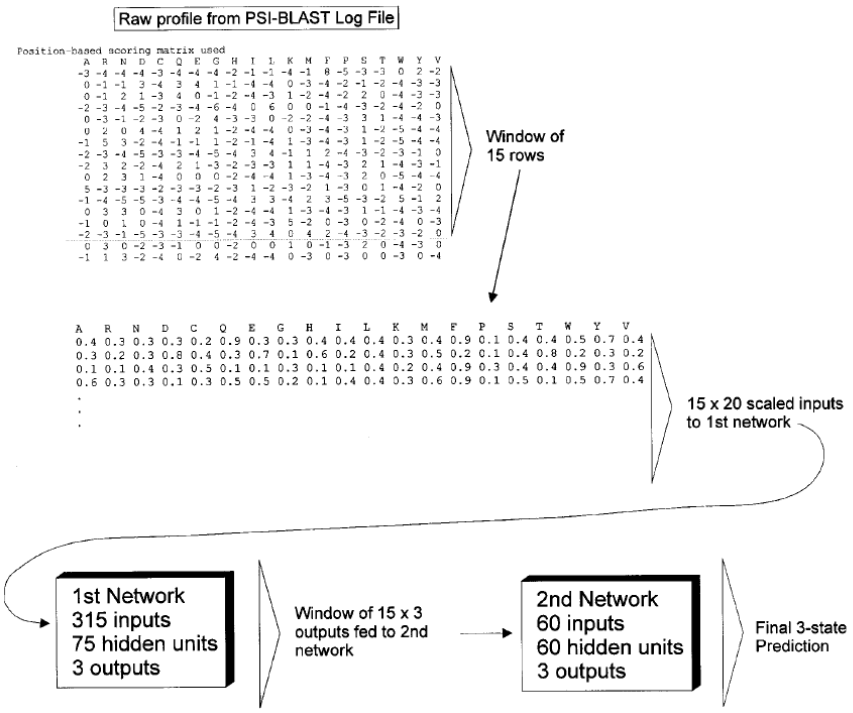


Figure 4.3.: PSIPRED flowchart: starting with a sequence profile, two feed-forward neural networks are operating consecutively and finally return a probability for each of the three secondary structure states: helix, sheet and random coil.

Figure 4.3 depicts the method. In order to optimally map local profile windows of width 15 onto their structural state, two successive feed-forward neural networks are operating consecutively. Here, the second network refines outputs from the first one. Altogether, about 27.000 parameters allow a non-linear adaptation of the network to the training data. Finally, each residue has a probability for helix, sheet and coil, respectively. Despite its methodological simplicity, PSIPRED still ranks among the top secondary structure prediction tools available. Nevertheless, it is obvious to investigate further structural properties by defining extended alphabets as described in the next section.

4.2.2. Predict-2nd

As mentioned in section 4.1.2, the classification of local structural elements into helix, sheet and random coil is by no means the only one possible. Refined so-called backbone and burial state alphabets have been defined (Karchin et al., 2004). Backbone alphabets focus on different structural features (DSSP and STRIDE) by subdividing a given state, e.g. beta sheets. DSSP is an eight state alphabet consisting of E (beta strand), H (alpha

helix), T (turn), S (bend), G (3-10 helix), B (short beta bridge), C (random coil) and I (pi helix). Furthermore, str, str2, str3 extend DSSP by manually splitting up the E state into six, seven and eight letters. They are motivated by the observation that parallel and anti-parallel beta sheets have distinct patterns of hydrophobic and polar residues. Burial alphabets attempt to express whether a residue is on the surface or in the interior of a protein. Karchin et al. (2004) have listed a whole series of them, most of which differ only in minor details.

Katzman et al. (2008) have developed a corresponding prediction tool called PREDICT-2ND. Similar to PSIPRED, it is a neural network which takes MSAs as input. In addition to PSIPRED, they extend their network in several respects: 1) instead of predicting three state secondary structure, they allow for a range of different alphabets of variable size, 2) the number of hidden layers is extended to three, 3) regarding the objective function they switched from an alphabet size dependent measure (Q_n, which is the percentage of correct predictions in the whole sequence) to a more size-independent number called 'bits-saved':

$$\text{bits-saved} = \frac{1}{n}\sum_{i=1}^{n}\log_2\frac{\hat{P}_i(c_i)}{P_0(c_i)},$$

where c_i, $i = 1, \ldots, n$, is the correct state at position i, P_0 is the background probability and $\hat{P}_i$ the predicted probability of c_i.

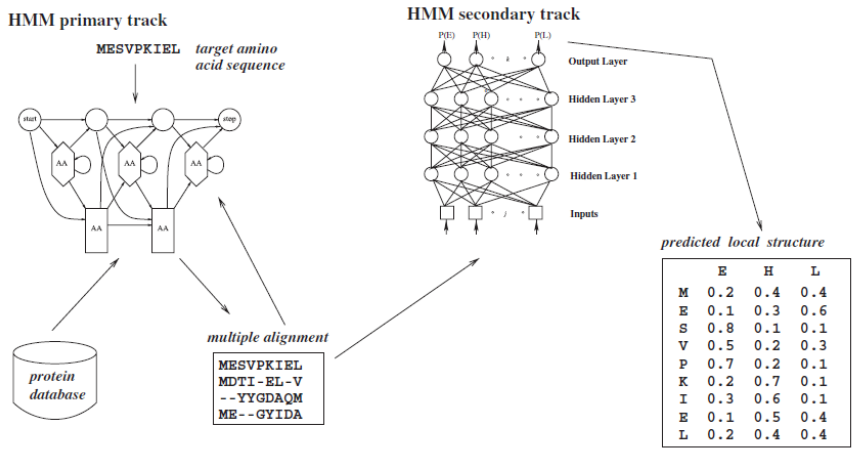


Figure 4.4.: PREDICT-2ND flowchart: starting with a multiple alignment, PREDICT-2ND calculates position specific probabilities for each letter in a given structural alphabet.

Like PsiPred, Predict-2nd takes only sequence profile information as input and returns for each residue the corresponding probabilities of each letter in the respective alphabet. Due to the network's complexity (three hidden layers and about 6.000 parameters), it is necessary to perform several training runs each starting from a different random initialization.
Even if various alphabets have been defined, they all rely on predefined structural states and are based on human expert knowledge. A more data driven approach that requires less prior information is presented in the following section.

4.2.3. SOM based method

Both PsiPred and Predict-2nd need predefined structural states. Therefore, training these networks requires the availability of labeled training data. But even if many different 1D structural properties have been defined, it is unclear whether all information in the local profiles windows is exploited. Ohlson et al. (2006) resorted to an unsupervised learning strategy to get rid of specific 1D structural properties. In particular, they construct a self organizing map (SOM) (Kohonen, 1982) and use it for scoring.
A SOM is a special type of neural network that is trained in an unsupervised way using a neighborhood function that preserves topological properties of the input space. Here, the input space consists of a (training) set of n training profile windows $\mathcal{T} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$, each having 20 rows, i.e. one for each amino acid, and D columns. As neighborhood function, the authors used the Euclidean distance. The SOM is then generated by Algorithm 1. In the end, the map consists of a grid, where each grid position p gets assigned a $20 \times D$-dimensional vector $\boldsymbol{Z}_p$. The values of all vectors will be learned by Algorithm 1 so that vectors next to each other in the grid are likely to share similar properties.
For prediction, any $20 \times D$ profile window $\boldsymbol{Y}$ can be mapped onto a SOM grid position $p_{\boldsymbol{Y}}$ by finding the SOM position p that minimizes the distance between $\boldsymbol{Y}$ and $\boldsymbol{Z}_p$:

$$p_{\boldsymbol{Y}} = \text{argmin}_p\{||\boldsymbol{Y}, \boldsymbol{Z}_p|| \; \forall \text{ grid positions } p \text{ in SOM}\}.$$

Provided with two SOM positions $p_{\boldsymbol{X}}$ and $p_{\boldsymbol{Y}}$ for a query and template residue, it is assumed that similar local profiles $\boldsymbol{X}$ and $\boldsymbol{Y}$ cause $p_{\boldsymbol{X}}$ and $p_{\boldsymbol{Y}}$ to be mapped close to each other in the grid and giving a high score. Ohlson et al. (2006) trained a neural network with $p_{\boldsymbol{Y}}$ and $p_{\boldsymbol{Y}}$ as input and the alignment score as output.
According to the authors, the inclusion of the SOM positions helped in some limited degree to better find remote homologs on fold level. Even though secondary structure

Algorithm 1: SOM profile window clustering

Data: Training profile windows $\mathcal{T} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$
Result: profile window SOM
Initialization: a grid of dimension $(5 \times 6 \times 7)$ is generated where at each grid position p a $20 \times D$-dimensional, randomly initialized vector $\boldsymbol{Z}_p$ is stored.;
for *10 epochs* **do**
  for *each training profile window* $\boldsymbol{X}_i$, $i = 1, \ldots, n$ **do**
    1. Find closest grid vector, $\boldsymbol{Z}_{closest}$, to point $\boldsymbol{X}_i$ according to Euclidean distance;
    2. $\boldsymbol{Z}_{closest} \leftarrow \boldsymbol{Z}_{closest} + \alpha \cdot (\boldsymbol{X}_i - v_{closest})$;
    3. update neighbors of $\boldsymbol{Z}_{closest}$ within radius r in the same way but by a smaller amount;
  reduce r and training rate α;

appears to be the most accessible and beneficial pattern encoded in sequence profiles, there is a small amount of extra information captured by their SOMs.
Despite of the generality of this approach, it has certain drawbacks: First, SOMs measure the similarity of profiles in terms of their Euclidean distance, which seems to be an unnatural measure for amino acid profiles. Second, a given profile is mapped onto only a single SOM position, corresponding to a hard assignment and ignoring noise in the data. Third, the training lacks any information about which profiles were aligned in the training set since only single profile windows are clustered. Fourth, the scoring of the SOM positions by the neural network lacks a background reference, i.e. these absolute SOM positions $p_{\boldsymbol{X}}$ and $p_{\boldsymbol{Y}}$ lack a possibility to discriminate signal from noise.

4.3. PPAS

In a comparative assessment of 20 sequence alignment tools, Yan et al. (2013) concluded that profile-profile tools considerably outperform methods relying on sequence-profile information only. For benchmarking our new method in section 5.1, we tested existing profile-profile alignment software. Even if many of them have been developed, most are useless in practice because of their very limited flexibility (e.g. fixed template databases, missing tools to generate input features).
PPAS (Yan et al., 2013) is part of the I-TASSER package that was developed in Yang

Zhang's lab. Its scoring function is defined as:

$$S_{\text{PPAS}}(i,j) = \sum_{a=1}^{20} F_q(i,a)L_t(a,j) + C \cdot \delta(ss_q(i), ss_t(j)) + \text{shift},$$

where the sum runs over all 20 amino acids, $F_q(i,a)$ is the query sequence frequency profile and $L_t(a,j)$ is the template log-odds profile, δ is the Kronecker function and C is an optimized weight. By default, both the query and template profiles are based on three iterations of PSI-BLAST against the 'non-redundant' nr database. Secondary structure on the query side is predicted by PSSPRED, an in-house tool. PPAS implements a position specific gap scheme, where no gaps are allowed inside secondary structure regions. An extension of PPAS incorporates profiles derived from a set of structural fragments that have similar spatial depth as the fragment at j-th position of the template. Yet, there is no tool available in the I-TASSER-package that allows constructing these profiles.
In (Yan et al., 2013), PPAS is benchmarked against HHSEARCH and found to have better (medium targets) or equal (hard targets) performance. Interestingly, the same benchmark shows that PPAS is ranked only slightly behind MUSTER, which is an extension of PPAS. It adds backbone-angle potential scores, depth profiles and solvent accessibility.

5. Homology detection with context states

5.1. Introduction

Top-performing methods for fold recognition and protein structure prediction are based on the pairwise alignment of query and template sequence profiles (Elofsson, 2002; Yan et al., 2013). They use the same dynamic programming algorithms as methods for pairwise sequence alignment, but they replace the substitution matrix score with a score that quantifies the similarity between sequence profile columns (e.g. the Euclidean distance or the scalar product). The power of these profile-profile alignment methods lies in leveraging the evolutionary information in the multiple sequence alignments (MSAs) that the sequence profiles were trained with.

Most top-performing structure prediction tools add to the profile column similarity score a secondary structure score which measures the similarity between the predicted secondary structure of the query protein and the known secondary structure of the template proteins. Such scores have been shown to improve the sensitivity for detecting remote homologs and the quality of the resulting alignments (Karplus et al., 2003; Xu and Xu, 2000). In order to maximize the information gain and therefore the improvements in alignment quality, various finer-grained alphabets of backbone structure states have been developed – together with tools to predict these states (Karchin et al., 2004, 2003; Katzman et al., 2008).

In addition to secondary structure, a number of other 1D structural properties are employed for improving sequence alignments in the twilight and midnight zone, such as solvent accessibility (Liu et al., 2007), residue coordination numbers (Karchin et al., 2004; Peng and Xu, 2010; Wu and Zhang, 2008), backbone dihedral torsion angles, 1D environmental fitness scores (Peng and Xu, 2009; Teichert et al., 2010). In all cases, the discretized 1D structural property of each position in the query is predicted from a local sequence profile window of 13 to 15 positions, and the similarity between predicted and actual 1D properties of the aligned query and template positions is scored in the alignment.

Most current pairwise profile-profile alignment tools integrate several of these 1D struc-

tural scores into their alignment algorithm. MUSTER (Wu and Zhang, 2008) is a threading program that extends its profile score by secondary structure, solvent accessibility, fragment depth profiles (Liu et al., 2007) and torsion angle potentials. Similarly, the Zhou lab has developed a series of alignment tools called SP^2, SP^3, SP^4, SP^5 and SPARKS-X (Yang et al., 2011) each of which expands the previous one by another 1D property. Interestingly, the latest implementation, SPARKS-X, got rid of the structure-derived sequence profiles that had been shown profitable in SP^3 and its successors. A possible reason for a decline in performance in such multi-component scoring functions is their suboptimal combination. In most implementations, all scores are considered to be independent and get added up using constant weights. Such an assumption obviously ignores the complex interdependencies between various individual scores. Ma et al. (2012) was the first to provide a theoretically solid framework to non-linearly combine any scores by employing conditional random fields (Lafferty et al., 2001) and a gradient tree boosting algorithm (Peng and Xu, 2009).

Whereas these 1D scores are limited to searching for templates with known structure, one can get independent of structural information by comparing 1D predictions with 1D predictions. Surprisingly, this works almost as well (Przybylski and Rost, 2004; Söding, 2005). We believe the reason is that these similarity scores reward the conservation of certain amino acid patterns in the local sequence context that the predictor associates with its conserved 1D property. In other words, the conservation of a 1D structural property leads to the conservation of local patterns of amino acid properties that is characteristic for this 1D property, and the conservation of these patterns is scored indirectly by comparing predicted 1D properties with each other.

Since the relationship of patterns to properties is "many to few", e.g. many quite different patterns are all characteristic for alpha helix states, more information might be extracted by learning and comparing conserved patterns directly, independent of what actual structural properties they are associated with. This is the approach we follow in this study. We cut out profile window pairs from structurally aligned pairwise training alignments and learn the set of the 32 best-conserved patterns in these homologous pairs using the Expectation Maximization algorithm. With these patterns which we call "context states", we define a score that helps to discriminate homologous from non-homologous positions by analyzing the conservation of patterns between the aligned positions. We show that the new context similarity score improves the alignment quality and fold recognition sensitivity of our pairwise alignment tools HHSEARCH and HHALIGN (Söding, 2005) and that this in turn results in better 3D homology models.

5.2. Methods

5.2.1. General approach and notation

We start from a large training set of N aligned profile window pairs of $D = 2d + 1 = 13$ columns. The alignments are obtained from structural alignments of the full-length protein domains from the SCOP database (see section 5.4.1). We seek to automatically identify the maximally conserved patterns ("context states") irrespective of any predefined structural or functional properties. The conserved patterns are represented by sequence profiles of length D.

We call the N aligned training profiles $\boldsymbol{X}_n = X_n(i-d,\cdot),\ldots,X_n(i+d,\cdot)$ and $\boldsymbol{Y}_n = Y_n(j-d,\cdot),\ldots,Y_n(j+d,\cdot)$, $n \in \{1,\ldots,N\}$. Here, $X_n(i,a)$ is the number of effective counts of amino acid a at position i in one training profile and position i is aligned to position j in the aligned profile. The effective counts are defined in the following way: Let $p_{\boldsymbol{X}_n}(i,a)$ be standard sequence profile built for sequence X, i.e., the probability of amino acid a occurring at position i in the MSA for X. The effective counts are defined as $X_n(i,a) = p_{\boldsymbol{X}_n}(i,a)\, N^{\text{eff}}_{\boldsymbol{X}_n}(i)$ and analogously $Y_n(j,a) := p_{\boldsymbol{Y}_n}(j,a)\, N^{\text{eff}}_{\boldsymbol{Y}_n}(j)$. Here, $N^{\text{eff}}_{\boldsymbol{X}_n}(i)$ (abbreviated "Neff" in the following) is the number of effective sequences at position i as defined in the section 1.4.1.

Each of the K conserved patterns (= context states) is parameterized by a sequence profile $\boldsymbol{p}_k$. $\boldsymbol{p}_k$ is a $D \times 20$ matrix with $p_k(j,a)$ being the occurrence probability of amino acid $a \in \{1,\ldots,20\}$ at profile column $j \in \{-d,\ldots,d\}$. Each context state has a mixture weight α_k. We abbreviate the model parameters by $\boldsymbol{\theta}_k = (\boldsymbol{p}_k, \alpha_k)$ and $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_K)$.

5.2.2. Generative model

We want to find parameters $\boldsymbol{\Theta}$ that maximize the likelihood function

$$L(\boldsymbol{\Theta}) = P\left((\boldsymbol{X}_1,\boldsymbol{Y}_1),\ldots,(\boldsymbol{X}_N,\boldsymbol{Y}_N)|\boldsymbol{\Theta}\right) = \prod_{n=1}^{N} P(\boldsymbol{X}_n,\boldsymbol{Y}_n|\boldsymbol{\Theta}). \tag{5.1}$$

All training samples are supposed to be independent of each other so that the likelihood can be decomposed into a product. We use a mixture model for $P(\boldsymbol{X}_n,\boldsymbol{Y}_n|\boldsymbol{\Theta})$ as shown in Fig. 5.1. The hidden variable $z_n \in \{1,\ldots,K\}$ indicates the index of the context state that gave rise to $(\boldsymbol{X}_n,\boldsymbol{Y}_n)$:

$$\prod_{n=1}^{N} P(\boldsymbol{X}_n,\boldsymbol{Y}_n|\boldsymbol{\Theta}) = \prod_{n=1}^{N}\sum_{k=1}^{K} P(\boldsymbol{X}_n,\boldsymbol{Y}_n,z_n=k|\boldsymbol{\theta}_k). \tag{5.2}$$

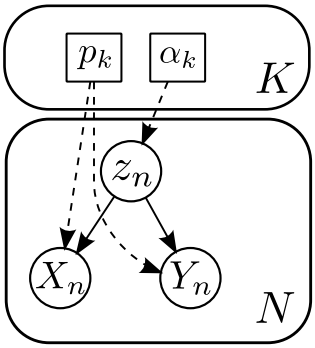


Figure 5.1.: Generative graphical model: Each of the N training profile pairs ($\boldsymbol{X}_n$, $\boldsymbol{Y}_n$) is generated by a mixture distribution with K components, the "context states". The hidden variables z_n encode the context state that gave rise to the n'th training sample ($\boldsymbol{X}_n$, $\boldsymbol{Y}_n$). Each of the D columns in these count profiles is modeled by a multinomial distribution over the 20 amino acids with parameters $\boldsymbol{p}_k$. The context states have mixture weights $\alpha_k (k = 1, \ldots, K)$.

Since our model assumes conditional independence of $\boldsymbol{X}_n$ and $\boldsymbol{Y}_n$ given the hidden context state z_n, it follows that

$$\prod_{n=1}^{N} P(\boldsymbol{X}_n, \boldsymbol{Y}_n | \boldsymbol{\Theta}) = \prod_{n=1}^{N} \sum_{k=1}^{K} P(\boldsymbol{X}_n | \boldsymbol{p}_k) P(\boldsymbol{Y}_n | \boldsymbol{p}_k) P(z_n = k | \alpha_k). \tag{5.3}$$

We model $P(\boldsymbol{X}_n|\boldsymbol{p}_k)$, the probability to observe counts $X_n(j,a)$ of amino acid $a = 1, \ldots, 20$ in column $j = -d, \ldots, d$, using a multinomial distribution for each column j,

$$P(\boldsymbol{X}_n | \boldsymbol{p}_k) = \prod_{j=-d}^{d} \left(\frac{\Gamma(N^{\text{eff}}_{\boldsymbol{X}_n}(j) + 1)}{\prod_{a=1}^{20} \Gamma(X_n(j,a) + 1)} \prod_{a=1}^{20} p_k(j,a)^{X_n(j,a)} \right)^{w_j}, \tag{5.4}$$

and analogously for $P(\boldsymbol{Y}_n|\boldsymbol{p}_k)$. Since the effective counts $X_n(j,a)$ can assume values outside the natural numbers, we replaced factorials $x!$ with Gamma functions $\Gamma(x+1)$. The $\boldsymbol{p}_k$ are discrete probability distributions and need to satisfy

$$\sum_{a=1}^{20} p_k(j,a) = 1, \text{ for } k = 1, \ldots, K \text{ and } j = -d, \ldots, d. \tag{5.5}$$

We assign a weight w_j to each column in eq. (5.9). The weights are parameterized as $w_j = w_{\text{center}}\, \beta^{|j|}$, so that central columns contribute more than flanking columns when

$\beta < 1$. The context state prior probabilities $p(z_n|\alpha_k)$ are simply the mixture weights,

$$P(z_n = k|\alpha_k) = \alpha_k, \text{ with } \sum_{k=1}^{K} \alpha_k = 1. \tag{5.6}$$

5.2.3. EM algorithm

To find the K context states, we maximize the likelihood of generating the pairs of aligned training profiles, given in eq (5.1). This can be achieved efficiently with the Expectation Maximization (EM) algorithm (Dempster et al., 1977). To account for the constraints in (5.5) and (5.6), we use the method of Lagrange multipliers to analytically perform the optimization in the M-step.
We find the $K = 32$ context states by maximizing the probability of the training data. More precisely, the optimization problem to solve is:

$$\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}}\, P(\mathcal{T}|\boldsymbol{\Theta}). \tag{5.7}$$

Here, $P(\mathcal{T}|\boldsymbol{\Theta})$ was factorized as follows (eq):

$$\prod_{n=1}^{N} \sum_{k=1}^{K} P(\boldsymbol{X}_n|\boldsymbol{p}_k) P(\boldsymbol{Y}_n|\boldsymbol{p}_k) P(z_n = k|\alpha_k), \tag{5.8}$$

where $\boldsymbol{z} = (z_1, \ldots, z_N)$ were hidden variables with $z_n \in \{1, \ldots, K\}$.
Furthermore we modeled $P(\boldsymbol{X}_n|\boldsymbol{p}_k)$ and $P(\boldsymbol{Y}_n|\boldsymbol{p}_k)$ using multinomial distributions:

$$P(\boldsymbol{X}_n|\boldsymbol{p}_k) = \prod_{j=-d}^{d} \left(\frac{\Gamma(N^{\text{eff}}_{\boldsymbol{X}_n}(j) + 1)}{\prod_{a=1}^{20} \Gamma(X_n(j,a) + 1)} \prod_{a=1}^{20} p_k(j,a)^{X_n(j,a)} \right)^{w_j} \tag{5.9}$$

and we had to satisfy the following constraints:

$$\sum_{a=1}^{20} p_k(j,a) = 1, \forall k = 1, \ldots K,\ j = -d, \ldots, d. \tag{5.10}$$

and

$$P(z_n = k|\alpha_k) = \alpha_k, \sum_{k=1}^{K} \alpha_k = 1. \tag{5.11}$$

5.3. EM algorithm

The EM-algorithm can be used to maximize the log-likelihood:

$$\log P(\mathcal{T}|\boldsymbol{\Theta}) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} P(\boldsymbol{X}_n, \boldsymbol{Y}_n, z_n = k|\boldsymbol{\theta}_k). \tag{5.12}$$

As there is a sum inside the logarithm, this expression can not be maximized analytically. Instead, the EM algorithm iterates the following three steps:

First, we define the (complete data) log-likelihood:

$$\mathcal{L}_c(\boldsymbol{\Theta}) := \sum_{n=1}^{N} \log P(\boldsymbol{X}_n, \boldsymbol{Y}_n, z_n|\boldsymbol{\Theta}). \tag{5.13}$$

Since the z_n are hidden, $\mathcal{L}_c(\boldsymbol{\Theta})$ is estimated as the following expectation under the posterior distribution of the z_n (E-step):

$$\mathbb{Q}(\boldsymbol{\Theta}|\tilde{\boldsymbol{\Theta}}) = \mathbb{E}_{\boldsymbol{z}|\mathcal{T},\tilde{\boldsymbol{\Theta}}}[\mathcal{L}_c(\boldsymbol{\Theta})] = \mathbb{E}_{\boldsymbol{z}|\mathcal{T},\tilde{\boldsymbol{\Theta}}}\left[\sum_{n=1}^{N} \log P(\boldsymbol{X}_n, \boldsymbol{Y}_n, z_n|\boldsymbol{\Theta})\right],\ z_n \in \{1, \ldots, K\}. \tag{5.14}$$

Here, $\tilde{\boldsymbol{\Theta}}$ are given estimates of $\boldsymbol{\Theta}$ from the previous iteration. They are needed to calculate the posteriors of the z_n.

Second, $\mathbb{Q}(\boldsymbol{\Theta}|\tilde{\boldsymbol{\Theta}})$ is maximized over $\boldsymbol{\Theta}$ (M-step):

$$\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}}\ \mathbb{Q}(\boldsymbol{\Theta}|\tilde{\boldsymbol{\Theta}}). \tag{5.15}$$

Since the logarithm now acts directly on $P(\boldsymbol{X}_n, \boldsymbol{Y}_n, z_n|\boldsymbol{\Theta})$ in $Q(\boldsymbol{\Theta}|\tilde{\boldsymbol{\Theta}})$, the M-step will have a closed form solution. Subsequently, $\tilde{\boldsymbol{\Theta}}$ is updated:

$$\tilde{\boldsymbol{\Theta}} \leftarrow \boldsymbol{\Theta}^*. \tag{5.16}$$

Third, the E-step and M-step are iterated until convergence or a given maximum number of times.

Let us now specify the E-step further. Exploiting the linearity of the expectation oper-

ator, eq (5.14) becomes:

$$\begin{aligned}\mathbb{Q}(\boldsymbol{\Theta}|\tilde{\boldsymbol{\Theta}}) &= \sum_{n=1}^{N} \mathbb{E}_{\boldsymbol{z}|\mathcal{T},\tilde{\boldsymbol{\Theta}}} \left[\log P(\boldsymbol{X}_n, \boldsymbol{Y}_n, z_n|\boldsymbol{\Theta})\right] \\ &= \sum_{n=1}^{N} \sum_{z_n=1}^{K} P(z_n|\boldsymbol{X}_n, \boldsymbol{Y}_n, \tilde{\boldsymbol{\Theta}}) \log P(\boldsymbol{X}_n, \boldsymbol{Y}_n, z_n|\boldsymbol{\Theta}).\end{aligned} \tag{5.17}$$

Due to the fact that the posteriors of the z_n are conditioned on $\tilde{\boldsymbol{\Theta}}$, which is known from the previous iteration, it can easily be calculated with Bayes' Theorem:

$$P(z_n|\boldsymbol{X}_n, \boldsymbol{Y}_n, \tilde{\boldsymbol{\Theta}}) = \frac{P(\boldsymbol{X}_n, \boldsymbol{Y}_n|z_n, \tilde{\boldsymbol{\Theta}})P(z_n|\tilde{\boldsymbol{\Theta}})}{\sum_{z_n} P(\boldsymbol{X}_n, \boldsymbol{Y}_n|z_n, \tilde{\boldsymbol{\Theta}})P(z_n|\tilde{\boldsymbol{\Theta}})}. \tag{5.18}$$

Here, the first term in the numerator decomposes to:

$$P(\boldsymbol{X}_n, \boldsymbol{Y}_n|z_n, \tilde{\boldsymbol{\Theta}}) = P(\boldsymbol{X}_n|z_n, \tilde{\boldsymbol{\Theta}})P(\boldsymbol{Y}_n|z_n, \tilde{\boldsymbol{\Theta}}), \tag{5.19}$$

whereas the second term in the denominator in (5.18) is:

$$P(z_n = k|\tilde{\boldsymbol{\Theta}}) = \tilde{\alpha}_k. \tag{5.20}$$

Taken together, the posterior probability in (5.18) results in:

$$\begin{aligned}P(z_n = k|\boldsymbol{X}_n, \boldsymbol{Y}_n, \tilde{\boldsymbol{\Theta}}) \propto \tilde{\alpha}_k &\prod_{j=-d}^{d} \left(\frac{\Gamma(N^{\text{eff}}_{\boldsymbol{X}_n}(j) + 1)}{\prod_{a=1}^{20} \Gamma(X_n(j,a) + 1)} \prod_{a=1}^{20} \tilde{p}_k(j,a)^{X_n(j,a)} \right)^{w_j} \times \\ &\prod_{j=-d}^{d} \left(\frac{\Gamma(N^{\text{eff}}_{\boldsymbol{Y}_n}(j) + 1)}{\prod_{a=1}^{20} \Gamma(X_n(j,a) + 1)} \prod_{a=1}^{20} \tilde{p}_k(j,a)^{Y_n(j,a)} \right)^{w_j} \\ \propto \tilde{\alpha}_k &\prod_{j=-d}^{d} \left(\prod_{a=1}^{20} \tilde{p}_k(j,a)^{X_n(j,a)+Y_n(j,a)} \right)^{w_j}.\end{aligned} \tag{5.21}$$

Starting from here, we can work out the expectation $\mathbb{Q}(\boldsymbol{\Theta}|\tilde{\boldsymbol{\Theta}})$:

$$\begin{aligned}\mathbb{Q}(\boldsymbol{\Theta}|\tilde{\boldsymbol{\Theta}}) &= \sum_{n=1}^{N} \mathbb{E}_{\boldsymbol{z}|\mathcal{T},\tilde{\boldsymbol{\Theta}}} \left[\log P(\boldsymbol{X}_n, \boldsymbol{Y}_n, z_n|\boldsymbol{\Theta})\right] \\ &= \sum_{n=1}^{N} \sum_{j=-d}^{d} w_j \Bigg(\log\Gamma(N^{\text{eff}}_{\boldsymbol{X}_n}(j)+1) + \log\Gamma(N^{\text{eff}}_{\boldsymbol{Y}_n}(j)+1) - \sum_{a=1}^{20} \log\Gamma(X_n(j,a)+1) \\ &\quad - \sum_{a=1}^{20} \log\Gamma(Y_n(j,a)+1) + \sum_{a=1}^{20} X_n(j,a)\mathbb{E}_{\boldsymbol{z}|\mathcal{T},\tilde{\boldsymbol{\Theta}}}[\log p_{z_n}(j,a)] \\ &\quad + \sum_{a=1}^{20} Y_n(j,a)\mathbb{E}_{\boldsymbol{z}|\mathcal{T},\tilde{\boldsymbol{\Theta}}}[\log p_{z_n}(j,a)] \Bigg) + \sum_{n=1}^{N} \mathbb{E}_{\boldsymbol{z}|\mathcal{T},\tilde{\boldsymbol{\Theta}}}[\log \alpha_{z_n}].\end{aligned} \tag{5.22}$$

Writing out the expectations yields:

$$\mathbb{E}_{\boldsymbol{z}|\mathcal{T},\tilde{\boldsymbol{\Theta}}}[\log p_{z_n}(j,a)] = \sum_{k=1}^{K} P(z_n = k|\boldsymbol{X}_n, \boldsymbol{Y}_n, \tilde{\boldsymbol{\Theta}})\ \log p_k(j,a) \tag{5.23}$$

and

$$\mathbb{E}_{\boldsymbol{z}|\mathcal{T},\tilde{\boldsymbol{\Theta}}}[\log \alpha_{z_n}] = \sum_{k=1}^{K} P(z_n = k|\boldsymbol{X}_n, \boldsymbol{Y}_n, \tilde{\boldsymbol{\Theta}})\ \log \alpha_k. \tag{5.24}$$

Since α_k and $p_k(j,a)$ describe distributions, they are constrained to satisfy:

$$\sum_{k=1}^{K} \alpha_k = 1 \ \text{ and } \ \sum_{a=1}^{20} p_k(j,a) = 1\ \forall k, j. \tag{5.25}$$

Having set up the E-step, we now turn to the M-step, i.e. the maximization of $\mathbb{Q}(\boldsymbol{\Theta}|\tilde{\boldsymbol{\Theta}})$ with respect to the constraints in (5.25). To find an analytical solution, we introduce Lagrange multipliers $\lambda, \mu_{kj} \in \mathbb{R}$, one for each constraint. According to the Lagrange formalism, one has:

$$\frac{\partial \mathbb{Q}}{\partial \alpha_k} = \sum_{n=1}^{N} \frac{P(z_n|\boldsymbol{X}_n, \boldsymbol{Y}_n, \tilde{\boldsymbol{\Theta}})}{\alpha_k} - \lambda \frac{\partial}{\partial \alpha_k} \left(\sum_{k=1}^{K} \alpha_k - 1 \right) = 0 \tag{5.26}$$

and

$$\begin{aligned}\frac{\partial \mathbb{Q}}{\partial p_k(j,a)} =& \sum_{n=1}^{N} \frac{w_j \left(X_n(j,a) + Y_n(j,a)\right) P(z_n = k | \boldsymbol{X}_n, \boldsymbol{Y}_n, \tilde{\boldsymbol{\Theta}})}{p_k(j,a)} \\ & - \mu_{kj} \cdot \frac{\partial}{\partial p_k(j,a)} \left(\sum_{a=1}^{20} p_k(j,a) - 1 \right) = 0.\end{aligned} \tag{5.27}$$

Finally, composing (5.26), (5.27) and (5.25), one ends up in the following M-step formulas:

$$\begin{aligned}\alpha_k =& \frac{\sum_{n=1}^{N} P(z_n = k | \boldsymbol{X}_n, \boldsymbol{Y}_n, \tilde{\boldsymbol{\Theta}})}{\sum_{k'=1}^{K} \sum_{n=1}^{N} P(z_n = k' | \boldsymbol{X}_n, \boldsymbol{Y}_n, \tilde{\boldsymbol{\Theta}})} \\ p_k(j,a) =& \frac{\sum_{n=1}^{N} P(z_n = k | \boldsymbol{X}_n, \boldsymbol{Y}_n, \tilde{\boldsymbol{\Theta}}) (X_n(j,a) + Y_n(j,a))}{\sum_{b=1}^{20} \sum_{n=1}^{N} P(z_n = k | \boldsymbol{X}_n, \boldsymbol{Y}_n, \tilde{\boldsymbol{\Theta}}) (X_n(j,b) + Y_n(j,b))}.\end{aligned} \tag{5.28}$$

A closer look at the update equations in (5.28) reveals that the state priors α_k are basically the normalized sums of state posteriors. On the other hand, $p_k(j,a)$, the multinomial parameters describing the profile distributions can be seen as weighted (and normalized) sums of the state posteriors, where the weights are given by the amino acid counts in the training profile window pairs. I.e. assuming, there is a bunch of training profile pairs in the dataset $\mathcal{T}$, which has a conserved cysteine at their central position $j = 0$ and there is one context state k which favours such profiles (meaning it has a high posterior), then $p_k(0, \text{cysteine})$ will become even more pronounced resulting in a state with a dominant cysteine column at its center.
Furthermore, since the z_n are described probabilistically by their posterior distributions, this approach is also referred to as *soft clustering*.

5.3.1. Scoring functions

Context states score

We define the context score for position i in profile $\boldsymbol{X}$ and position j in profile $\boldsymbol{Y}$ as a log-odds score,

$$S_{\text{ctx}}(\boldsymbol{X}_i, \boldsymbol{Y}_j) = \log \left(\frac{P(\boldsymbol{X}_i, \boldsymbol{Y}_j | \boldsymbol{\Theta})}{P(\boldsymbol{X}_i | \boldsymbol{\Theta})\, P(\boldsymbol{Y}_j | \boldsymbol{\Theta})} \right), \tag{5.29}$$

i.e., the logarithm of the ratio of probability

$$P(\boldsymbol{X}_i, \boldsymbol{Y}_j|\boldsymbol{\Theta}) = \sum_{k=1}^{K} P(\boldsymbol{X}_i|z_n{=}k, \boldsymbol{\theta}_k)\, P(\boldsymbol{Y}_i|z_n{=}k, \boldsymbol{\theta}_k)\, \alpha_k \tag{5.30}$$

for $\boldsymbol{X}_i$ and $\boldsymbol{Y}_j$ to have been generated together from the same context state, divided by the probability $P(\boldsymbol{X}_i|\boldsymbol{\Theta})P(\boldsymbol{Y}_j|\boldsymbol{\Theta})$ for $\boldsymbol{X}_i$ and $\boldsymbol{Y}_j$ to have been generated independently of each other. By applying Bayes' Theorem twice,

$$P(\boldsymbol{X}_i|z_n, \boldsymbol{\Theta}) = \frac{P(z_n|\boldsymbol{X}_i, \boldsymbol{\Theta})P(\boldsymbol{X}_i|\boldsymbol{\Theta})}{P(z_n|\boldsymbol{\Theta})}\,, \tag{5.31}$$

this expression can be transformed into the following form,

$$S_{\text{ctx}}(\boldsymbol{X}_i, \boldsymbol{Y}_j) = \log \sum_{z_n} \frac{P(z_n|\boldsymbol{X}_i, \boldsymbol{\Theta})P(z_n|\boldsymbol{Y}_j, \boldsymbol{\Theta})}{P(z_n|\boldsymbol{\Theta})}. \tag{5.32}$$

Note the analogy to the log-sum-of-odds scoring function for profile-profile alignment that was derived in (Söding, 2005),

$$S_{\text{aa}}(p_X(i), p_Y(j)) = \log \sum_{a=1}^{20} \frac{p_X(i,a)\, p_Y(j,a)}{f(a)}\,. \tag{5.33}$$

Here, the amino acid a is analogous to our context state k. The nominators describe the probability for the two profile columns i and j to co-emit amino acid a, or for the two profile contexts to emit context state $z_n = k$, respectively. $f(a)$, the background frequency of amino acid a, is analogous to α_k. Multiplying by $1/f(a)$ (or $1/\alpha_k$) corrects for the fact that frequent amino acids match by chance more frequently than rare ones. In our context similarity score, however, we compare sequence contexts by comparing count profiles of D columns instead of single profile columns.
Finally, the total score is a linear combination of profile column score, context states score, and HHsearch's standard 3-state secondary structure score based on PSIPRED predictions Söding (2005):

$$\begin{aligned} S_{\text{total}}(i,j) = (1 - w_{\text{ctx}})\, S_{\text{aa}}(p_X(i), p_Y(j)) + w_{\text{ctx}}\, S_{\text{ctx}}(\boldsymbol{X}_i, \boldsymbol{Y}_j) \\ + w_{\text{ss}}\, S_{\text{ss}}(i,j). \end{aligned} \tag{5.34}$$

Since the context states score is based on a window of size $D = 13$, the weight of the score should be adapted for two reasons:

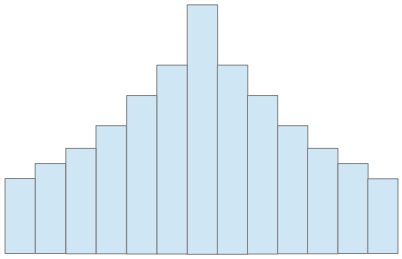
1. when calculating the score at consecutive positions i and $i+1$, there is an overlap of $D-1$ positions, i.e. redundancy.

2. calculating the score involves D columns in contrast to the column-column amino-acid score which considers only a single column.

Thus, the default scoring scheme:

$$S_{total} = S_{aa} + w_{\text{ctx}} S_{\text{ctx}} \tag{5.35}$$

is adapted to corrected for both effects. We assume that in a context state of width D the central column has the highest weight of one and it drops to the sides:



Supposing the decay is exponential, then the area A in blue is:

$$A = 1 + 2\beta \frac{1-\beta^d}{1-\beta}, \;\; d = \lfloor D/2 \rfloor. \tag{5.36}$$

We assume the redundancy to be a linear function of A:

$$r(A) = 1 + \frac{1-A}{D-1}. \tag{5.37}$$

This is a linearly decreasing line with $r(A) = 1$ for only one column. We finally correct the weights of S_{aa} and S_{ctx} as follows:

$$\left(1 - \frac{w_{\text{ctx}}}{w_{\text{center}}\, A} r(A)\right) S_{\text{aa}} + \frac{w_{\text{ctx}}}{w_{\text{center}}\, A} S_{\text{ctx}}. \tag{5.38}$$

The denominator $w_{\text{center}}\, A$ corrects for the area, $r(A)$ corrects for overlaps. Given context states of only one column, $D = 1$, then $r(A) = 1$, $A = 1$ which looks reasonable.

`str` alphabet score

Apart from secondary structure, we tested a fine-grained structural alphabet `str`, which was explicitly developed to improve the alignment quality in difficult cases for remote homology detection and homology modeling and which performed well in several CASP competitions (Karchin et al., 2003). It is an enhanced version of the DSSP alphabet (Kabsch and Sander, 1983) which subdivides the E-letter ($\beta-$strand) into six cases according to properties of a residue's relationship to its strand partners. We applied the improved four-layer neural networks of (Katzman et al., 2008) and determined for all query and template residues the probabilities for each of the 13 letters in the `str` alphabet.
We denote $p^{\text{str}}_{\boldsymbol{X}}(i,s)$ as the probability for letter $s \in \{1,\ldots,13\}$ at position i of profile $\boldsymbol{X}$ and similarly $p^{\text{str}}_{\boldsymbol{Y}}(j,s)$ for the $\boldsymbol{Y}$. The `str` structural score S_{str} is defined as a log-sum-of-odds score in analogy to eqs.(5.32,5.33):

$$S_{\text{str}}(i,j) = \sum_{s=1}^{13} \frac{p^{\text{str}}_{\boldsymbol{X}}(i,s)\, p^{\text{str}}_{\boldsymbol{Y}}(j,s)}{p^{\text{str}}_{\text{bg}}(s)}, \tag{5.39}$$

where $p^{\text{str}}_{\text{bg}}$ is the background probability for `str` state s in a large set of proteins. This `str` score was added to the total score with its own optimized weight.

5.3.2. Data sets

First, we filtered the SCOP (V1.75, Lo Conte et al., 2000) to obtain a set with a maximum pairwise sequence identity of 20% and enriched each SCOP20 sequence by generating a multiple sequence alignment with our iterative HMM-HMM searching tool `hhblits` (Remmert et al., 2012) (2 iterations against uniprot20 with standard parameters). Then each MSA was converted into an HMM via `hhmake` (Söding et al., 2013) with standard parameters. Finally, the dataset was divided into two sets by assigning the members of every fifth fold into a smaller set S_{train} (1492 domains) and the rest into a set S_{test} (5426 domains). Query and templates for the training and optimization set are then sampled from S_{train}, whereas test alignments are sampled from S_{test}. This procedure is important to ensure that none of the sequences in the test sets are homologous to any of the sequences in the training and optimization sets.

5.3.3. Parameter optimization

Since the time to compute the context score is proportional with K, we need to keep K low in order not to significantly slow down HHSEARCH. The improvements between $K = 128$ and $K = 32$ were moderate, so we chose $K = 32$. As $D = 13$ for the window width was found to perform well in various related applications (e.g. Biegert and Söding, 2009) we chose the same value without further optimization.

We needed to optimize the parameters w_{center} and β describing the weights w_j in section 5.2.2 and the weight w_{ctx} of the context score in eq. (5.34). We found that we could get better results by using separate parameter sets for training the context states library $(w^{\text{tr}}_{\text{center}}, \beta^{\text{tr}})$ (where no weight w_{ctx} is needed) and for the alignment stage $(w^{\text{al}}_{\text{center}}, \beta^{\text{al}}, w_{\text{ctx}})$. Since systematic testing of $w^{\text{tr}}_{\text{center}}$ and β^{tr} requires to generate a context library for each setting and furthermore the performance then depends on the other parameters, these were also adapted from (Biegert and Söding, 2009) ($w^{\text{tr}}_{\text{center}} = 1.3$ and $\beta^{\text{tr}} = 0.85$), so that the left and rightmost columns in a context profile get a weight $w_{j=-6} = w_{j=6} = 0.49$. We checked libraries with lower $w^{\text{tr}}_{\text{center}} = 0.2$ and 0.5, but this led to flatter context states and a drop in performance.

To optimize the alignment algorithm parameters, we performed a grid search for $w^{\text{al}}_{\text{center}} \in \{0.2, 0.25, 0.5, 1\}$ and $w_{\text{ctx}} \in \{0.8, 0.9, 1, 1.1, 1.2\}$ and measured the average of alignment sensitivity and precision on 1000 pairwise alignments where query and template were sampled from S_{train}. We obtained best results for $w^{\text{al}}_{\text{center}} = 0.2$ and $w_{\text{ctx}} = 1$. Surprisingly, $w^{\text{al}}_{\text{center}}$ turned out to be clearly smaller than 1 so that the context states become flatter during scoring.

We optimized the parameters $(w_{\text{ctx}}, w^{\text{al}}_{\text{center}}, \text{corr})$ specifically for the ROC5 homology detection benchmark by maximizing the area under the ROC5 curve, using the same context state library as in the alignment quality benchmarks. We used all sequences in S_{test}. In addition to w_{ctx} and $w^{\text{al}}_{\text{center}}$, the parameter corr from HHSEARCH was reoptimized. Differing from the setting for alignment quality, we arrived at $w_{\text{ctx}} = 0.6$, $w^{\text{al}}_{\text{center}} = 0.4$ and $\text{corr} = 0.2$. The optimization of the secondary structure score weight and the `str` alphabet weight were done on the same set and yielded $w_{\text{ss}} = 0.25$, $w_{\text{str}} = 0.12$.

5.4. Results

5.4.1. Training

We sampled up to 10 pairs of proteins in S_{train} and accepted if their `tmalign` score was between 0.5 and 0.85. If a window of width $D = 13$ centered at a structurally aligned

residue pair had at least 9 pairs within a distance of five angstroms in the structural alignment, the window was selected as a training sample and the D columns in the two corresponding count profiles were cut out. This procedure returned 141.508 training profile window pairs from 2987 pairwise alignments. Subsequently, these training pairs were filtered by calculating the mean column score S_{aa} (equation (5.33)) over all $D = 13$ columns and we rejected the trivial cases with $S_{\text{aa}} > 1.5$ (46.839 samples). At the beginning of training, we initialized the context states library randomly and ran 25 EM iterations. Different initializations led to quite similar log-likelihood values and libraries so that the training appears to be very robust.
A graphical visualization of the context states library is given in Figure 5.2.

5.4.2. Alignment quality

Alignments of high quality are important for many applications such as homology modeling and phylogenetic reconstructions. To differentiate between hard and easier cases, we created two sets of pairwise alignments: first, we randomly sampled 6000 query-template pairs from S_{test} under the condition that both query and template were in the same SCOP superfamily but in a different family ("hard set"). Furthermore, their `tmalign` score (Zhang and Skolnick, 2005b) had to be in the range between 0.5 and 0.85. The second set ("easier set") consists of 3000 pairwise alignments sampled from S_{test} where both query and template had to be in the same SCOP family and their `tmalign` score was restricted to lie between 0.6 and 0.95. The mean `tmalign` score of the alignments in the hard set was 0.61, while it was 0.72 in the easier set.
Whereas structural similarity reflects evolutionary divergence, the difficulty for an HMM-HMM alignment algorithm also depends strongly on the amount of evolutionary information available in the two profile HMMs. Even structurally quite similar pairs can be difficult to align when their profile HMMs were only trained on thin MSAs with few homologous proteins. Vice versa, even very remote homologs can often be reliably aligned when their profile HMMs were trained on thick, diverse MSAs.
To test the influence of the context similarity score on the amount of evolutionary information available in the profile HMMs, we created variant test sets of HMMs trained on MSAs with low diversity. These reflect better the diversity of MSAs encountered in practice than the typically rich and diverse MSAs from sequences in the SCOP, which mostly belong to large, very well studied protein families. To this end, we reduced the number of effective sequences (Neff) of the MSAs to a maximum value of 3 by using `hhfilter` (Remmert et al., 2012) with the `-neff 3` option. This command removes the

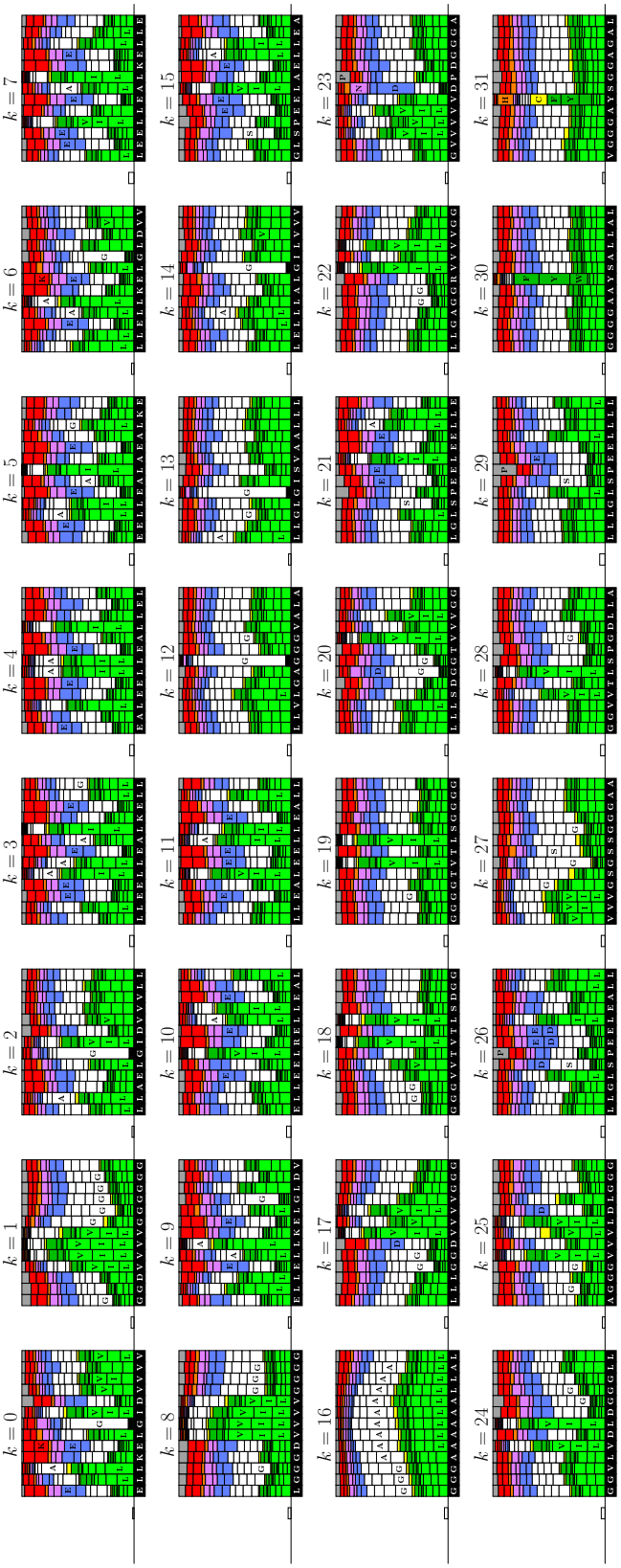


Figure 5.2.: Library of 32 context states: this context states library has been obtained by clustering aligned profile window pairs and thereby extracting the most conserved profile patterns within these training pairs. We use a color coding to visualize chemical properties of each amino acid (green: hydrophobic, blue: polar negatively charged, white: small, red: polar positively charged). Various different patterns can be identified, e.g. $k = 3, 4, 5$ are characteristic for aliphatic alpha helices.

sequences that are most diverged from the MSA's master sequence until the target Neff value is reached. Neff quantifies the diversity (or amount of evolutionary information) in an MSA (section 1.4.1). It is 1 for a single sequence and can reach at most 20. In summary, we have created four different test sets: $\text{hard}_{\text{Neff\_def}}$, $\text{hard}_{\text{Neff\_low}}$, $\text{easier}_{\text{Neff\_def}}$ and $\text{easier}_{\text{Neff\_low}}$.

We measured the alignment accuracy in terms of residue-based sensitivity and precision, where sensitivity $= TP/(TP + FN)$ and precision $= TP/(TP + FP)$. A true positive (TP) is a pair of residues that is aligned correctly, i.e., occurs in the reference alignment by `tmalign` (Zhang and Skolnick, 2005b). Similarly, a false positive (FP) is a residue pair occurring in the test alignment but not in the reference alignment. A false negative (FN) is a pair that is occurs in the reference alignment but not in the test alignment. All alignments were generated in global alignment mode using HHALIGN with option `-mact 0`.

We evaluated six different score combinations on each of the four benchmark sets (Tables 5.1–5.4: (1) the baseline version ("profile") that uses only the column score S_{aa} (eq 5.33) and no secondary structure score, (2) the secondary structure score based on PSIPRED predictions (Jones, 1999) for the first and 3D structure-based DSSP assignments for the second sequence of each pair ("ss") (Söding, 2005), (3) the secondary structure score based only on PSIPRED predictions for both sequences ("ss_{pred}") (Söding, 2005), (4) the sum of the score in (3) and the score based on the predictions of the 13-state `str` alphabet (eq. 5.39) ("ss+str") that was optimized for its positive impact on alignment quality (Karchin et al., 2003), (5) the context similarity score (5.32) ("ctx"), and (6) the sum of the score in (3) and the context similarity score (eq. 5.32) ("ss + ctx"). Versions (1) to (3) are already part of the hh-suite software, whereas scores (4) to (6) were added to the code of HHSEARCH and HHALIGN in this work.

Tables 5.1 and 5.2 show the results of the alignment benchmark for the hard test set with default diversity and with low diversity MSAs, respectively. As pointed out before, the score "ss_{pred}" that makes use of only predicted secondary structure performs almost as well as the score "ss" that requires the actual secondary structure of one of the aligned proteins, for high and low diversity MSAs. When combined with the secondary structure score based on DSSP, both the "str" alphabet-based score ("ss+str") and the context similarity score ("ss+ctx") lead to additional improvements, but these are clearly more pronounced for the "ss+ctx" score, which achieves the highest sensitivity and precision on high and low diversity MSAs. All three secondary structure classes profited to a similar degree from the additional scoring terms.

Interestingly, the improvements owing to the secondary structure scoring and to the

hard set - default Neff: $\text{hard}_{\text{Neff\_def}}$						
	profile	ss_{pred}	ss	ss + str	ctx	ss + ctx
PSIPRED		•	•	•		•
str				•		
ctx					•	•
DSSP			•	•		•
sens	0.435	0.464	0.471	0.478	0.464	**0.488**
prec	0.400	0.430	0.439	0.445	0.423	**0.451**
sens_h	0.456	0.481	0.487	0.491	0.489	**0.503**
prec_h	0.424	0.452	0.461	0.467	0.449	**0.473**
sens_e	0.467	0.501	0.515	0.524	0.494	**0.532**
prec_e	0.471	0.505	0.521	0.530	0.495	**0.535**
sens_c	0.372	0.395	0.401	0.406	0.394	**0.414**
prec_c	0.321	0.342	0.348	0.352	0.337	**0.357**

Table 5.1.: Residue-bases alignment sensitivity and precision of six versions of HHALIGN on 6000 pairwise alignments in the hard set with default diversity MSAs (average Neff 6.55). The upper part summarizes which information is used by each versions (PSIPRED predictions, 13-state `str` prediction (Katzman et al., 2008), the new context score, and the 7-state DSSP secondary structure assignments from the known 3D structure. The lower part gives the overall sensitivity and precision, below subdivided into helix (h) extended beta strand (e), and coil (c) residues, as assigned by DSSP.

context score are much stronger for low-diversity MSAs than for high-diversity ones (improvement of "ss+ctx" over "ss" of 3.6%/2.7% (sens/prec) for high-diversity MSAs and of 11.5%/9.2% (sens/prec) for low-diversity MSAs). While the secondary structure score relying only on predictions ("ss_{pred}") performs similarly to the context similarity score for high-diversity MSAs, the new context score is clearly superior for low-diversity MSAs.

On the easier dataset, secondary structure was still beneficial but the relative improvements in sensitivity/precision declined from +3.6%/ + 2.7% for more distantly related pairs to +1.3%/ + 1.1% for the easier cases. In contrast to the hard cases, the *str*-based score and the context scoring led to only minor gains.

But as for the hard set of protein pairs, when MSA diversity was low, *str* and in particular the context score again yielded significant improvements (Table 5.4) over the secondary structure score alone (sensitivity/precision gain: +1% and +0.9% for *str* and +4.8% and +3.9% for *ctx*, respectively). In summary, our new context similarity score consistently improved the alignment quality when combined with the standard secondary structure scoring in HHSEARCH. In the cases in which no secondary structure is available, the

hard set - low Neff: $\text{hard}_{\text{Neff\_low}}$						
	profile	ss_{pred}	ss	ss + str	ctx	ss + ctx
PSIPRED		•	•	•		•
str				•		
ctx					•	•
DSSP			•	•		•
sens	0.305	0.350	0.364	0.372	0.393	**0.406**
prec	0.288	0.331	0.346	0.354	0.363	**0.378**
sens_h	0.318	0.367	0.377	0.384	0.412	**0.420**
prec_h	0.303	0.350	0.364	0.371	0.386	**0.399**
sens_e	0.324	0.371	0.393	0.402	0.412	**0.434**
prec_e	0.338	0.387	0.409	0.418	0.420	**0.443**
sens_c	0.267	0.303	0.315	0.322	0.340	**0.350**
prec_c	0.234	0.266	0.278	0.283	0.291	**0.302**

Table 5.2.: Residue-bases alignment sensitivity and precision as shown in Table 5.1) but on the hard set with low diversity MSAs (averaged Neff 2.85).

easier set - default Neff: $\text{easier}_{\text{Neff\_def}}$						
	profile	ss_{pred}	ss	ss + str	ctx	ss + ctx
PSIPRED		•	•	•		•
str				•		
ctx					•	•
DSSP			•	•		•
sens	0.639	0.653	0.658	0.661	0.658	**0.667**
prec	0.611	0.625	0.632	0.635	0.627	**0.639**

Table 5.3.: Residue-based alignment sensitivity and precision based on 3000 pairwise alignments in the easier benchmark set.

context score ("ctx") also consistently improves alignment quality in comparison with the other purely sequence-based score ("ss_{pred}"). The extent of the improvements is greater the more difficult the alignment is: the more diverged the two proteins are and the lower the diversity of the MSAs is that their profile HMMs were trained on.

5.4.3. Comparison with the profile alignment tool PPAS

Next, we wanted to compare HHSEARCH/HHALIGN with other profile-profile alignment tools. Although many profile-profile alignment methods have been developed by the protein structure prediction community, we found that these methods were not designed to run independently of their protein structure prediction pipeline and could not be run

easier set - low Neff: $\text{easier}_{\text{Neff\_low}}$						
	profile	ss_{pred}	ss	ss + str	ctx	ss + ctx
PSIPRED		•	•	•		•
str				•		
ctx					•	•
DSSP			•	•		•
sens	0.532	0.564	0.573	0.578	0.572	**0.601**
prec	0.518	0.549	0.559	0.564	0.548	**0.581**

Table 5.4.: Sensitivity and precision of 3000 pairwise alignments as in Table 5.3 for low-diversity MSAs.

on user-supplied MSAs or sequence profiles. Since our goal here is to compare alignment methods and not methods to generate sequence profiles, we could not benchmark these tools. Finally, we found that PPAS, a profile-profile alignment tool developed in the lab of Yang Zhang, could be modified to run on user defined database profiles. Hence we compared the context state version of HHALIGN with PPAS. Its developers found that it gave equal or better results than HHSEARCH on a benchmark with hard and medium targets (Yan et al., 2013) and performed only slightly worse than MUSTER from the same group (Wu and Zhang, 2008), an extension of PPAS that includes several 1D structure-based scores. Unfortunately, we were not able to benchmark MUSTER as it could only search through precomputed template profiles and it was impossible to create the template-side 1D structure profiles etc. oneself.

Since PPAS requires profiles in PSI-BLAST format, we converted our template MSAs into PSI-BLAST format by calling `blastpgp` with the `-C` option and a dummy database containing a single sequence only. Yet for the query we had to keep the dependency on the PSI-BLAST output, because PPAS needs to parse it directly. For the default Neff benchmarks ($\text{easier}_{\text{Neff\_def}}, \text{hard}_{\text{Neff\_def}}$), we ran PPAS with three PSI-BLAST iterations, and used the default MSAs from S_{test} for HHALIGN. For the low Neff benchmarks ($\text{easier}_{\text{Neff\_low}}, \text{hard}_{\text{Neff\_low}}$), we reduced the number of iterations to two, the minimum valid value for PPAS to run. This resulted in an average diversity of Neff = 5.9 for the easy set and 5.67 for the hard set which is clearly above our filtered low Neff MSAs (Neff = 2.84). Thus, we converted these query PSI-BLAST alignments into a format readable by HHALIGN, ensuring that PPAS and HHALIGN received the same input.

We compared PPAS with two versions of HHALIGN: version (3) with secondary structure scoring based on PSIPRED and DSSP and version (6) that additionally includes the context score. Bot versions exceeded PPAS's sensitivity and precision by $14 - 17\%$

hard set						
	default Neff			low Neff		
	PPAS	HHalign		PPAS	HHalign	
		ss	ss+ctx		ss	ss+ctx
sens	0.415	0.471	**0.488**	0.392	0.425	**0.454**
prec	0.388	0.439	**0.451**	0.370	0.401	**0.423**

Table 5.5.: Sensitivity and precision of 6000 pairwise alignments in the hard benchmark set. PPAS makes use of predicted and DSSP secondary structure. It is compared with HHalign with secondary structure and context score.

easier set						
	default Neff			low Neff		
	PPAS	HHalign		PPAS	HHalign	
		ss	ss+ctx		ss	ss+ctx
sens	0.607	0.658	**0.667**	0.599	0.630	**0.647**
prec	0.585	0.632	**0.639**	0.581	0.610	**0.623**

Table 5.6.: Sensitivity and precision of 3000 pairwise alignments in the easier benchmark set.

on the hard set and $7 - 10\%$ on the easier set. Surprisingly, HHsearch even without secondary structure information outperformed PPAS (data not shown).

5.4.4. Application to homology modeling

A critical step in homology modeling is the generation of accurate alignments between the query and template sequences. From the Msa of the query and template sequences and the template structures, a three dimensional model for the query protein is built.
To test the impact of the improved alignment quality, we built homology models from the query-template alignments generated by the three tool versions described in the previous subsection. We used MODELLER (Sali and Blundell, 1993) to construct the 3D structural models, the most widely used tool for homology modeling. MODELLER is based on a maximum likelihood approach to optimize the fit of the 3D structure to a list of distance restraint probability functions, extracted from the query-template alignment in combination with the template structures. The results of the homology modeling benchmark are shown in Table 5.7.
As expected, the better alignments lead to better homology models: The context score in HHalign improved models on the hard set on high diversity Msas by 10.2% (default Neff) and 9.9% (low Neff) over PPAS models and on the easier set by 6.4% (default Neff)

set	Neff	PPAS	HHalign	
			ss	ss+ctx
hard	default	0.458	0.495	**0.505** (+2.0%)
	medium	0.444	0.468	**0.488** (+4.2%)
easier	default	0.610	0.644	**0.649** (+0.8%)
	medium	0.604	0.628	**0.640** (+1.9%)

Table 5.7.: Mean TMscores of 3D homology models built based on the pairwise query-template alignments from the three profile-profile alignment methods. Methods are tested on the same hard and easy set of query-template protein pairs as in the previous section. A diverse and medium diversity set of Msas was built from the query and template sequences using three and two iterations of PSI-BLAST, respectively.

and 5.9% (low Neff). Again, the more more difficult the query-template alignments, the larger were the improvements due to the context similarity score.

5.4.5. Remote homology detection

When searching large databases like the PDB, numerous hits are typically found. Yet, in practice, often only of few of them are of interest. For homology modeling, for instance, it suffices to detect 1 to 5 suitable homologous templates. Consequently, it is important to rank homologous proteins on top. We therefore analyze the sensitivity for remote homology detection using a ROC5 plot. For each query protein, one computes the ROC5 value, that is the area under the ROC curve up to the fifth false positive. The ROC5 plot shows the fraction of queries for which the ROC5 value is above the threshold on the x-axis. An overall number that summarizes the performance on the ROC5 benchmark is the area under the ROC5 curve.

We performed an all-against-all search with HHsearch in local alignment mode (the standard setting for template searches in our HHpred structure prediction server) on the proteins in S_{test}. We defined members belonging to the same superfamily as true positives (TPs) and those of different folds as false positives (FPs). Pairs with both proteins within the four- to eight-bladed $\beta-$propellers (SCOP fold IDs *b*.66 - *b*.70) were treated as unknown, and the same for Rossmann-like folds (*c*.2 − *c*.5, *c*.30, *c*.66, *c*.78, *c*.79, *c*.111) and all low resolution protein (*i*.∗), as they are known to be homologous despite being classified into different folds in SCOP. The ROC5 analysis in Figure 5.3 shows that adding secondary structure ("ss") increases the (area under the ROC5 curve (AUC) from 0.583 to 0.609 (4.4%). *str* and *ctx* scoring give moderate improvements to

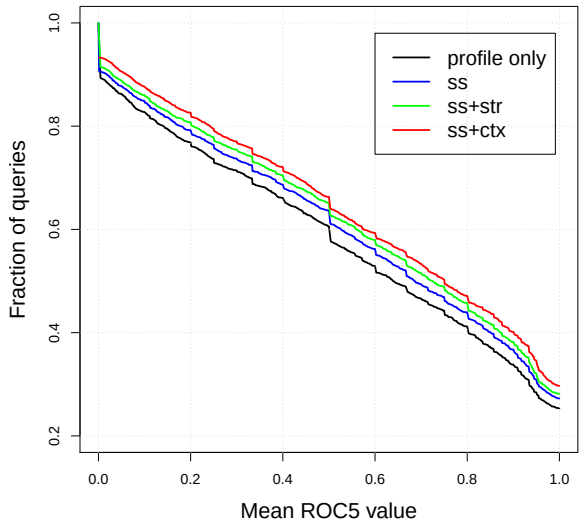


Figure 5.3.: ROC5 plot: Fraction of query HMMs whose ROC5 value is above the value on the x-axis. The ROC5 value for a query is the average sensitivity in a query-specific ROC plot up to the fifth false positive match.

0.625 and 0.641 (2.6% and 5.2% compared to "ss"), respectively.

5.5. Discussion

Context-specific pseudocounts Evolutionary constraints on the amino acids at each position of a given protein restrict their substitutions. Every single amino acid with its physico-chemical properties interacts with its environment in a characteristic way. These interactions highly dependent on the local structural and sequence context. For instance, a cysteine that is part of a zinc finger requires another cysteine in its neighborhood. In (Angermüller et al., 2012; Biegert and Söding, 2009), we exploited this concept to learn (in a generative maximum likelihood approach) a set of 4000 patterns best describing a diverse and large set of training sequence profiles. Using these, we could enrich sequences and sequence profiles with context-specific pseudocounts. This approach is implemented in all hh-suite programs since version 2.0.15. Hence, the improvements observed here come on top of those already reported for context-specific pseudocounts.

The difference between the previous approach and the one taken here is the degree to which we demand conservation of patterns. In (Angermüller et al., 2012; Biegert and Söding, 2009), conservation needed to be just good enough to leave a clear pattern in

the training profiles built from relatively closely related sequences. Here, in contrast, we use pairs of remotely homologous proteins and structural alignments, which are more reliable than HMM-HMM alignments at low sequence similarities, to find patterns that are highly conserved across large evolutionary distances (corresponding to the SCOP superfamily level).

1D structural properties In contrast to secondary structure similarity scores and similar scores based on the conservation of $1D$ structural properties, we take an unsupervised approach of learning the conserved patterns. Hence we do not need to know what particular property led to the conservation of the patterns we learn. The resulting context score is quite complementary to the secondary structure similarity score. Therefore, while we have not succeeded in capturing all possible conserved patterns in our 32-state library, we have manifestly learned conserved patterns whose information cannot be reduced to a 3-state or even 13-state alphabet of local backbone geometries.

Other unsupervised approaches Two approaches also take an unsupervised approach for learning local patterns to improve alignments. Ohlson et al. (2006) employ a self organizing map (SOM) to learn sequence profiles. They assigned SOM states to each local profile window in the two proteins to be aligned. To score aligned positions, they trained a neural network that took as input a profile-profile score, secondary structure values and grid coordinates of the aligned SOM states. Improvements due to the SOM states were small. Ma et al. (2012) and Ma et al. (2013) developed a non-linear extension of conditional random fields which includes as features the local sequence profile neighborhood. They reported substantial improvements in alignment quality, but unfortunately we could not compare their method to ours because it was not designed to accept user defined alignments.

Failed approach 1: discriminative learning We trained the context states library by maximizing the likelihood in equation (5.1). However, what we really would like to maximize is the discriminatory effect of the resulting score in eq. (5.32). We therefore put much effort into maximizing an objective function that equalled the sum of scores of positive training samples minus the sum of scores for negative training samples. Yet, this function is no longer a likelihood, precluding use of the EM algorithm. Moreover, it proved to be very prone to degenerate solutions unless we carefully enforced the restraint that the probability in the denominator in eq. (5.32) is equal to the average probability of that context state over all training states, which in turn allowed only for small learning rates. In addition, a more flexible target function also requires a significantly larger amount of training data to outplay a generative approach (Ng and Jordan, 2001; Angermüller et al., 2012).

Failed approach 2: transitions between context states Our model in section 5.2.2 assumes that both profile windows to be aligned stem from the same context state. We can drop this assumption by allowing transitions between context states k and k' and learning the matrix of transition probabilities $P(k'|k)$. Then, the score is

$$S_{\text{cs}}^{\text{smat}}(\boldsymbol{X}_n, \boldsymbol{Y}_n) = \log\left(\sum_{k=1}^{K} \frac{P(z_n = k|\boldsymbol{X}_n)}{P(z_n = k)} \sum_{k'=1}^{K} P(z_n = k'|\boldsymbol{Y}_n) P(k'|k)\right). \quad (5.40)$$

Inspecting the matrix revealed it being diagonal dominant as expected and some moderate substitution probabilities for similar states. In summary, though, alignment quality and sensitivity did not improve. We assume that $K = 32$ states are not yet fine-grained enough to necessitate substitutions and the effect of allowing transitions might become more important only for larger K.

Conclusion The new context score helps most in the difficult cases: (1) when little evolutionary information is contained in the HMMs to be aligned, and (2) when proteins are only remotely related. In the first case, integrating the sparse evolutionary information *vertically* within an MSA leads to only little noise suppression (i.e. the distinction of correct from incorrect alignments). Therefore we profit most from pulling together information *horizontally* along the MSAs. In the second case it makes sense to focus on the features that are best conserved among remote homologs, which is what our context score was trained to do.

Bibliography

The EMBO Conference: Critical assessment for protein structure prediction. **2010**.

S. Altschul, R. Carroll, and L. DJ. *Weights for Data Related by a Tree* **1989**;*207*:647–653.

S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic Local Alignment Search Tool. *J Mol Biol* **1990**;*215*:403–410.

S. F. Altschul, T. L. Madden, E. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research* **1997**;*25*(17):3389–402.

C. Angermüller, A. Biegert, and J. Söding. Discriminative modelling of context-specific amino acid substitution probabilities. *Bioinformatics* **2012**;*28*(24):3240–7.

H. M. Berman. The Protein Data Bank: a historical perspective. *Acta crystallographica Section A, Foundations of crystallography* **2008**;*64*(Pt 1):88–95.

A. Biegert and J. Söding. Sequence context-specific profiles for homology searching. *Proceedings of the National Academy of Sciences of the United States of America* **2009**;*106*(10):3770–5.

C. Bishop. Mixture Density Networks. *Technical report*, Aston University, **1994**.

G. M. Boratyn, C. Camacho, P. S. Cooper, G. Coulouris, A. Fong, N. Ma, T. L. Madden, W. T. Matten, S. D. McGinnis, Y. Merezhuk, Y. Raytselis, E. W. Sayers, T. Tao, J. Ye, and I. Zaretskaya. BLAST: a more efficient report with usability improvements. *Nucleic Acids Research* **2013**;pages 1–5.

Casp. The EMBO Conference: Critical assessment for protein structure prediction (CASP10). CASP committee, **2012**.

J. Cheng. A multi-template combination algorithm for protein comparative modeling. *BMC structural biology* **2008**;*8*(2):18.

J. a. R. Dalton and R. M. Jackson. An evaluation of automated homology modelling methods at low target template sequence similarity. *Bioinformatics* **2007**;*23*(15):1901–8.

A. Dempster, N. Laird, and D. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *J of the Royal Statistical Society Series B* **1977**;*39*:1–38.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. Biological sequence analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, **1998**.

S. Eddy. Profile hidden Markov models. *Bioinformatics* **1998**;*14*:755–763.

R. C. Edgar and K. Sjölander. A comparison of scoring functions for protein sequence profile

alignment. *Bioinformatics* **2004**;*20*(8):1301–8.

A. Elofsson. A Study on Protein Sequence Alignment Quality. *Proteins* **2002**;*339*(September 2001):330–339.

E. Faraggi, T. Zhang, Y. Yang, L. Kurgan, and Y. Zhou. SPINE X: improving protein secondary structure prediction by multistep learning coupled with prediction of solvent accessible surface area and backbone torsion angles. *Journal of computational chemistry* **2011**;*33*(3):259–67.

N. Fernandez-Fuentes, B. K. Rai, C. J. Madrid-Aliste, J. E. Fajardo, and A. Fiser. Comparative protein structure modeling by combining multiple templates and optimizing sequence-to-structure alignments. *Bioinformatics* **2007**;*23*(19):2558–65.

A. Fiser, R. K. Do, and A. Sali. Modeling of loops in protein structures. *Protein science : a publication of the Protein Society* **2000**;*9*(9):1753–73.

A. Hildebrand, M. Remmert, A. Biegert, and J. Söding. Fast and accurate automatic structure prediction with HHpred. *Proteins* **2009**;*77 Suppl 9*:128–32.

S. Hua and Z. Sun. A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach. *Journal of molecular biology* **2001**;*308*(2):397–407.

K. Illergard, H. Ardell, David, and A. Elofsson. Structure is three to ten times more conserved than sequence–a study of structural response in protein cores. *Proteins* **2009**;*77*(3):499–508.

D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of molecular biology* **1999**;*292*(2):195–202.

K. Joo, J. Lee, J.-H. Seo, K. Lee, B.-G. Kim, and J. Lee. All-atom chain-building by optimizing MODELLER energy function using conformational space annealing. *Proteins* **2009**;*75*(4):1010–23.

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

R. Karchin, M. Cline, and K. Karplus. Evaluation of local structure alphabets based on residue burial. *Proteins* **2004**;*55*(3):508–18.

R. Karchin, M. Cline, Y. Mandel-Gutfreund, and K. Karplus. Hidden Markov Models That Use Predicted Local Structure for Fold Recognition : Alphabets of Backbone Geometry. *Proteins: Structure, Function, and Genetics* **2003**;*51*(October 2002):504–514.

K. Karplus, R. Karchin, J. Draper, J. Casper, Y. Mandel-Gutfreund, M. Diekhans, and R. Hughey. Combining Local-Structure, Fold-Recognition, and New Fold Methods for Protein Structure Prediction. *Proteins* **2003**;*53*:491–96.

S. Katzman, C. Barrett, G. Thiltgen, R. Karchin, and K. Karplus. PREDICT-2ND: a tool for generalized protein local structure prediction. *Bioinformatics* **2008**;*24*(21):2453–9.

L. Kinch, S. Yong Shi, Q. Cong, H. Cheng, Y. Liao, and N. V. Grishin. CASP9 assessment of free modeling target predictions. *Proteins* **2011**;*79 Suppl 10*:59–73.

T. Kohonen. Self-Organized Formation of Topologically Correct Feature Maps. *Biol Cybernetics* **1982**;*69*:59–69.

J. Lafferty, A. McCallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Technical report*, **2001**.

P. Larsson, B. Wallner, E. Lindahl, and A. Elofsson. Using multiple templates to improve quality of homology models in automated homology modeling. **2008**;(1):990–1002.

R. A. Laskowski, M. W. MacArthur, D. S. Moss, and J. M. Thornton. *PROCHECK*: a program to check the stereochemical quality of protein structures. *Journal of Applied Crystallography* **1993**;*26*(2):283–291.

M. Levitt. Accurate Modeling of Protein Conformation by Automatic Segment Matching., journal = J. Mol. Biol. **1996**;*226*:507–533.

M. Levitt and M. Gerstein. A unified statistical framework for sequence comparison and structure comparison. *Proceedings of the National Academy of Sciences of the United States of America* **1998**;*95*(May):5913–5920.

S. Liu, C. Zhang, S. Liang, and Y. Zhou. Fold recognition by concurrent use of solvent accessibility and residue depth. *Proteins* **2007**;*68*:636–645.

L. Lo Conte, B. Ailey, T. J. Hubbard, S. E. Brenner, a. G. Murzin, and C. Chothia. SCOP: a structural classification of proteins database. *Nucleic acids research* **2000**;*28*(1):257–9.

J. Ma, J. Peng, S. Wang, and J. Xu. A conditional neural fields model for protein threading. *Bioinformatics* **2012**;*28*(12):i59–66.

J. Ma, S. Wang, F. Zhao, and J. Xu. Protein threading using context-specific alignment potential. *Bioinformatics* **2013**;*29*(13):i257–65.

V. Mariani, F. Kiefer, T. Schmidt, J. Haas, and T. Schwede. Assessment of template based protein structure predictions in CASP9. *Proteins* **2011**;*79 Suppl 1*:37–58.

D. S. Marks, T. Hopf, and C. Sander. Protein structure prediction from sequence variation. *Nature biotechnology* **2012**;*30*(11):1072–80.

J. Moult, K. Fidelis, A. Kryshtafovych, T. Schwede, and A. Tramontano. Critical assessment of methods of protein structure prediction (CASP)–round x. *Proteins* **2014**;*82 Suppl 2*(October):1–6.

U. Mückstein, I. Hofacker, and P. Stadler. Stochastic pairwise alignments. *Bioinformatics* **2002**; *18*(ii):153–160.

S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology* **1970**;*48*(3):443–53.

A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Adv Neural Inf Proc Syst* **2001**;**14**:841–848.

T. Nugent and D. T. Jones. Accurate de novo structure prediction of large transmembrane

protein domains using fragment-assembly and correlated mutation analysis. *Proceedings of the National Academy of Sciences of the United States of America* **2012**;*109*(24):E1540–7.

T. Ohlson, V. Aggarwal, A. Elofsson, and R. M. MacCallum. Improved alignment quality by combining evolutionary information, predicted secondary structure and self-organizing maps. *BMC bioinformatics* **2006**;*7*:357–66.

N. Ohsen, I. Sommer, and R. Zimmer. Profile-profile alignment: a powerful tool for protein structure prediction. *Pac Symp Biocomput* **2003**;pages 252–263.

X. M. Pan. Multiple linear regression for protein secondary structure prediction. *Proteins* **2001**; *43*(3):256–9.

J. Pei and N. V. Grishin. AL2CO: calculation of positional conservation in a protein sequence alignment. *Bioinformatics* **2001**;*17*(8):700–12.

J. Peng and J. Xu. Boosting Protein Threading Accuracy. *Res Comput Mol Biol* **2009**;*5541*:31–45.

J. Peng and J. Xu. Low-homology protein threading. *Bioinformatics* **2010**;*26*(12):i294–300.

J. Peng and J. Xu. RaptorX: exploiting structure information for protein alignment by statistical inference. *Proteins* **2011**;*79 Suppl 1*:161–71.

D. Przybylski and B. Rost. Improving fold recognition without folds. *Journal of molecular biology* **2004**;*341*(1):255–69.

M. Remmert, A. Biegert, A. Hauser, and J. Söding. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. *Nature methods* **2012**;*9*(2):173–5.

A. Sali and T. Blundell. Comparative Protein Modelling by Satisfaction of Spatial Restraints. *J Mol Biol* **1993**;*234*:779–815.

T. Schwede, J. Kopp, N. Guex, and M. Peitsch. SWISS-MODEL: an automated protein homology-modeling server. *Nuc Acid Res* **2003**;*31*:3381–85.

T. Smith and M. Waterman. Identification of Common Molecular Subsequences. *Journal of Mol Biol* **1981**;*147*:195–7.

J. Söding. Protein homology detection by HMM-HMM comparison. *Bioinformatics* **2005**; *21*(7):951–60.

J. Söding and M. Remmert. Protein sequence comparison and fold recognition: progress and good-practice benchmarking. *Current opinion in structural biology* **2011**;*21*(3):404–11.

J. Söding, M. Remmert, and A. Hauser. HH-suite for sensitive sequence searching based on HMM-HMM alignment., **2013**.

L. Song, C. Zhang, S. Liang, and Y. Zhou. Fold recognition by concurrent use of solvent accessibility and residue depth. **2007**;*68*:636–45.

F. Teichert, J. Minning, U. Bastolla, and M. Porto. High quality protein sequence alignment by combining structural profile prediction and profile alignment using SABER-TOOTH. *BMC*

bioinformatics **2010**;*11*:251.

B. Wallner and A. Elofsson. All are not equal : A benchmark of different homology modeling programs. *Protein Science* **2005**;*15*(3):1315–1327.

G. Wang and R. L. Dunbrack. Scoring profile-to-profile sequence alignments. *Protein Sci* **2004**; *13*:1612–1626.

Z. Wang, J. Eickholt, and J. Cheng. MULTICOM: a multi-level combination approach to protein structure prediction and its assessments in CASP8. *Bioinformatics* **2010**;*26*(7):882–8.

Z. Wang, F. Zhao, J. Peng, and J. Xu. Protein 8-class secondary structure prediction using conditional neural fields. *Proteomics* **2011**;*11*(19):3786–92.

S. Wu and Y. Zhang. MUSTER: Improving protein sequence profile-profile alignments by using multiple sources of structure information. *Proteins* **2008**;*72*(2):547–56.

J. Xu. Fold recognition by predicted alignment accuracy. *IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM* **2005**;*2*(2):157–65.

Y. Xu and D. Xu. Protein Threading using PROSPECT: Design and Evaluation. *Proteins* **2000**; *40*:343–54.

R. Yan, D. Xu, J. Yang, S. Walker, and Y. Zhang. A comparative assessment and analysis of 20 representative sequence alignment methods for protein structure prediction. *Scientific reports* **2013**;*3*:2619.

Y. Yang, E. Faraggi, H. Zhao, and Y. Zhou. Improving protein fold recognition and template-based modeling by employing probabilistic-based matching between predicted one-dimensional structural properties of query and corresponding native properties of templates. *Bioinformatics* **2011**;*27*(15):2076–82.

X. Ye, G. Wang, and S. F. Altschul. An assessment of substitution scores for protein profile-profile comparison. *Bioinformatics* **2011**;*27*(24):3356–63.

W. Zhang, S. Liu, and Y. Zhou. SP5: improving protein fold recognition by using torsion angle profiles and profile-based gap penalty model. *PloS one* **2008**;*3*(6):e2325.

Y. Zhang. I-TASSER server for protein 3D structure prediction. *BMC bioinformatics* **2008**; *9*:40.

Y. Zhang. I-TASSER: Fully automated protein structure prediction in CASP8. *Proteins* **2009**; *77*:100–13.

Y. Zhang and J. Skolnick. SPICKER : A Clustering Approach to Identify Near-Native. *J Comput Chem* **2004**;*25*:865–871.

Y. Zhang and J. Skolnick. The protein structure prediction problem could be solved using the current PDB library. *Proceedings of the National Academy of Sciences* **2005**a;*102*(4):1029–34.

Y. Zhang and J. Skolnick. TM-align: a protein structure alignment algorithm based on the TM-score. *Nucleic acids research* **2005**b;*33*(7):2302–9.