Identifying near-native multi-fragment sequence alignments in protein structure prediction

Fabian Salomon

Robotics and Biology Laboratory

Technische Universität Berlin

A thesis submitted for the degree of

Diplom-Ingenieur für technische Informatik

2012/05

Die selbstständige und eigenhändige Ausfertigung versichert an Eides statt
Berlin, den 24.05.2012

1. Reviewer: Prof. Dr. Oliver Brock
2. Reviewer: Prof. Dr. Manfred Opper

Abstract

The introduction of protein fragment libraries in the mid-1990s meant a huge leap forward for protein structure prediction. With them, one was able to combine the best matching parts from a "scrapyard" of protein parts instead of focusing on just one template protein.

Up until today, commonly used fragment libraries only contain relatively small, independent fragments. Consequently, these libraries can only model the (sequentially) local context, but can't model structurally conserved regions that are sequentially discontiguous. We therefore developed a library of so called "building blocks". A building block is a set of structurally contiguous, sequentially discontiguous fragments found in two or more proteins.

Existing methods of scoring a sequence alignment can only score each fragment independently or as a contiguous sequence (including the inbetween parts). They are therefore not optimally suited for scoring building block alignments. This thesis examines how the knowledge about the dependency between building block fragments can be exploited for a more specific scoring.

In order to achieve this, a number of different features is calculated for each building block that aligns sequentially to a certain target. The goal is to be able to combine these features in a way that one can distinguish alignments that are similar to the native structure of the target from those which are not.

Judging from the performance on CASP9 targets, the proposed setup appears to work well for template based modeling targets. Using the setup, I was able to cover 61 targets (15 more than the control) with 100 % near-native building block matches. Both the proposed setup and the control achieved roughly the same number of residues that were covered with near-native matches.

Abstract

Normalerweise enthalten Protein-Fragment Bibliotheken relativ kleine, unabhängige Fragmente, die zusammengesetzt werden können um die Vorhersage einer Proteinstruktur in die richtige Richtung zu leiten. Wir haben eine Bibliothek von sogenannten "Building Blocks" entwickelt. Ein Building Block ist eine Menge von strukturell zusammenhängenden, sequentiell unzusammenhängenden Fragmenten. Bestehende Methoden um ein Sequenz-Alignment zu bewerten können Fragmente nur unabhängig voneinander oder als zusammenhängende Sequenz (inklusive der dazwischenliegenden Abschnitte) bewerten. Diese Diplomarbeit untersucht, wie das Wissen über die Abhängigkeiten zwischen Building Block Fragmenten genutzt werden kann, um eine spezifischere Bewertung zu erhalten. Dazu wird eine Anzahl unterschiedlichster Features für jeden Building Block, der sequentiell an einem bestimmten Zielprotein aligned, berechnet. Das Ziel ist es, diese Features so zu kombinieren, dass man Alignments, die ähnlich der nativen Stuktur eines Zielproteins sind, erkennen kann. Ausgehend von den Ergebnissen auf CASP9-Zielproteinen scheint der vorgeschlagene Ansatz gut für die Vorsage von Proteinen aus der TBM-Kategorie geeignet.

Contents

List of Figures

LIST OF FIGURES

Glossary

CASP "Critical Assessment of Techniques for Protein Structure Prediction": Biannual competition/benchmark starting in May 2012 for the 10th time (CASP10). Data from the previous run (CASP9) is used to benchmark the features presented in chapter 5.

correct match a match that is superimposable to the corresponding regions of the target structure with an RMSD (or statistical significance) below a certain threshold

FM "Free modeling": Targets that do not have a protein with detectable homology have to be predicted "de novo" and are therefore more challanging than TBM targets.

match a sequential (building block) match as found by our retrieval module; not necessarily a "correct match"

PDB "Protein Data Bank": A database of experimentally determined protein 3D-structures. Freely accesible at http://www.pdb.org/

PDBSS The subset of the PDB that we used to generate the building block database on. See section 2.2

RMSD "Root mean squared distance" calculated between the atoms of two structures given an optimal superimposition. Usually only calculated for the backbone atoms. Specifically, in this thesis, the RMSD is calculated between a building block match and the corresponding residues of the target protein; see section 4.4

SCOP "Structural Classification of Proteins": a humanly curated hierarchical classification of protein structural domains. See also section 5.7

TBM "Template based modeling": If the PDB contains a homolog for a particular target, the homolog can be used as a template. Such targets are relatively easy to predict with high accuracy.

1

Introduction

1.1 Protein structure prediction

Anfinsen's dogma [Anfinsen, 1973] states that generally the three-dimensional shape of a protein in its native conformation is defined by its amino acid sequence. Once a protein is synthesized (for example from DNA in a cell) it folds into this native state which has the lowest kinetically accessible Gibbs free energy. Since the postulation of this strong relationship between sequence and structure, it has been the dream of researchers to be able to predict the atom positions of a protein's native state without knowing anything except its sequence. Protein structure prediction is nowadays one of the most important topics in bioinformatics, largely due to the potential impact that a "solution" to that problem would have on drug design and other biological use cases.

Unfortunately, while we have a good understanding of inter-atom forces, it is not viable to search the lowest energy state by enumerating all possible atom configurations. The number of possible conformations and thus computation time grows unimaginably high even for simplified models and relatively small proteins; this problem is commonly referred to as "Levinthal's paradox": even though a protein finds its energy minimum within negligible time, all of the world's computing power is not sufficient to replicate this. In order to still be able to predict the native conformation of a protein, one has to significantly reduce the search space. To achieve this, knowledge about the problem domain has to be employed. In most cases, this means that researchers use structural information that has been determined through experimental methods like x-ray diffraction, nuclear magnetic resonance spectroscopy (NMR) and electron

microscopy. This information is publicly available through the Protein Database (PDB).

Up until today, the problem of protein structure prediction has not been sufficiently solved and it is questionable if it ever will be. However, methods have emerged that can – in many cases – predict a structure that is reasonably close to the native structure. Most of these methods are based on finding proteins with matching sequence and then using their structural information as a starting point for searching a nearby local energy minimum. The reasoning behind this is that similar amino acid sequences are probably evolutionary related (homologous) and thus will probably result in similar 3-dimensional arrangements. This is backed up by the finding that structure is more evolutionary conserved than sequence, i.e. sequence mutations happen more often than structure mutations and sequence mutations do not necessarily result in significant structural changes [Chothia and Lesk, 1986]. One of the reasons for this is that a protein's function depends on its structural properties; therefore mutations that significantly change a protein's structure are – on average – less likely to be evolutionary successful. Combinatorially, a protein of length n can have one of 20^n different sequences; thus a high sequence identity is a strong indication for evolutionary relatedness. This means that a protein might have mutated over time and even though a couple of amino acids might have been exchanged, inserted or deleted, the structure has remained relatively unaffected.

For target sequences that have close homologs ("templates") in the PDB, prediction quality is usually quite high. In other situations, called "(template) free modeling" or "de novo modeling" such close homologs do not exist; which makes good predictions much more challenging. Recent advances in the field of de novo modeling are based on the observation that proteins in the PDB can be assembled from a limited part list taken from multiple proteins that are unrelated amongst each other [Kolodny et al., 2002]. In comparison to homology modeling, fragment sizes are much smaller (typically 3 to 15 residues) and misleading information is much more common.

In the biannual CASP (Critical Assessment of Protein Structure Prediction) competition, research teams across the world are testing their prediction algorithms against each other. During the competition, protein sequences whose experimentally determined structure is soon to be published are broadcast to all contestants. Within limited time, contestants have to return their best-bet structure prediction; making it impractical to use a lot of human intervention during the process. The target structures are selected

to cover proteins where homology modeling can provide useful information, as well as proteins where there are no close homologs available.

In the context of this thesis a new approach is developed to participate in the 10th CASP competition starting in May 2012. The outline of this approach is to extract a large number of "building blocks" from a set of known protein structures. For a query protein with unknown structure, a couple of sequentially matching building blocks are selected and used to constraint a energy minimization with a commonly used process known as "Monte Carlo simulated annealing" [Kirkpatrick et al., 1983]. The following sections describe the building blocks approach in more detail and specify the goals and contribution of this thesis.

1.2 Building block definition

The protein building block definition as described here was originally formulated by former research team member Nasir Mahmood. Syntactically, a building block instance is a set of sequentially disjunct protein fragments that originate from the same protein chain; it can be fully described by naming the PDB accession code, the chain ID and the residue number ranges. On the semantic level, the primary condition is that there exists another building block instance from a *different* protein whose backbone atoms can be superimposed with an RMSD lower than a certain threshold (currently around 5 Å). The fragments have to be spatially contiguous, i.e. there shall be atom-level contacts between the fragments.

This semantic implies that a pair of building block instances always has the same amount of fragments and the same number of residues per fragment, because only such fragments can be superimposed with each other. The discovery of building blocks is done with a structural aligner that is based on calculating the mutual distances between CA atoms within each protein chain, resulting in a symmetric distance matrix. The aligner then tries to find similar sub-matrices for each pair of proteins which act as seeds for superimposition attempts. Our implementation is based the textual description in Lessel and Schomburg [1994].

There are additional conditions, for example a pair of building blocks has to have similar secondary structure and similar solvent accessibility values as calculated by DSSP [Kabsch and Sander, 1983]. Other filters and post-processing steps are applied as

well. As these are subject to frequent adjustments, their exact definitions are omitted here. Their common purpose it to weed out "chance alignments" and generally gear the selection to be in sync with the rationale described below.

An example of a three-fragment building block can be seen in fig. 1.1 and fig. 1.2.

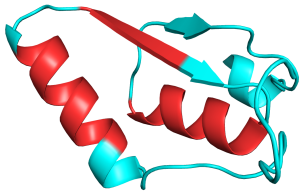


Figure 1.1: Residues 187–251 of 3be7A (rendered with PyMOL)

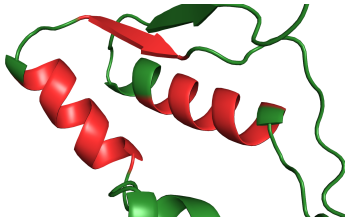


Figure 1.2: Residues 296–371 of 1vemA (rendered with PyMOL)

1.3 Building blocks rationale

Statistics of the PDB[1] show that even though the database is growing rapidly, the growth of "unique folds" seems to go down. According to Kinch et al. [2011], the 2010 CASP run (CASP9) only contained 4 entirely new folds (out of $\approx$ 120 targets). Zhang and Skolnick [2005] even go as far as to say that their "results are highly suggestive that the protein-folding problem can in principle be solved based on the current PDB library". So, one could hypothesize that the data to solve protein structure prediction is already there, it just has to be utilized properly.

To understand how the building block approach tries to unearth new potential, one has to point out the differences to existing fragment libraries. Most of them – see [Bujnicki, 2006] for a review of current fragment assembly methods and libraries – contain only relatively small, independent fragments. So the only context they have is from residues that are sequentially nearby (local context). Most protein folds are packed; thus, amino acids from different regions of the sequence can end up quite near in three-dimensional space. The atoms from such regions are close enough to significantly influence each others' force field. Such atoms are said to be "in contact". The basic idea of the building block approach is that these sequence-wise "long-range" or "non-local" interactions not only have a significant influence on the structures of the individual

[1] http://www.rcsb.org/pdb/statistics/contentGrowthChart.do?content=fold-scop

fragments; also, if two sequentially distant fragments and their non-local interactions are structurally conserved, they restrict the overall topology of the protein. If one would be able to retrieve such fragment pairs by sequence, one could use them to guide the energy minimization towards the global minimum more effectively than single-fragment approaches.

As mentioned before, protein structure is more conserved than sequence. It is believed that this is due to the fact that a protein's function, ligand binding abilities etc. depend on the protein structure. A sequence mutation that preserves the protein's function is therefore more likely to survive natural selection than one that doesn't. Protein function is often achieved through complex motion; such motion might in many cases be facilitated by contacts. This theory is most intuitively accessible for so called allosteric proteins: these proteins have two distinct sites, an active and a regulatory site. The binding of a ligand at the regulatory site changes the protein's behavior at the active site. Suel et al. [2002] stated that in these proteins, "a small subset of residues forms physically connected networks that link distant functional sites in the tertiary structure". Those residues were found to be evolutionary conserved across functionally related proteins and are therefore though to be responsible for stabilizing or even facilitating the proteins' function. Therefore we hypothesize that substructures that fulfill the building block criteria are more evolutionary conserved than other substructures. Based on this assumption, we hope that we can improve existing protein structure prediction approaches by creating and using a library of building blocks.

Other researchers are trying to model these interactions as well including Kifer et al. [2011] who are acknowledging the role of non-local interactions in the process of protein folding and are trying to model these by allowing relatively long, contiguous fragments. Daniluk and Lesyng [2011] are building "local descriptors" that allow sequence discontinuity "by identifying all residues in contact with [a] central residue" and expanding from there. They say that this "reflects approximately the range of local, most significant physico-chemical interactions between [a] central residue and other amino-acids.", thereby adding "a three-dimensional context to the local properties of proteins". Hvidsten et al. [2009] are even going as far as to say that the "lack of good long-range distance constraints is the main challenge in template-free modeling" and

local descriptors could play a significant role in upcoming protein structure prediction methods.

1.4 Project context

In the context of this thesis there are three research team members working on the building block database generation and utilization. There are also two other theses focusing on different aspects of the project. Before I describe the contribution of this thesis, I will describe the other works and their interaction in the CASP pipeline:

Retrieval After a new target sequence is received, matching building blocks have to be retrieved from the database through a sequence alignment. This module is developed by fellow student Stefan Dörr. The ground truth for his retrieval is the subset of building blocks that match correctly on the target's native structure within a certain threshold. For testing purposes, one can use proteins with known structure (for example from the last CASP run) and benchmark against their ground truth. The goal is of course to retrieve most of the building blocks in the ground truth without allowing too much noise.

Assembly The assembly modules predict the target's structure using a (sub-)set of the retrieved building block matches. Independent strategies are developed by fellow student Mahmoud Mabrouk and by research team member Michael Schneider. The basic concept is to turn the building block matches into constraints for the energy minimization. One mayor challenge here is to adapt to different situations: for some targets a huge number of building blocks can be retrieved, for some only a few. Some of these are redundant, some are complementary and some are even contradictory.

A constraint can be interpreted as a harmonic energy potential between two residues added to the energy function. The rigidity/slope of this energy potential is configurable. As by now we have very little experience using these constraints; in particular, it is unclear what a reasonable number of constraints would be and how tolerant the structure prediction is towards incorrect and/or mutually incompatible constraints. I hypothesize that a low number of correct (i.e. native-like), mutually compatible, very rigid constraints is to be preferred. This is because I suspect that many lax constraints

would just add noise to the energy landscape. Many rigid constraints on the other hand would take far too much precedence over the established energy minimization process. Non-mutually-compatible constraints would likely either cancel each other out or cause the prediction not to converge.

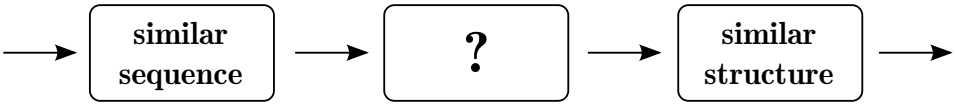
1.5 Goals

If using a small number of highly rigid constraints, it is vitally important that all of them are correct. For example, a building block with two fragments, one of them sequentially matching in the first third of the sequence and one of them in the last third of the sequence with a small spatial distance between them, would cause the rest of the prediction to accommodate the assumption that these two parts are very close to each other. This would severely influence the overall structure (unless the energy minimization "overrides" the constraints). One can easily imagine how it would cause the whole prediction to be far different from the native structure if that building block match were incorrect i.e. sequentially similar but structurally different from the target's native structure. So, for most of this thesis I will assume that the goal is to find a relatively small number of 100 % correct building block matches. It is acceptable if the necessary filtering comes at the cost of leaving a couple of residues uncovered (i.e. specificity is more important than sensitivity). Due to our CASP participation being more like a proof of concept than a dead-serious effort to win, I further assume that it is always preferable to have some constraints (even if they are all incorrect) than running a "vanilla" prediction without any constraints.

The retrieval module will probably always return a certain amount of matches that are not actually close to native. While the retrieval module focuses on optimizing sensitivity while keeping a reasonable specificity, this thesis takes the retrieval result and optimizes specificity while keeping reasonable sensitivity. So, the focus of this thesis is right between the retrieval and the assembly module(s).

On a higher level, the challenge here – and for protein structure prediction in general – is that retrieval is done on the target *sequence*, but the fragments are expected to match the target's *structure*. However, a good sequence alignment does not always translate to a fragment with correct local structure. One could say that between finding

a good sequence alignment and finding and correct structural match, there is a "black box" that is not yet sufficiently understood.



Existing efforts like HHSearch focus on making the sequence retrieval ever more sophisticated but each fragment is scored *independently*. I hypothesize that the information about the relationship between the fragments of a building block match can be used to get one step closer of modeling the "black box" between sequential and structural similarity. The building block criteria, namely the spatial constraints between fragments and the sheer co-occurrence of those fragments offer an opportunity to create more specific scoring features.

A first attempt of distinguishing correct from incorrect matches would be to use the fragment-wise metrics returned by the sequence aligner and somehow accumulate them to be applicable to two-fragment building block matches. The way that the retrieval module currently works is that it uses HHsearch to align building block fragments independently. Only if at least two fragments of the same building block match on the target sequence, the building block is considered a match. Because all of a building block's matched fragments are required to be similar to the native structure *individually* as a prerequisite for the building block being correct as a whole, it seems reasonable to choose the lowest of the fragment alignment scores as a (conservative) score for the whole building block. To test the performance of this feature ("seq_score_min"), we determined the RMSD of a building block compared to the target's native structure (for details see section 4.4). We consider a building block to be correct if its RMSD is smaller than 2.7 Å. Figure 1.3 shows that this feature alone cannot reliably predict whether or not a building block is correct, i.e. it is not possible to find a horizontal cut that separates correct from incorrect matches with reasonable accuracy. Even though there is some correlation for high values of seq_score_min, such values are relatively rare and it has to be anticipated that most target sequences do not yield even a single building block with such a high score. Other combinations of the various scores produced by HHsearch do not work much better either (see section 5.1).

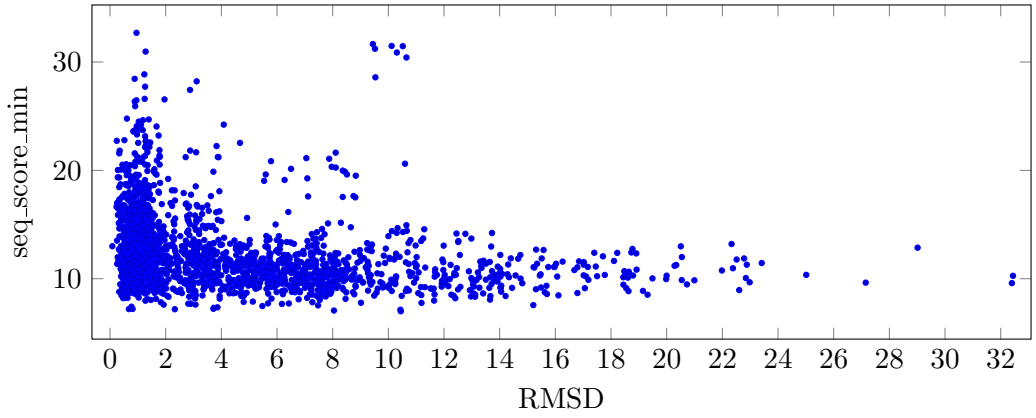


Figure 1.3: Feature seq_score_min: given the alignment scores for a pair of fragments, take the smaller of the two scores; for a detailed description of both axes, see section 4.4 and section 5.1

I do not think that I can improve on the way HHsearch calculates its scores, because these techniques (substitution matrices, sequence profile generation, e-values, secondary structure prediction) have been around for a long time and they have already been fine-tuned extensively. Thus, the goal of this thesis is to come up with different features that can augment HHsearch's scores to classify correct vs. incorrect building block matches more reliably. It has to be anticipated that no *single* biological aspect can be exploited to do so. Instead, one probably needs to combine a lot of different aspects to deal with the ubiquitous noise found in the biological domain. The nature of this thesis is therefore highly empirical; instead of focusing on a few very specific hypotheses and either proving or disproving them, I developed a setup that enabled me to "play" with the data, come up with new ideas for features, test them quickly and reiterate often. Sometimes I came up with a well formulated hypothesis and derived a feature that models it, sometimes I acted more opportunistically and tried to exploit a potential data source with any idea that came to mind.

1.6 Related work

Much of the biological motivation and related work differs from feature to feature, so I will explain most of it together with the features in chapter 5. Some related work for sequential clustering will be mentioned in chapter 3. This leaves us with related work

about the overall idea of identifying near-native alignments through various scores.

The simplest way of scoring a given sequence alignment is to calculate the sequence identity. It is the percentage of amino acids that are exactly the same in the target sequence and in the template's sequence. Unfortunately, this way of scoring is only suitable for very closely related sequences.

To conquer this, scientists developed amino acid substitution matrices like PAM [Dayhoff et al., 1978] and BLOSUM [Henikoff and Henikoff, 1992]. They are both based on the observation that certain amino acid pairs are more likely to be switched by point mutation than others and that some amino acids occur more often than others in nature. For example, a hydrophobic amino acid is relatively unlikely to be replaced by a hydrophilic amino acid and vice versa. Both matrices are derived from experimental data and have different versions for different expected mutation rates. Substitution matrices in their common form contain so called "log-odds" values, so that positive values indicate a likely evolutionary replacement and negative values a less-than-likely evolutionary replacement. To calculate the likelihood for a whole sequence, one just has to sum up the individual values. This sum/score can be interpreted as a "chance alignment" for negative and small positive values.

Most sequence alignment algorithms also have a way of allowing gaps in the alignments. Penalties for opening a new gap or expanding an existing gap are usually added to the log-odds sum. This approach however is not well suited for building block alignments because gaps are highly unfavorable *inside* a building block fragment, but the gap size is relatively irrelevant *between* building block fragments (c.f. section 5.5).

Another problem is that with a growing number of total fragments in a library, the number of chance alignments goes up. "To assess whether a given alignment constitutes evidence for homology, it helps to know how strong an alignment can be expected from chance alone." [Altschul, 1999]. E-value or expected value is a familiar concept in statistics and popularized in terms of sequence alignments by the BLAST algorithm: The idea is to not only take into account the cumulated log-odds of the substitution matrix, but also the size of the fragment database and the length of the sequence alignment. This is because big databases statistically have a higher likelihood of returning high scoring templates than small databases; especially for short query sequences, because they likely have many near-exact matches in the database. The E-value of a sequence alignment is therefore defined as the probability of observing its log-odds sum (or better)

in the database by chance, so low E-values are better than high ones. For example an E-value of 5 means that on average one would expect 5 distinct same length matches with similar or better log-odds in a random database of the same size.

For single-template homology modeling Sadowski and Jones [2007] assembled "a set of 732 targets for which a choice of ten or more templates exist with 30 – 80 % sequence identity". They then tested which sequence (profile) alignment algorithm (BLAST, PSI-BLAST, HHpred) ranked the templates most accurately. They also tested a couple of other algorithms and hypotheses that were suggested in various publications:

- a sequence conservation score (C_{valdar}),
- whether focusing on structurally-determined subsets improves the score's performance. Such subsets were determined by "secondary-structure state, solvent accessibility or proximity to atoms in HETATM records".
- "whether differing pH, temperature, quaternary state, resolution, and space group might have an effect on the choice of template"
- a model quality assessment program (MODCHECK) using pair-potentials,
- a linear combination of a total of 30 features amending MODCHECK by including hydrogen bonding, solvation, torsion angles, van der Waals interactions and stereochemical quality. Feature weights were optimized systematically.

They mentioned that "the question of how to select the best template from a set of alternatives has not previously received much attention" and that their study was "the first full study of template selection per se". Their findings were that BLAST could identify a template with an RMSD of 0.5 Å or better from the best possible template in 75 % of the cases. PSI-BLAST and HHpred performed only slightly better due to the high sequence identity. PH, resolution etc. did not improve the sequence alignment scores. The structurally-determined subsets did not perform better than the full sequences, but still produced acceptable results. Interestingly, coil regions performed nearly as good as the full sequences; the building block definition currently does not allow coil regions, though. MODCHECK was tested independently from the sequence alignment scores, but performed poorly. The amended version (MODCHECK-HD) however performed only 1 % worse than BLAST. A combination of MODCHECK-HD

and BLAST resulted in a 7 % improvement over BLAST alone. This research shows that template selection is a relatively untapped field that can improve sequence based scores. It also shows that some features could work well for a certain set of proteins (after all they were mentioned in other publications), but do not add any value across-the-board. Sadly, the findings can not directly be applied to the building block library because building block fragments are much shorter, can come from many relatively unrelated proteins and typically have much less sequence identity.

Simons et al. [1999] wanted to model the fact that local amino acid sequences bias "towards a small number of alternative local structures". Looking at single fragments, they have applied the Bayes theorem to calculate "the probability of a structure given the amino acid sequence":

$$P(\text{structure} \mid \text{sequence}) = P(\text{structure}) \cdot \frac{P(\text{sequence} \mid \text{structure})}{P(\text{sequence})}$$

This means that if a certain sequence fragment aligns very frequently, but relatively seldomly matches a target's structure, this can be seen as a negative factor if one has other options (c.f. section 5.11). The same group also developed the well known "I-sites" library [Bystroff and Baker, 1998]. The library consists of clusters of small sequence fragments and a representative structure for each of them. For each cluster they generated "a confidence curve [that] maps similarity score to the probability of correct local structure". That way, clusters that are often false-positives need higher similarity scores in order to achieve the same level of confidence as other clusters.

Bartlett and Taylor [2008] used a combination of sequence and physico-chemical scores to distinguish correct from incorrect folds. They used Statistical Coupling Analysis (see section 5.9) to find residue pairs in multiple sequence alignments (MSAs) that likely have co-evolved. They used the MSAs to predict a protein fold and found that decoys that had a small distance between the supposedly co-evolving residues were more likely to be correct than those with high distances.

An important difference between this and the approach in this thesis is that they tested their scores on actual predictions vs. the native structure. This is known as "fold selection", "fold discrimination" or "decoy discrimination", i.e. the use case is to generate a library of protein-like folds and identify the fold(s) that are probably close to the native structure. Fold selection techniques play an important role in protein structure prediction because many tactics do some kind of sampling (e.g. multiple runs

of simulated annealing) and need to choose the best sample as the final result. In contrast, this thesis tries to filter out incorrect matches before they are even used for the structure prediction. One could argue that fold selection is more tailored to the ultimate goal of making a good prediction. The problem is however that predictions are computationally expensive. Because there are relatively many building block matches per target sequence, the time complexity of creating a model for each possible subset of the matched building blocks would be impractical ($\mathcal{O}(2^n)$ for n matches). Therefore it is very beneficial to have a way of prioritizing matches.

Another fold selection paper, this time dealing with *ab initio* predictions, is Huang et al. [1999]. The authors combined three scores ("RAPDF", "HCF" and "shell") to select the most promising structure from "a library of 500 possible structures for each of 11 small helical proteins". HCF (Hydrophobic compactness function) is based on the observation that real proteins are packed or "globular" in order to bury hydrophobic residues (see section 5.6). This measure is simply "the square of the radius of gyration of the carbon atoms" c.f. Simons et al. [1997]. RAPDF (Residue-specific all-atom probability discriminatory function) and shell are both scores that use a database of known structures to score inter-atom and inter-residue distances, respectively. Shell [Park et al., 1997] sums up a residue-type dependent score for all non-adjoining residues that are in contact. A contact is assumed if both their "interaction centers" are within 7 Å. A residue contact score is used in section 5.8 as well.

Gao et al. [2009] tested different scores to predict the *local* quality of decoy fragments. Therefore they extracted all gaplessly aligned fragments and calculated the RMSD to the corresponding native structure fragment. Their goal was to predict this RMSD using sequence and template structure alone. They tested 8 types of features and did a feature selection to determine how leaving one type of feature out influences sensitivity and specificity of the predictions. According to the authors, their method is "the first method to directly predict the quality of fragments that are automatically determined by the sequence-structure alignments rather than fragments with fixed length". They found that even though "local sequence evolutionary information (i.e. sequence similarity) is the mayor factor in predicting local quality [...] structural information such as solvent accessibility and secondary structure helps to improve the prediction performance". One of their best performing features was the length of the fragment, so they decided to cancel this aspect out through a statistical significance score, see section 4.4 for an adaption

of this concept. Notable features were based on sequence profiles, secondary structure, solvent accessibility (c.f. section 5.6) and quality of the overall model (Z-Score). In terms of this thesis, sequence profiles and secondary structure are handled by HHsearch, but Z-Score is not applicable. Unfortunately, the relative importance of their features is not very relevant to building block matches because their single-fragment matches come from global alignments while this thesis is about multiple fragment matches that are locally aligned. Also, they were predicting the RMSD of the the decoy regions vs. native and this thesis tries to predict the RMSD of fragments vs. native.

2

A database of building blocks

Before we can dive into building block clustering and features, lets define what a building is on a lower level. This is necessary in order to understand what properties a building block has and what kind of data sources are available to calculate features from.

2.1 Storage format

The original output format as used by Nasir Mahmood's aligner (and initial post-processing) is an ASCII format spread over multiple folders (one folder for each pair of aligned chains). The format looks like this:

```
195lA:1b2sE
#1(2,23,2.4,EH) = 52-61:50-59 | 68-80:71-83
#2(3,24,3.1,HHH) = 4-14:20-30 | 22-27:79-84 | 31-37:101-107
```

In Extended Backus-Naur Form that is:

```
BblockFile = Header <EOL> { BBlock <EOL> }
Header = PdbId , ':' , PdbId
BBlock = '#' , BblockId , '(' , MetaData , ') = ' , Fragment ,
         { | Fragment }
MetaData = NumberOfFragments , ',' , NumberOfResidues , ',' ,
           Rmsd , ',' , SecondaryStructure
Fragment = Witness1StartResidue , '-' , Witness1EndResidue , ':' ,
           Witness2StartResidue , '-' , Witness2EndResidue
```

i.e. the header lists the two protein chains that were superimposed and then one building block per line. This format makes sense as an interchange/archive format because it does not require anything else than a file system and each parallel building block generation process can write to a different file.

However, this file format is not well suited for ongoing work and experiments: When brainstorming new ideas it is important to be able to get a quick impression about statistical properties of the dataset; for example one could be interested in a histogram of fragment counts per building block (see fig. 4.2). These statistics could help to identify mistakes or provide early pointers regarding the viability of ideas. If one would like to extract these statistics from the before-mentioned flat file format, one would have to write a new piece of code for every such query. Each time the query is run, the entire building block database would have to be parsed. Even more importantly, every time that someone wants to store another bit of meta information, existing parser code could stop working. This would be particularly cumbersome because several researchers are working on this dataset and everyone is using his/her favorite programming language for prototyping.

Therefore it was an obvious decision to use a relational database for storage. Nearly every programming language can access such a database via SQL with only a few lines of code. Existing SQL queries keep working when new columns are inserted into a table or new tables are added to the database. The storage format is relatively efficient and frequently accessed data can be indexed for better performance. Different incarnations of the dataset can be addressed through different SQL database names.

2.2 Data source and quality

When talking about biological data it is important to know that almost all data is flawed to some degree. This is because data is mostly determined experimentally and because in nature it's hard to find simple and across-the-board applicable rules. I like to think of nature as a million years old software suite that nobody ever cared to refactor.

Another thing that has a huge impact on the database and how suitable it is to predict new proteins is the selection of the proteins that are used to generate the building block database. Due to the all vs. all nature of the building block generation it is impractical to extract building blocks from the whole PDB. Therefore one needs to

choose a suitable PDB subset (PDBSS) which should satisfy the following criteria as good as possible:

- as much as possible of the structural diversity of known proteins should be covered.
- to ensure that the conserved portions of each fold can be identified, for each structure it should contain a couple of other structures with a common ancestor.
- sequences should not be highly redundant i.e. have a certain maximum sequence identity.

The current version of the PDBSS contains around 5200 proteins and covers all SCOP superfamilies. The number of proteins per superfamily is limited to 25. The resulting database contains 166 million building block instances, which is sadly too much for some of our algorithms to handle. Thus, the post-processing radically throws away building blocks to reduce the database to a manageable size of 3.34 million building block instances.

2.3 Eliminating redundancy

A lot of the initial effort of this thesis went into reducing redundancy in the dataset. As the focus shifted later on, I will skip most of it here. One concept that remained important is the merger of "building blocks instances" into "building blocks".

So far we only talked about building blocks in terms of an alignment of two structures (i.e. PDB chains). These structures will be called "witnesses" from now on. A building block is defined as a substructure that occurs (with slight differences) in both witnesses. These two occurrences are called "instances". As it turns out, it happens quite a lot that the exact same portions of a structure are alignable with more that one other structure. For example, let's look at the following six building block instances in three different structures:

```
195lA:1b2sE
#1(2,23,2.4,EH) = 52-61:50-59 | 68-80:71-83
1b2sE:1b2uD
#2(2,23,2.1,EH) = 50-59:11-20 | 71-83:33-45
195lA:1b2uD
#3(2,23,2.6,EH) = 52-61:11-20 | 68-80:33-45
```

Again, each line defines two building block instances and one relationship between them. These relationships are visualized in fig. 2.1.

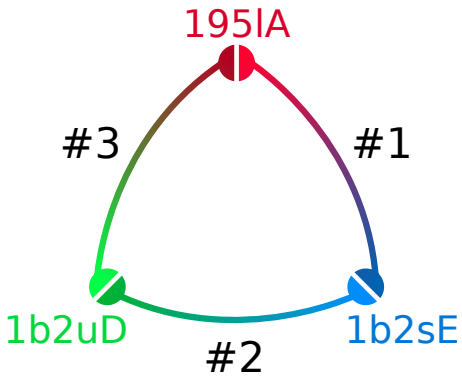


Figure 2.1: Illustration of a building block with three instances

As you can see, these six building block instances are indeed only three *distinct* instances because they have the same residue ranges. Only unique instances are stored to the DB. The alignment of two unique instances is called a "combination".

From a naive point of view, the three combinations would make up three building blocks. But because these three combinations share the same instances it would make more sense to define them as *one* building block. More formally:

$$\forall a, b \in \text{instances} : \ (a, b) \in \text{combinations} \Rightarrow \text{bblock}(a) = \text{bblock}(b)$$

You might wonder why the above is not an equivalence relation: Intuitively, every building block instance is interconnected with every other instance via a combination as in fig. 2.1. However, this is not always the case due to RMSD cut-offs and post-processing. In other words, the instances belonging to the same building block do not necessarily form a clique i.e. there is not necessarily a combination between any two instances. These "sparsely connected building blocks" can arguably be a problem if they contain so many instances that there exist instance-pairs that are only connected through multiple other instances; because then one can't really assume a structural similarity for such instance-pairs any more. However the advantage of this definition is that it is simple and it partitions the building block instances so that each instance belongs to exactly one building block. In section 5.4 I developed a couple of features whose plots can be interpreted as follows: For a database of 1.6 million building blocks, there is no

evidence that sparsely connected building blocks produce inferior matches. For a very large database of 160 million building blocks, the above definition produces building blocks with hundreds of thousands of instances that are very sparsely connected. So without further analysis it can be said that for such a big database, the above definition would have to be revised; but for the databases we are currently working with, we can ignore these concerns, because the number of instances per building block is usually reasonably small.

One mayor difference between instances and building blocks can be easily seen from fig. 2.2: Each instance has exactly one witness and therefore a well-defined sequence. A building block instead has a sequence profile, thus at each site (= residue offset/position) there are multiple possible amino acids; all of them occurring with a certain probability, as calculated from the constituent instances.

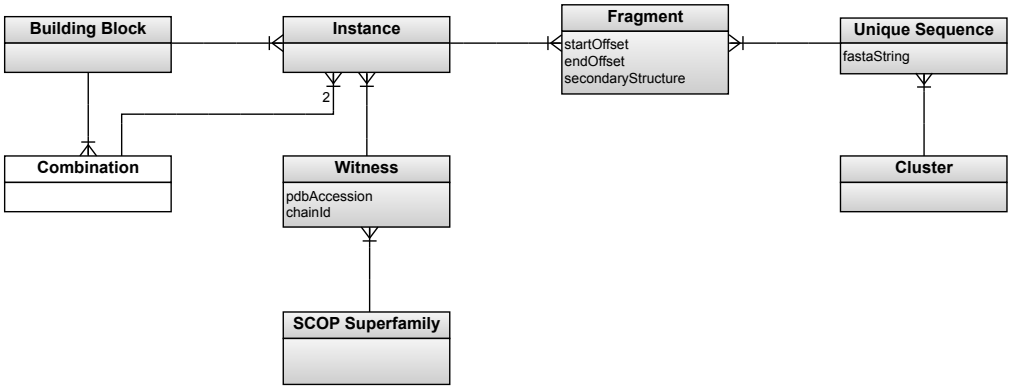


Figure 2.2: Entity relationship diagram of the building block database

3

Excursus: Sequential clustering

Later on – in section 5.11 – a sequential clustering of all building block fragments is needed. This excursus will shed some light on how such a clustering can be obtained and what obstacles have to be overcome.

The choice of algorithms is much restricted by the huge size of the database. In total there are 10 million fragments with 1.3 million distinct sequences. This makes it expensive to even obtain a similarity graph, left alone an optimal clustering.

The sequence similarity graph is an undirected graph with weighted edges, where the mutual similarity is used as the edge weight. After one has obtained the similarity graph, the nodes have to be clustered. A graph $G = (V, E, w)$ is seen as a set of n nodes V, edges E and a weight function $w\colon E \to \mathbb{R}$; an edge is a tuple of nodes. Graph mining algorithms can be categorized whether or not they can honor edge weights; For those that don't, an edge cutoff c must be applied so that $\forall e \in E : \; w(e) \geq c$. Even for algorithms that can work with edge weights, it makes sense to apply an edge cutoff, because firstly it saves storage space and computation time and secondly log-odds similarities can be negative and most algorithms can't handle negative weights. A negative similarity score means that two fragments are less than likely to be evolutionary related.

When comparing fragments one has to decide whether one wants to allow gaps, i.e. if the modeled sequence mutations should include insertions or deletions. Due to the fact that the retrieval module doesn't allow gaps, it seems acceptable if the clustering doesn't allow gaps as well. This makes sequence alignments less complex and the naturally results in disjunct similarity graphs for each fragment length.

3.1 Connected component analysis

To get an impression about the graph, it makes sense to start with a very simple graph mining method called "Connected Component Analysis". A path $p(v_1, v_n)$ between two nodes v_1, v_n is defined as a set of edges so that

$$(v_1, v_2), (v_2, v_3), (v_3, v_4), \ldots, (v_{n-1}, v_n) \in E$$

A connected component $C \subseteq V$ is defined as a subgraph where $\forall\, v_i, v_j \in C\ \exists\, p(v_i, v_j)$. An algorithm that extracts all connected components from a graph is fairly easy to implement and can also be used as a preprocessing step for other algorithms because most graph mining algorithms allow connected components to be processed in parallel.

Due to the way the similarity graph is generated, fragments with different lengths always are in different connected components. Most of these fragment length induced connected components cannot be divided any further. However, a clustering for the intended use case would need a number of clusters that is approximately between $\frac{|V|}{5}$ and $\frac{|V|}{100}$ and cluster sizes should be relatively uniform (i.e. no extremely big clusters, few singletons). Even if one considerably increases the weight cutoff or sparsifies the graph with one of the methods in section 3.5, the resulting connected components tend to be one huge chunk and a few very small chunks; as such they remain unusable as a clustering.

3.2 Conventional clustering algorithms

Most unsupervised learning algorithms employ some kind of distance function that calculates a distance for each pair of nodes. The distance is usually derived from the nodes' features, but for graph mining purposes the distance is given explicitly as the edge weight. I tried different algorithms like hierarchical clustering and k-means, but all of them either returned unusable clusterings or had unacceptable runtime characteristics.

3.3 Dense subgraph mining

The density γ_S of a subgraph $S \subseteq V$ is defined as

$$\gamma_S = \frac{2|E_S|}{|V|(|V|-1)}$$

$$E_S = \{(v_i, v_j) \in E \mid v_i, v_j \in S\}$$

i.e. the actual number of edges within the subgraph in relation to the maximum possible number of edges with the subgraph. A subgraph with $\gamma_S = 1$ is called a complete subgraph a.k.a. a "clique". This definition is for unweighted graphs only, but one can easily extend it for weighted graphs. For example, one could require a cluster to be a clique and have a minimum average edge weight. One way to find a meaningful clustering could be to find a so called "minimum clique cover" i.e. partitioning the graph into cliques so that every node is part of exactly one clique and that the total number of cliques is minimal. As a first step to find the minimum clique cover, one would have to enumerate all maximal cliques, i.e. all – not necessarily disjunct – cliques that cannot be extended. The Bron-Kerbosch algorithm is widely used to solve this sub-problem with a worst-case time complexity of $\mathcal{O}(3^{n/3})$. The problem with minimum clique cover is that one is quite inflexible about the properties of the resulting clustering. For example, a sparse graph would yield too many singleton clusters.

To allow for more flexibility, I experimented with $\gamma_S < 1$. Doing that, I became concerned that this would allow clusters that contain quite unrelated fragments. To counter this problem, I tried to employ a concept called "feature vector network" where the graph is augmented with a feature vector for each node. An algorithm called "CoPaM" [Moser et al., 2009] searches for all maximal subgraphs that not only satisfy density criteria but also have similar feature vectors ("cohesive patterns"). For the feature vectors I used easy to obtain biological properties like secondary structure, solvent accessibility and hydrogen bonds. That way, a maximally extended cohesive pattern would not only have a certain minimum density but its nodes would also share (some of) the biological features. I heavily extended the above mentioned algorithm to make better use of the available resources on the RBO group's computer cluster.

As it turned out, even for a small PDBSS and using 600 CPUs in parallel, the algorithm had worrying runtime and memory requirements. My efforts came to a halt when the current program state exceeded the 64 gigabytes of RAM on the cluster's frontend node. Apart from the size of the dataset in general, the main reason for this problem was apparently a lot of redundant cohesive patterns. An analysis of intermediary results showed that a small clique produced a lot of cohesive patterns that all contained this clique together with another node. These kind of redundancy problems were also the reason for the development of another algorithm called "GAMer" [Günnemann

et al., 2010]. Unfortunately, this algorithm has even worse average runtime and is harder to parallelize. Both algorithms search for an ideal solution, i.e. an enumeration of all maximal cohesive patterns. However, a heuristic solution is sufficient for the intended use case. It might have been possible to develop a heuristic algorithm with acceptable runtime characteristics, but I abandoned further development because of a shift in focus of this thesis in December 2011.

3.4 Spectral clustering

Another way of looking at the graph is to interpret the edge weights as probabilities of a "random walk". The random walk metaphor works like this: an imaginary token is initially placed on any of the nodes. At each iteration, the token is moved to another node that is connected via an edge to the current node. Which node is chosen is determined randomly with probabilities relative to the edge weights. Clusters are then characterized by the token residing significantly more often within a cluster than in other subgraphs of the same size.

So called spectral clustering algorithms try to extract such clusters analytically by finding the eigenvectors of the adjacency matrix A. This (symmetric) matrix can be obtained from the edge weights:

$$A_{ij} = \begin{cases} w(e) & \text{if } e = (v_i, v_j) \in E \\ 0 & \text{else} \end{cases}$$

To represent these types of algorithms, I picked an open source implementation called "SCPS" [Nepusz et al., 2010]. SCPS has been specifically developed for clustering homologous protein sequences. The algorithm first finds the eigenvectors corresponding to the k largest eigenvalues of the adjacency matrix. Then these k n-dimensional vectors are transposed into n k-dimensional vectors. The vectors are then normalized and clustered into k clusters using k-means. The authors have tested their algorithm on a similarity graph of 14,183 protein sequences and benchmarked their clustering against SCOP superfamilies. They have seen improved results compared to Connected Component Analysis, hierarchical clustering and "TribeMCL" (see below).

Applied to the building block fragment database, SCPS had two issues: For one thing the eigenvector calculation takes unacceptably long and for another thing the algorithm

produced a relatively small quantity of huge clusters and the rest were singleton clusters. As k is essentially the only parameter that the algorithm takes, its results were unusable.

The (Tribe-)MCL algorithm uses the same random walk metaphor, but has an entirely different approach. Instead of dealing with eigenvectors, the algorithm simulates the random walk on the adjacency matrix explicitly. Each iteration consist of three steps: (re-)scaling, expansion and inflation. The scaling step turns the matrix into a stochastic matrix, i.e. each column sum up to 1. The expansion step multiplies the matrix with itself and the inflation step sets each entry to the i-th power of itself. i is a parameter that affects the clustering granularity. The algorithm terminates when an equilibrium state is reached. The clusters are then extracted from the resulting matrix by identifying all its connected components.

3.5 Graph preprocessing

As mentioned before, the graph contains regions that are highly connected with large parts of the graph, i.e. degree(S) is $\mathcal{O}(n^2)$; an ideal cluster would have a degree $\mathcal{O}(|S|^2)$. Depending on the algorithm, the impact these regions have on the clustering qualtity differs, but it generally is noticeably negative. Accoring to the MCL manual "the network should not have nodes of very high degree, that is, with exorbitantly many neighbours. Such nodes tend to obscure cluster structure and contribute to coarse clusters" [van Dongen, 2010a]. One way of sparsifing the graph is to find the edges with the t highest weights for each node (equivalent to finding the t nearest neighbours); then, one only keeps edges that are within this threshold for *both* adjoining nodes. "A good heuristic is to choose a value [for t] that does not significantly change the number of singletons in the input graph" [van Dongen, 2010b]. For the building block fragments I used $t = 10$.

4

Building block alignments

This chapter covers how building blocks alignments on a target sequence can look like and generally lays out the ground work for chapter 5.

4.1 Nomenclature

Before we dive into the specifics of building block alignments, let's be totally clear about how these alignments shall be classified. To be precise, the building block project contains two separate classification experiments: The first is done by the retrieval module and the second is the subject of this thesis. Both share the same class definition: The positive class contains building blocks whose structure matches the structure of the target protein (within a certain RMSD threshold) and the negative class contains building blocks that don't. However, the two experiments do not share the same ground truth: For the retrieval experiment the ground truth is the whole building block database correctly classified into the two classes. For this thesis the ground truth is only the *retrieved subset* correctly classified. Therefore the meaning of the terms "true positive", "false positive", "true negative" and "false negative" is different for both experiments. This thesis works only on the building blocks that were labeled as positives by the retrieval module, called "matches". In the following, true positives from the retrieval perspective are called "correct matches" and the false positives from the retrieval perspective are called "incorrect matches". The goal here is to label the matches as "correct" or "incorrect" with a high accuracy; therefore the terms true positive etc. will from now on only refer to the classification experiment from this thesis'

perspective. Figure 4.1 shows a few examples of what building block matches can look like (one match per row). Each feature has to be calculated once per building

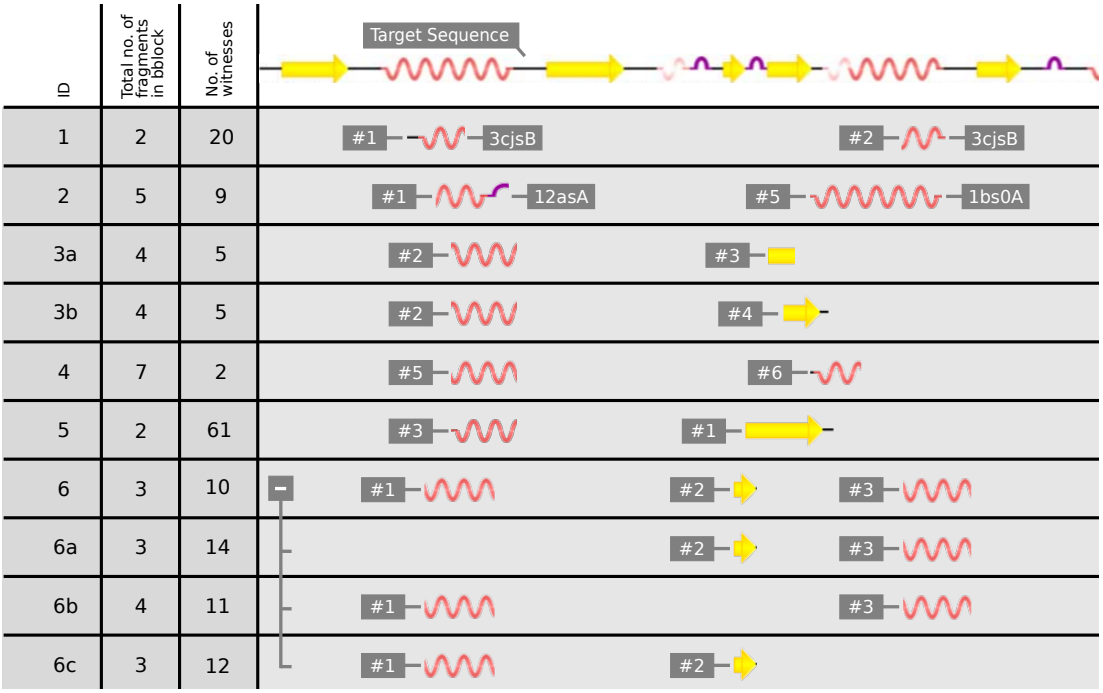


Figure 4.1: Diagram showing the diversity of building block matches. Target sequence on top (not to scale).

block match. Then, using these features, the matches have to be classified as correct or incorrect. The diagram is supposed to help in understanding what exactly a building block match is and what kind of variety with respect to building blocks, witnesses and sequence clusters has to be taken into account. Understanding this is a prerequisite to understanding the features discussed later. Please compare the following statements with the diagram:

- A building block match consists of two or more fragment matches. For example, ID5 has two fragments aligned and ID6 has three fragments aligned. Accordingly, matches are called “two-fragment matches”, “three-fragment matches” etc.

- Individual fragment alignments are not necessarily compatible with each other. For example, ID3 has #3 and #4 align at overlapping portions of the target sequence. In this case, #2, #3 and #4 cannot be called a three-fragment match,

but rather two separate two-fragment matches. Similarly if a particular fragment # aligns at more than one spot of the target sequence (not shown in diagram).

- More-than-two fragment matches can be combinatorially disassembled into two-fragment matches, see ID6.

- Fragments of a match do not necessarily come from the same witness, for example fragment ID1 #1 and #2 both originate from "3cjsB", but ID2 #1 and #5 come from two different witnesses.

- Fragments that align at the exact same residue range of the target sequence (e.g. ID3 #2, ID4 #5, ID5 #2) are ideally in the same sequence cluster (see ID3 #2 and ID4 #5). But due to aliasing in the sequence clustering and the much more sophisticated way that HHsearch aligns the fragments, this is not always the case (see ID5 #2).

- Some fragments of a building block can remain unaligned, i.e. the fragments in a building block match can be a subset of the fragments in the originating building block. For example, ID1 has 2 of 2 total fragments aligned; ID2 has 2 of 5 total fragments aligned.

- Even though each building block is unique, a subset of its fragments can often be found in multiple other building blocks. E.g. building block B_1 consists of fragments $\{F_1, F_2, F_3\}$ and building block B_2 consists of fragments $\{F_2, F_3, F_4\}$. In consequence, if fragments F_2 and F_3 were to align (compatibly) on a target sequence, this would produce two matches. In practice, only one of these matches is flagged "unique" and all the corresponding building blocks are called "equivalent". The "total no. of fragments in bblock" and "no. of witnesses" is then defined as the maximum across all equivalent building blocks. See how ID6a/b/c have different values for these properties.

4.1.1 Formal definitions

In order to hopefully be able to define features more clearly in the next chapter, I use the following formal definitions (see also fig. 2.2):

- A building block BB is as set of building block instances BBI and a set of fragment indexes $F = 1, \ldots, f$ (each BBI has the same number of fragments f).

- A building block instance BBI is defined by its witness w and two functions o_b and o_e mapping the fragment indexes to their begin and end residue numbers on the witness' sequence respectively.

- A two-fragment building block match BBM is defined by the building block BB, the building block instance of the first fragment BBI_1, the building block instance of the second fragment BBI_2, the fragment indexes of the two matched fragments $M = \{i_1, i_2\} \subseteq F$ and two functions m_b and m_e mapping the fragment indexes to their begin and end residue numbers on the target's sequence respectively. Note that HHsearch is able to align sub-fragments. Therefore $m_e(i) - m_b(i)$ isn't necessarily the same as $o_e(i) - o_b(i)$. In order to be able to determine the actual start and end residues of the *sub*-fragment in the witness sequence, the function s maps the fragment index to the internal offset in the fragment; a fully matched fragment has $s(i) = 0$. The actual begin offset in the witness sequence is then $o_b(i) + s(i)$ and the end index is $o_b(i) + s(i) + m_e(i) - m_b(i)$.

- *eq* is a relation that takes a BB and an M as an argument and returns tuples of BB's and M's that are equivalent (including the arguments). Note the many features employ this relation but sometimes it will be omitted in feature definitions for the sake of readability.

4.2 What makes a good feature?

A posteriori, a good feature is a feature the results in a good prediction/classification performance on a representative test set, see section 6.2. But a priori this information is not available. Because there are sheer endless possibilities for features, it makes sense to talk a little bit about what kind of features we are looking for.

In general, we are either looking for features that turn HHsearch's scoring output into features that are applicable for two-fragment building block matches (see section 5.1) or features that model an aspect that is neglected by HHsearch. Here are some thoughts about which features might be preferable to others:

building block rationale Features that are somehow related to the building block rationale (see section 1.3) are preferable to those that are not. For example features that could indicate fragment co-evolution or inter-fragment contacts.

target-dependence As we will later see, some features are only dependent on the building block (or its subset of matched fragments), but not on the target protein. While such features can make sense as a general ability of a building block to be a correct match, features that depend on both the building block and the target protein are probably superior because they are more specific. As a matter of fact, during the CASP experiments, only the sequence of the target protein is known, so features ideally employ the target's sequence somehow.

resolution In general, features that may have lots of different values are preferred to features that have only a few possible values (provided they have similar levels of noise). This is because one can always discretize/classify a value, but increasing the resolution is impossible. Features that rely on counting certain occurrences usually have finite positive integer values. Because the size of the PDBSS is only a four-figure number and the amino acids per chain are usually a small three-figure number, features often have coarse resolutions. That is, they often revolve around small one-figures.

missing values A related problem is that some features cannot be calculated for all building block matches. This makes them less useful for classification.

redundancy Features that are correlated with each other contain redundant information when used together. Apart from the fact that such features add a lot less value than non-correlated features, model quality could be negatively influenced by such features [Hall [1999]; c.f. section 6.2]. Unfortunately, because many features rely on the same biological aspects, many of them are correlated. Also, the fragment lengths are a common influence on most features; some are correlated to the length of the smaller fragment, some to the size of the bigger fragment and some to the added length.

Due to the fact that building blocks can contain an arbitrary number of fragments (see fig. 4.2) and each target structure could match a subset of these, there is a great level of complexity when defining features for every possible kind of match. If you

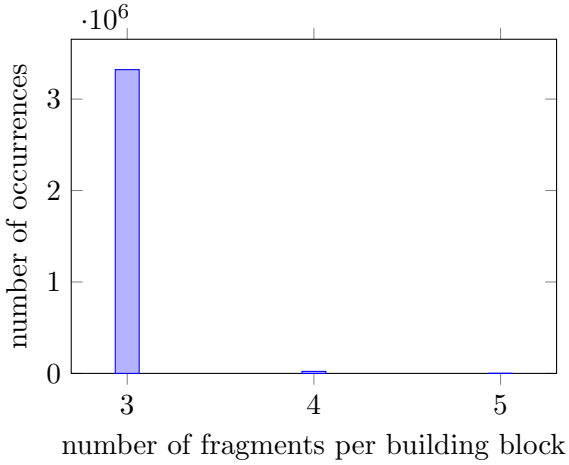


Figure 4.2: Histogram of fragments per building block, i.e. viewing a building block as a set of fragments, this graph shows the sizes of these sets for all building blocks in the database. Note that building blocks with a fragment count of 2 are currently filtered from the building block database (c.f. section 2.2).

look at fig. 4.3, you can see that most building block matches contain two fragments. There are multiple reasons why two-fragment alignments are most common: For once, the number of matched fragments per building block is limited by the total number of fragments that this building block has. Also, the higher the number of total fragments, the lower the probability of a building block being retrieved as a whole; this could be due to some fragments actually not being part of any evolutionary conserved region (i.e. noise in the database) or the fragments not having co-evolved for the particular target. Even if the complete building block is in the retrieval module's ground truth, the fragments' sequences could vary in retrievability so that some can't be retrieved with the chosen threshold. Then finally, for relatively short targets, the number of non-mutually-exclusive matching fragments is upper bounded. The problem here is that features that are calculated for matches with differing fragment counts can't be offhandedly compared. Even for two-fragment building blocks it is hard enough to obtain a sufficiently diverse training set; and it's even harder to do this for building blocks with more that two fragments. Furthermore, defining features gets quite difficult if one has to take into account an arbitrary fragment count and some of the features described later can't even possibly be calculated for other than two fragments. However, any match with more than two fragments can be decomposed combinatorically into multiple

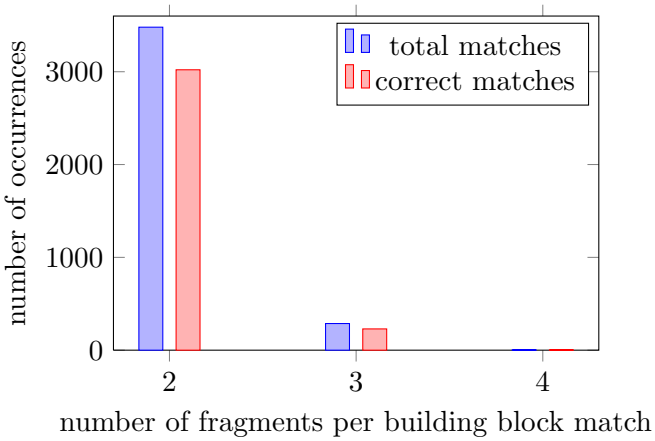


Figure 4.3: Histogram of fragments per building block match, i.e. viewing a building block as a set of fragments, this graph shows the sizes of the building block subsets that matched on one of the CASP9 target proteins; sizes 0 and 1 omitted

two-fragment matches (see fig. 4.1 ID6a/b/c). After classifying all of these two-fragment combinations independently into correct and false matches, one can reassemble the ones that were classified as correct. Due to these considerations I will define and evaluate the features below only for two-fragment matches; matches with more fragments will be decomposed and analyzed independently.

When examining features visually, one usually looks for an obvious linear relation, i.e. values scattered around a diagonal line with the x-axis being the RMSD from native in this case. In lieu of such a relation, one searches for a cutoff that separates both classes with reasonable accuracy. Either of these kinds of relations are quite rare for biological problems, though. Often one encounters features where most samples are situated in one region of the plot. Thus, the fact that an example is placed in that region carries almost no information gain. But there are some outliers that are mostly part of one class (usually the negative class), so by applying a rigid cutoff one can at least classify a small number of examples with relatively high confidence.

For non-visual ways of examining a feature's usefulness see section 6.2.

4.3 Fragment co-evolution

Many of the following thoughts are based on the hypothesis that – in terms of probability theory – the correctness of one building block fragment is not independent from the

correctness of another fragment from the same building block. Figure 4.4 shows the five events that are necessary for a two-fragment building block to be correct. Of course, for

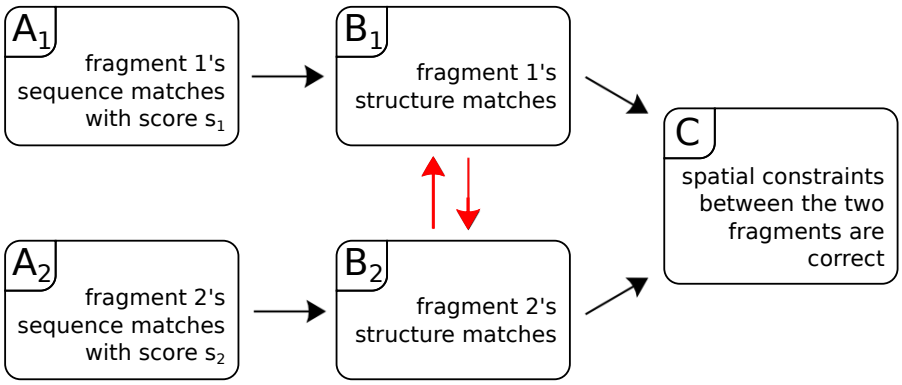


Figure 4.4: Definition of various events (and their dependencies) that can occur for a building block/target combination. Co-evolution shown in red

this thesis, we only have to deal with cases where A_1 and A_2 are given. Note that A_i and B_i are required to occur at the same position.

What we are interested in is the probability $P(B_1 \cap B_2 \cap C \mid A_1 \cap A_2)$. Now, let's say that A_1, A_2 and B_1 are true, then we know that there is very likely a homology between the target and the building block's witnesses. Knowing that these proteins are evolutionary related increases the probability of more homologous fragments existing between these proteins. Therefore one could say that:

Lemma 4.1

$$P(B_1 \cap B_2 \mid A_1 \cap A_2) > P(B_1 \mid A_1) \cdot P(B_2 \mid A_2)$$

I tested this hypothesis using individual fragment alignments on CASP9 targets. Therefor I divided fragment alignments into two classes: The first class were matches where there existed another fragment alignment from the same building block on the same target. The second class was made up of the remaining "orphan" matches. Each class had more than 5000 samples and I made sure that the E-value distribution for each target was very similar in both classes. The orphan class had an average RMSD (each fragment was compared individually to the matching region of the target protein's native structure) of 3.15 Å while the non-orphan class had an average of 2.88 Å. Figure 4.5 shows that

non-orphans perform better across the whole RMSD value range, i.e. for low RMSDs there are more non-orphan fragments than orphan fragments; at the same time, matches

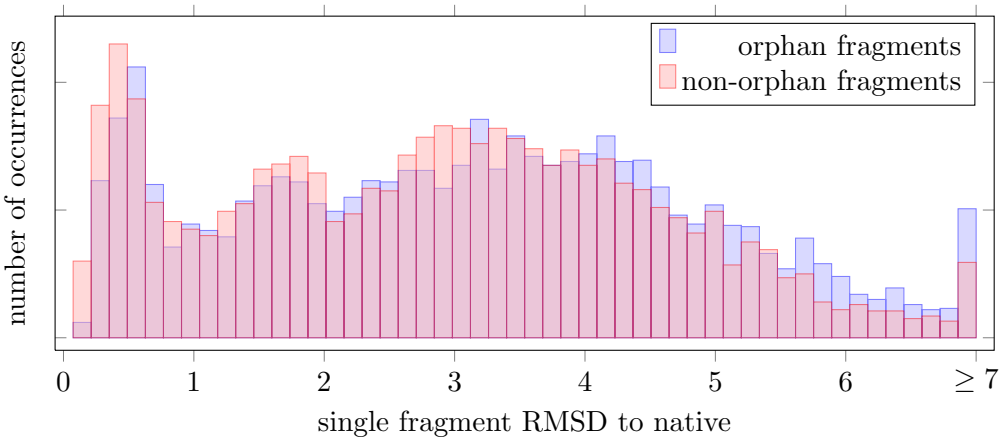


Figure 4.5: Histogram of RMSDs (compared to native) of orphan fragments vs. non-orphan fragments.

with an RMSD of 4 or more are consistently occurring more often for the orphan class. This supports the claim made in lemma 4.1.

4.4 Statistical significance of RMSD

For testing purposes we use sequences where the native structure has already been experimentally determined. Of course this usually only makes sense if the target structure was not part of the PDBSS. In terms of this thesis I mostly tested with the ≈ 120 targets of the previous CASP run (CASP9). While these targets have the disadvantage of having no aspiration to be representative for the protein fold space, they have the advantage of being closest to a realistic trial run for the upcoming CASP.

Because the native structures from previous CASP runs are available, we asses the quality of a building block match by comparing it to the target's native structure. Using HHsearch's alignment output, one can create a bijection between the backbone atoms (N, C_α, C, O) of the relevant protein regions of both the target and the witness. Using this bijection, an optimal superimposition is calculated by PyMOL ("pair fit"). The superposition algorithm ignores all non-mapped residues but keeps the intra-protein distances between fragments rigid.

There are cases where the matched fragments come from different witnesses. Unfortunately one cannot reasonably use a combination of both witnesses to extract constraints from because the resulting constraints would not necessarily be native-like. Therefore, one has to decide to use only one of the building block's witnesses as a "representative".

The rest of this section did not have an impact on the final classifier (see section 7.3), but it is still mentioned here because it is an interesting piece of information about the characteristics of building blocks: As mentioned in section 1.6, Gao et al. [2009] found that the length of a fragment correlates with the RMSD of a single-fragment superposition. This correlation might cause a classifier/predictor to bias towards short fragments. However, a long fragment with an RMSD of 2.5 Å might be more valuable for protein structure prediction than a very short fragment with an RMSD of 1.5 Å. To counter this, Gao et al. [2009] introduced a measure of statistical significance. First, they superimposed 10000 random (gapless) pairs of fragments for each fragment length. The statistical significance of an RMSD is then defined as the probability of observing a worse RMSD for a random pair of fragments of the same length. This concept can be easily adapted to two-fragment building blocks by extracting appropriate building block subsets for each combination of lengths l_1 and l_2; then superimposing these 10000 times to random regions of another protein.

$$\mathrm{StatSig}(l_1, l_2, r) = \frac{\#\text{ random superimpositions of length } l_1, l_2 \text{ with RMSD} \geq r}{10000}$$

The average combined length of a two-fragment building block match is around 19.5 residues; the random superimposition data suggests that a StatSig of 99.9 % corresponds roughly to an RMSD of 2.7 Å for such matches. Figure 4.6 shows that for two 6-residue long fragments, the RMSD needs to be about 1.9 Å to reach the same level of statistical significance. However, as we will see in section 7.2, none of the fragment length based features (see section 5.3) were among the top features. The reason for this might be that even though there is are relationship between fragment length and RMSD, the prevailing factor that determines whether a building block match superimposes with a low or a high RMSD is whether or not the spatial constraints between the two fragments are correct. Due to the bad performance of fragment length based features, I did not investigate whether it might be better to classify by statistical significance instead of RMSD.

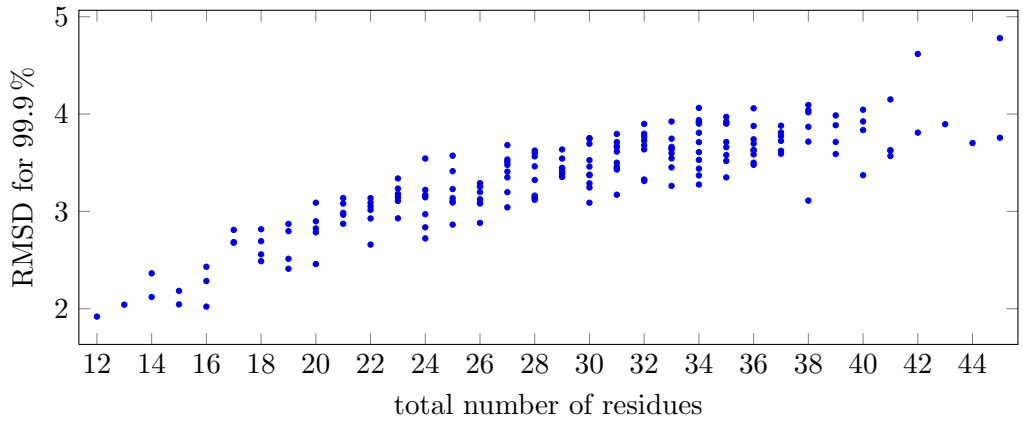


Figure 4.6: RMSD needed for a statistical significance of 99.9 %. The abscissa shows the combined number of residues of two fragments. Each point represents the RMSD necessary given a pair of two fragment lengths. For each abscissa, there can be multiple ordinates because there are multiple ways that two fragment lengths can result in the same total length.

5

Features

This chapter describes the various features that were calculated for each two-fragment building block match. In total, I calculated ≈ 140 different feature variations. Only the most interesting and representative variations are described here and only a fraction of these features are plotted. Which of the features are plotted depends on whether or not they are selected by the feature selection (see section 6.2) or I consider them to be otherwise interesting. Each feature has a unique identifier e.g. seq_evalues_min which is used throughout this thesis (feature descriptions, plot axes, results) and the implementing code as well. This way, this chapter also functions as a documentation for part of the code.

All feature plots have the RMSD between the building block matches and the target's native structure (see section 4.4) as their x-axis. Due to the large number of plots, the individual plots are not interpreted unless they show counter-intuitive data (namely fig. 5.6 and fig. 5.33). Using the thoughts in section 4.2, it is often apparent whether one particular feature is useful or not; section 7.2 will list the best performing features according to objective criteria.

5.1 Sequence similarity based features

As already mentioned in section 1.5, HHsearch provides a score together with every alignment it finds. It appears that this score is quite noisy – especially for values below a certain threshold, see fig. 1.3.

Apart from the "(raw) score", the HHsearch output also contains a "Probability" and an "E-value" (see section 1.6) for each match. According to the documentation, the "Probability" also takes into account how well the secondary structure of the template and the predicted secondary structure of the target match; making it supposedly more sensitive than the E-value. In terms of fig. 4.4 one could say that "E-value" tries to be proportional to $P(A_i)$ and "Probability" tries to model $P(B_i \mid A_i)$. When ordering all aligned fragments by "E-value", the "Rank" of an alignment is the position of the alignment in the ordered list. Considering only building block matches from one target, the discriminative power of "Rank" and "E-value" is essentially the same. But when looking at building block matches from different targets, "Rank" states how good an alignment is relative to other alignments on the same target.

In order to turn these fragment-wise features into features for two-fragment building block matches, I generated the following features.

seq_score_min $min_{i \in M}\text{score}(i)$ → fig. 1.3

seq_score_max $max_{i \in M}\text{score}(i)$

seq_score_sum $sum_{i \in M}\text{score}(i)$, i.e. the score that a concatenation of both fragments would theoretically achieve (provided that the matched target regions were concatenated as well).

seq_probab_min $min_{i \in M}\text{probability}(i)$ → fig. 5.1

seq_probab_max $max_{i \in M}\text{probability}(i)$ → fig. 5.2

seq_probab_prod $prod_{i \in M}\text{probability}(i)$, i.e. joint probability of independent events.

is_50_30 Söding [2006] recommend that fragments with a "Probability" greater than 50 % (or greater than 30 % if they are ranked among the best three matches) can quite confidently considered to be correct. It also states that a 95 % alignment can almost certainly be considered to be correct. This feature models whether one or both of the fragments fulfills these criteria. → fig. 5.3

seq_evalues_min $min_{i \in M}\text{evalue}(i)$ → fig. 5.4

seq_evalues_max $max_{i \in M}\text{evalue}(i)$ → fig. 5.5

seq_rank_min $min_{i \in M} \text{rank}(i)$

seq_rank_max $max_{i \in M} \text{rank}(i)$

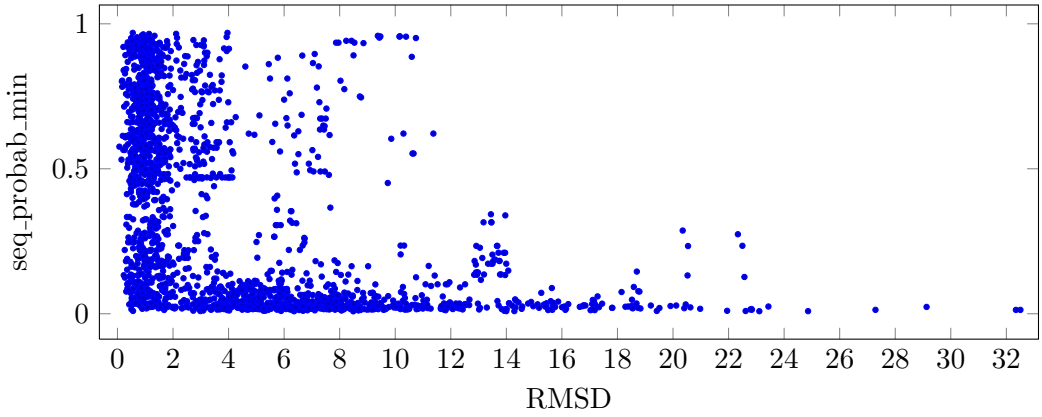


Figure 5.1: Feature seq_probab_min; see section 5.1

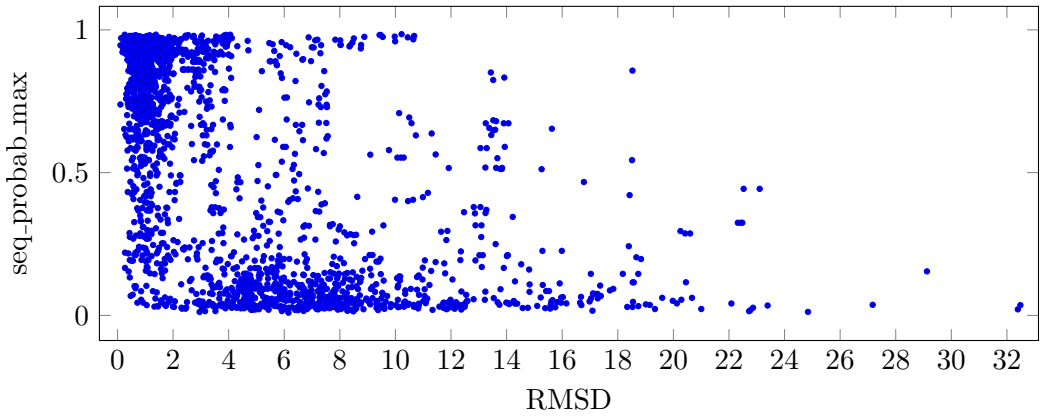


Figure 5.2: Feature seq_probab_max; see section 5.1

5.2 Sequence profile alignment feature

One of HHsearch's main "selling points" is its possibility of profile-profile alignments. That means that both the target and the templates can be sequence *profiles* instead of sequences. A sequence profile is a $n \times 20$ stochastic matrix, i.e. for each of the n positions in the sequence, it contains probabilities that represent how likely a particular amino acid occurs at this position. HHsearch has an elaborate way of bootstrapping

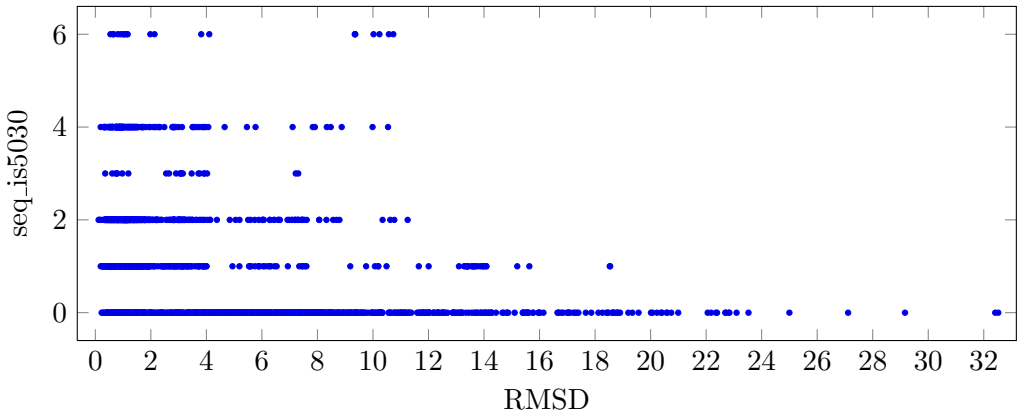


Figure 5.3: Feature seq_is5030 (ordinal feature); see section 5.1

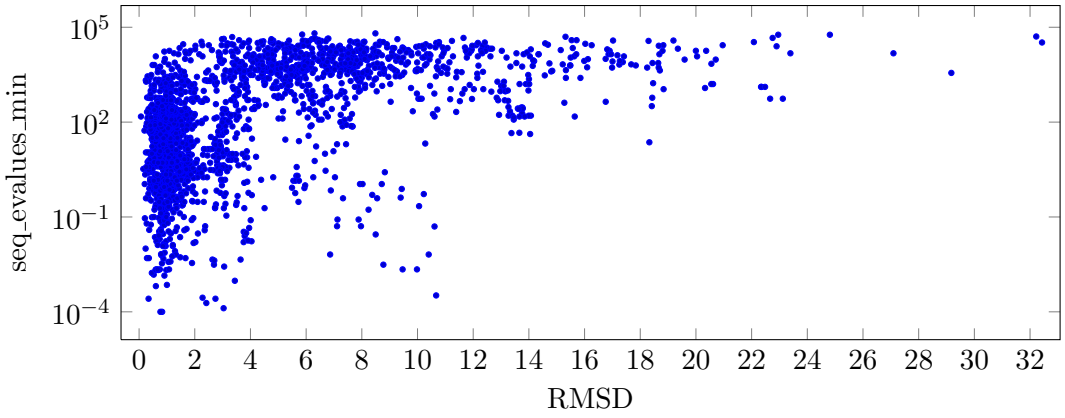


Figure 5.4: Feature seq_evalues_min; see section 5.1

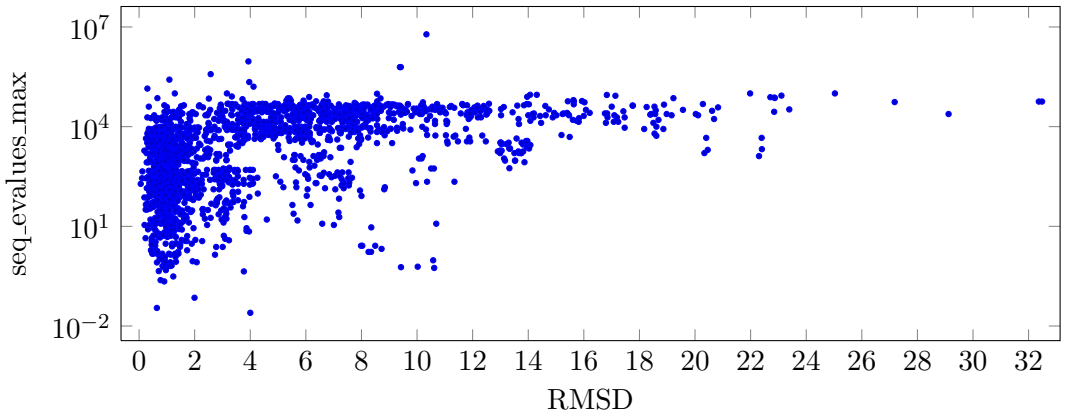


Figure 5.5: Feature seq_evalues_max; see section 5.1

these profiles: Simply put, given a sequence(-profile) it iteratively finds homologs and merges these homologs with the profile until the profile converges. As a result, the profile should reflect the sequence variability of all known homologs.

A building block can have a different sequence for each witnessing protein. For homologous building block instances, these sequences are expected to be similar. However, building block instances are not necessarily homologous; they can also just be analogous due to convergent evolution. Under the assumption that most building blocks consist of homologous instances only, it seems reasonable to start the above-mentioned bootstrapping process with a sequence profile defined by the instances. However, it turned out that retrieval performed better using profiles obtained for each witness sequence independently. As I was told, the reason seems to be that many building blocks have such unrelated sequences that the profiles become too unspecific to match anywhere. Apparently, the amount of analogy (i.e. convergent evolution) in the building block database is quite high. Details can be found in Stefan Dörrs' thesis.

While I do not doubt these results and their consequences, I think there is still a chance to employ sequence profile alignments for scoring. This way, a diverse sequence profile will not hinder a fragment from being retrieved, but those with a conclusive sequence profile might get a slight advantage.

Implementation is straightforward: given all instances of a building block, the sequence profile matrix P_i for each fragment can be built. Each row corresponds to one of the 20 amino acids and each column corresponds to one amino acid position. Let A be the set of all 20 amino acids. Let $p(i, j, k) : M \times A \times \mathbb{N} \to \mathbb{R}$ be a function that returns the jth row and kth column of P_i, i.e. the probability of amino acid j occurring at the kth position of fragment i. Let $s(a, b) : A \times A \to \mathbb{Z}$ be the log-odds that amino acid a gets replaced by b through mutation (according to the PAM30 substitution matrix). Let $t(k) : \mathbb{N} \to A$ be the amino acid of the target sequence at position k. Then, **profile_alignment_score** is $\sum_{i \in M} \sum_{k=m_b(i)}^{m_e(i)} \sum_{j \in A} p(i, j, k + s(i) - m_b(i)) \cdot s(j, t(k))$. All these products are summed up to the final score which is plotted in fig. 5.6.

5.3 Miscellaneous features

There are a number of miscellaneous, simple features that do not qualify for a whole section dedicated to them.

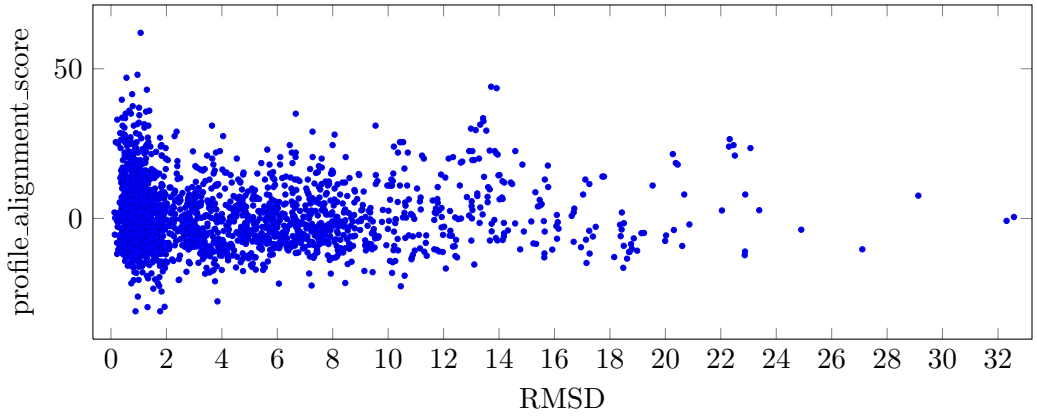


Figure 5.6: Feature profile_alignment_score. Interestingly, this plot shows that both high and low scores are correlated with low RMSDs. As discussed in section 5.2, low scores can reasonably occur even for low RMSDs. However, it is unclear why they seem to occur less often for high RMSDs.

frags_hit As explained in section 4.2, more-than-two-fragment matches can be disassembled into multiple two-fragment matches. This results in a child-parent relationship for such matches. This feature is defined as the maximum number of fragments across all parents. If the match has no parents, the value is set to 2. The idea behind this feature is that an x-fragment match is probably a stronger pointer towards co-evolution than an $x-1$-fragment match (c.f. lemma 4.1 for $x = 2$). In consequence, a two-fragment match that is a child of a three-fragment match might be superior to a two-fragment match that has no parent. On the other hand, with each additional fragment, there comes the possibility of a chance-alignment. If one of these aspects outweighs the other, this feature could work.

frags_total The maximum number of fragments across all equivalent database building blocks, i.e. $\max_{\{eqBB,eqM\}\in \mathrm{eq}(BBM)} \mathrm{f}(eqBB)$

length_match_ratio As already mentioned, matches can consist of fragments that are only *part* of the original building blocks' fragment. The boundaries of a building block are defined through what is considered to be the conserved part of a structure; if only a small part of a fragment is aligned, this would mean that either the match is incorrect or the conserved core of the concerned structures is much smaller than expected. This feature is defined as $\min_{i\in M} \frac{a_i}{l_i}$ with $a_i = m_e(i) - m_a(i) + 1$ being

the length of the aligned part of the fragment and $l_i = o_e(i) - o_a(i) + 1$ the length of the of the full fragment as stored in the building block database. → fig. 5.8

length_min $\min_{i \in M} a_i$ (this can be seen as a limiting factor for the number of contacts between the fragments c.f. section 5.8)

length_max $\max_{i \in M} a_i$

length_total $\sum_{i \in M} a_i$ (see section 4.4.)

no_of_witnesses The number of distinct witnesses where a building block match (or its equivalents) can be found, i.e. given a relation w that returns the set of witnesses for a building block, this feature is $\left| \bigcup_{\{eqBB, eqM\} \in \mathrm{eq}(BB, M)} \mathrm{w}(eqBB) \right|$. → fig. 5.9

equivgroups_pdbs Similar to no_of_witnesses, but instead of counting distinct building blocks, summing them up, i.e. $\sum_{\{eqBB,\, eqM\} \in \mathrm{eq}(BB,\, M)} |\mathrm{w}(eqBB)|$. → fig. 5.10

hbonds_min DSSP [Kabsch and Sander, 1983] can calculate the electrostatic hydrogen bond energy for each residue of a known protein *structure*. This feature is $\min_{i \in M} \sum_{r=o_b(i)}^{o_e(i)} h(r)$ with $h(r)$ being the hydrogen bond energy at residue r[1].

hbonds_max $\max_{i \in M} \sum_{r=o_b(i)}^{o_e(i)} h(r)$ → fig. 5.11

hbonds_sum $\sum_{i \in M} \sum_{r=o_b(i)}^{o_e(i)} h(r)$

hbonds_avg hbonds_sum divided by length_total. High (absolute) hbonds values can mean a highly conserved secondary structure.

is_largest_local This feature counts the number of fragments that belong to a building block instance that is the largest instance for any *pair* of witnesses. Our most radically filtered version of the building block database only contains building blocks where this feature is "1". It's not clear why, but building blocks retrieved from this database had quite high precision (but low coverage). Maybe such building blocks have a lower probability of being chance alignments. → fig. 5.12

frags_reverse Fragments are aligned along the target sequence in a certain order, i.e. if $m_b(i_1) < m_b(i_2)$ one can say that fragment i_1 is aligned *before* fragment

[1] DSSP actually calculates four separate values; $h(r)$ is the sum of all of them.

i_2. Fragments also occur in a certain order in the witness' sequence, i.e. if $o_b(i_1) < o_b(i_2)$ one can say that fragment i_1 is originally before fragment i_2. Due to what has to be considered a bug in the current building block database, the fragment order is the same across all witnesses and is conveniently given by the fragment indexes, i.e. $i_1 < i_2 \Leftrightarrow o_b(i_1) < o_b(i_2)$ This feature compares the fragment order of the alignment with the original fragment order; it is "1" if $i_1 < i_2 \Rightarrow m_b(i_1) < m_b(i_2)$, "0" otherwise. For example, the feature would be "1" for ID2 and "0" for ID5 in fig. 4.1. Figure 5.7 shows that a reverse sequential ordering can easily be accommodated in structure. However, this feature is

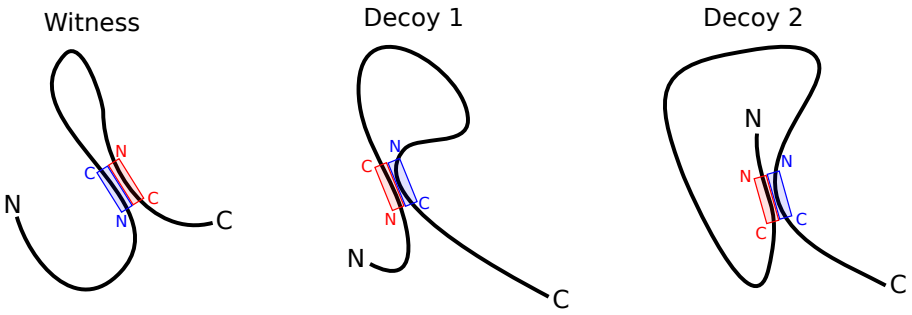


Figure 5.7: Illustration of reverse sequential order alignments. The structure of "Decoy 2" is probably incompatible with the building block match because the fragments lie next to each other in reverse *structural* direction. Note that such a match can also occur for non-reverse sequential order alignments.

based on the assumption that a reverse sequential ordering is unlikely to happen through evolution because it can't happen through point mutations. Instead, the sequence must be cut somewhere between the two fragments and the two snippets must be re-joined in reverse order. It has been found that such mutations do in fact happen in nature; they are called "circular permutations" [Cunningham et al., 1979; Bliven and Prlić, 2012]. However they are relatively rare and it is questionable if building-block-style local sequence-structure relationships would be preserved between portions of the sequence that are split up by such permutations. → fig. 5.13

frags_adjoining This feature is "1" if $|i_1 - i_2| = 1$, otherwise "0". That is, if the two aligned fragments had no other fragments between them in the witness sequence.

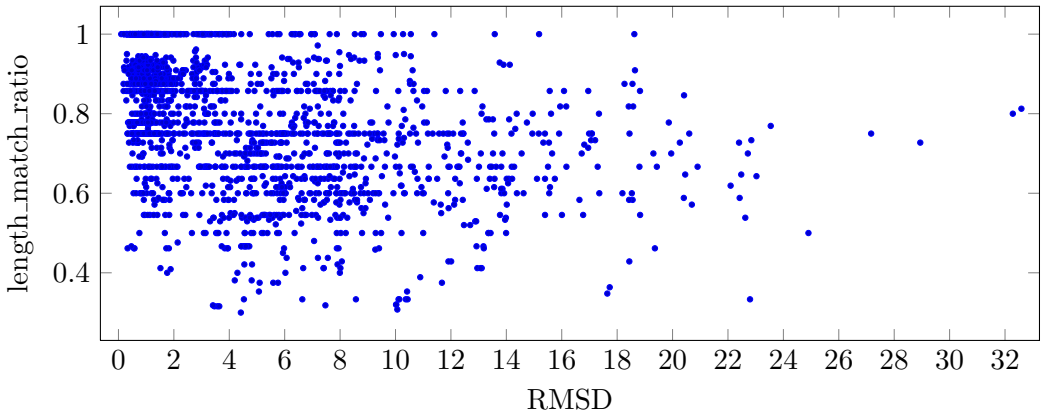


Figure 5.8: Feature length_match_ratio; see section 5.3

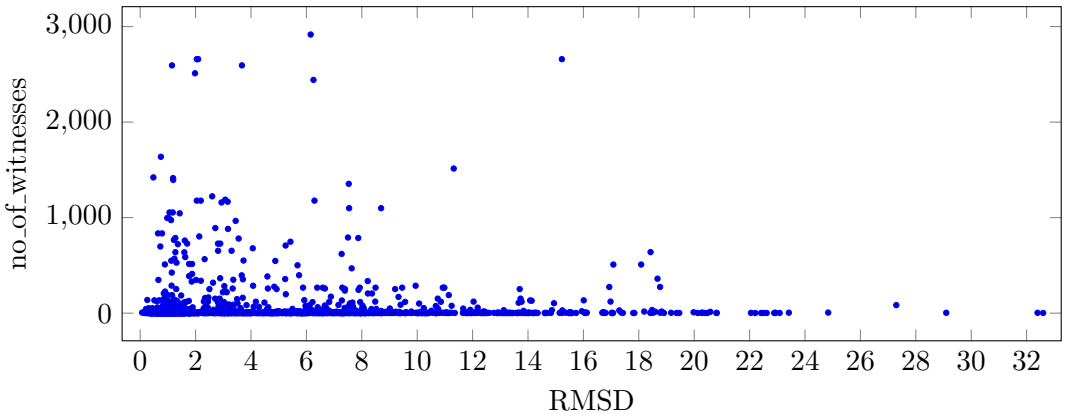


Figure 5.9: Feature no_of_witnesses; see section 5.3

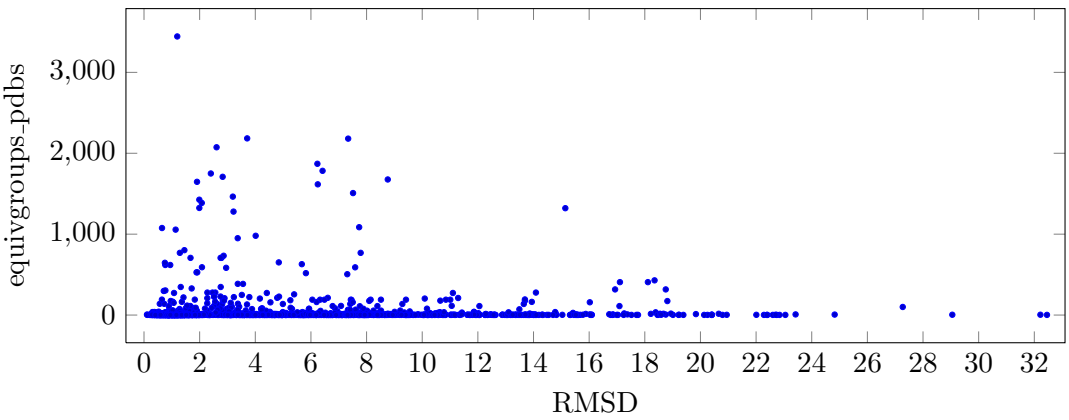


Figure 5.10: Feature equivgroups_pdbs; see section 5.3

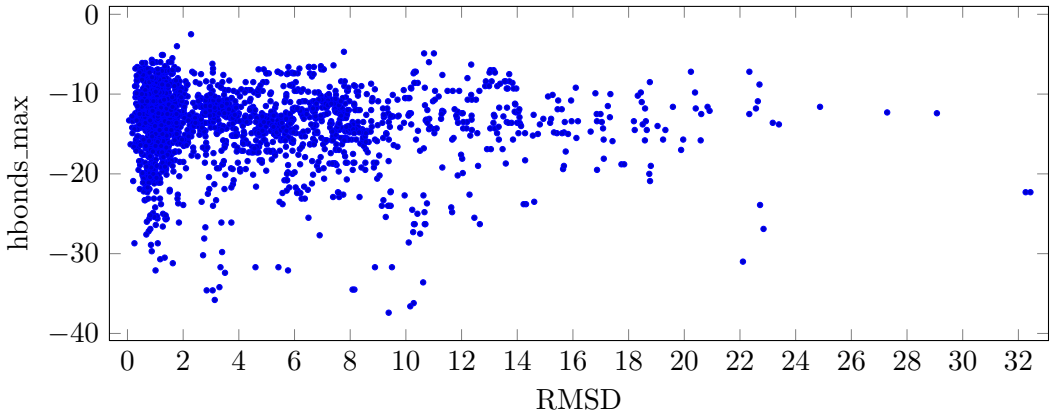


Figure 5.11: Feature hbonds_max; see section 5.3

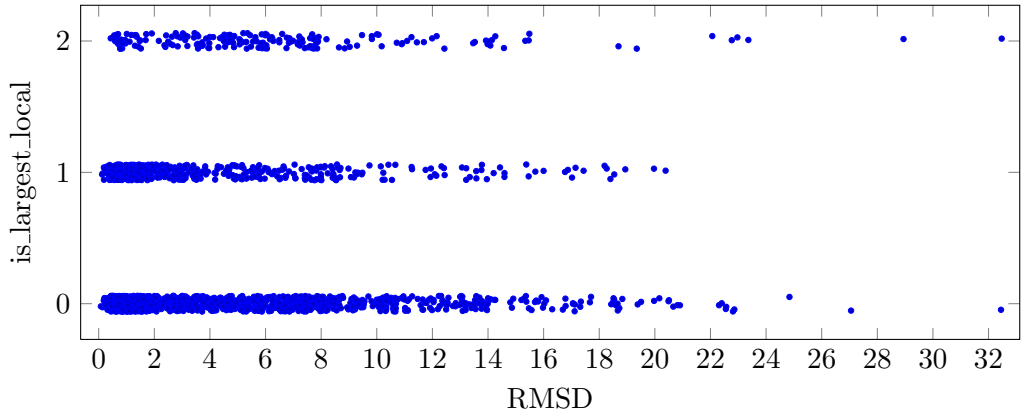


Figure 5.12: Feature is_largest_local; see section 5.3

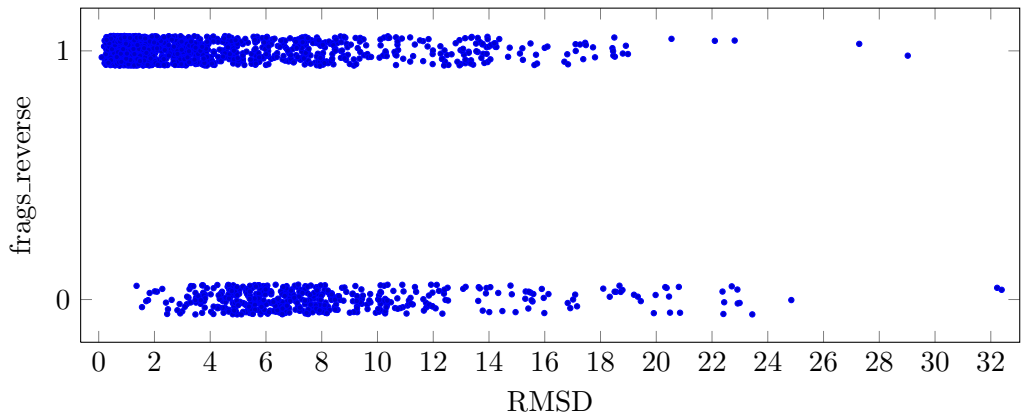


Figure 5.13: Feature frags_reverse; see section 5.3

5.4 Witness density based features

As mentioned before, the fragments of a building block match do not necessarily have to come from the same witness/instance (see ID2 in fig. 4.1). In cases where the fragments come from different witnesses, it might be interesting to see whether the RMSD between the building block instances can be used to predict the RMSD to the target structure.

Using the nomenclature introduced in chapter 3, the relationship between the instances of one building block can be interpreted as a weighted graph $G = (V, E, d)$ where there exists an edge between two building block instances if the two instances are superimposable with an RMSD below the threshold that was used to generate the building block database (c.f. section 2.3). The graph is a connected component, but not necessarily a clique. That means the fragments of a building block match do not even necessarily have an edge between their instances. This might cause concerns whether matches like these are inferior to matches where both fragments come from the same instance or where both instances are connected via an edge. So, using the information about instance relationships might not only yield valuable features but also (in-)validate decisions that were made earlier in section 2.3.

witness_same_instance This feature is "1" if both fragments come from the same building block instance, "0" otherwise. → fig. 5.14

witness_exists_combination This feature is "1" if both fragments come from the same building block instance or $(BBI_1, BBI_2) \in E$, "0" otherwise. → fig. 5.15

witness_combinations This feature is the total number of edges $|E|$ in G.

witness_density This feature is the number of actual edges divided by the maximum possible number of edges for the given number of instances: $\frac{2|E|}{|V|(|V|-1)}$. → fig. 5.16

witness_rmsd_min $\min_{e \in E} d(e)$.

5.5 Sequence separation based features

Each two-fragment match has a "gap" in between, i.e. a number of residues that are not part of the match. For the target sequence, this is a single non-negative integer value and for the building block instances it is a distribution of positive integer values.

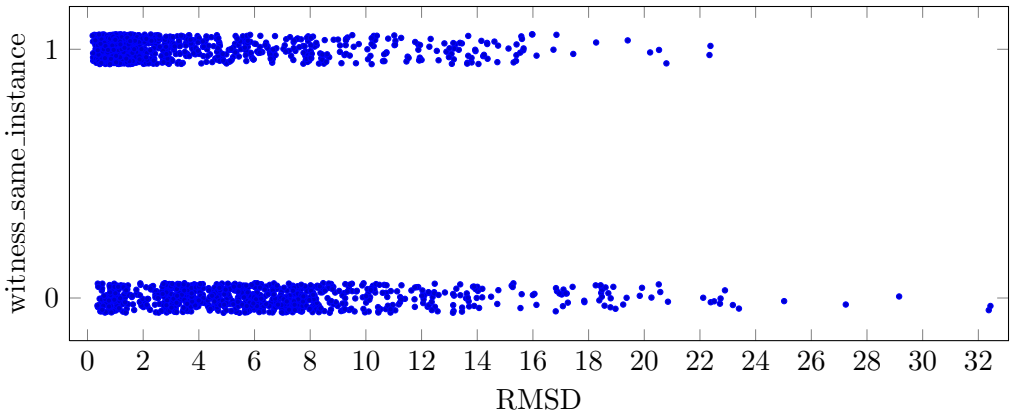


Figure 5.14: Feature witness_same_instance; see section 5.4

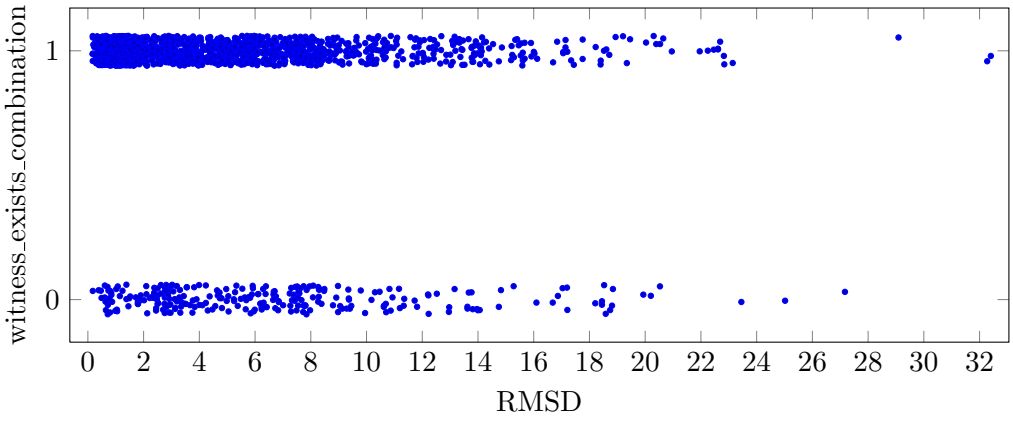


Figure 5.15: Feature witness_exists_combination; see section 5.4

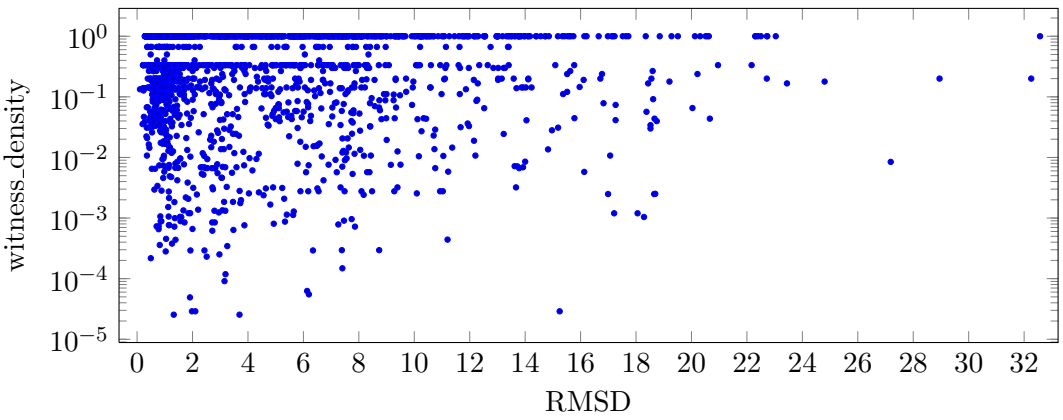


Figure 5.16: Feature witness_density; see section 5.4

The ideas is that there might be a dependency between these gap sizes: Let's say that all witnesses *and* the target have roughly the same gap size. Maybe such a match is preferable to a match where the witnesses are less consistent and/or the target structure has a completely different gap size. I suspect this because such big insertions are less likely for homologous proteins and therefore the probability of a chance alignment is higher for such matches. Even if big sequence insertions/deletions between the two fragments were to happen evolutionary, they could affect the overall topology of the protein and thereby even overcome the conservedness of the building block.

Very small gap sizes are particularly interesting: The smaller the gap size, the higher the impact that local interactions have. Also, the probability that the region *between* the bracing fragments is conserved (but not part of the building block because it has no well-defined secondary structure), increases with smaller gap sizes. If the region between the fragments is conserved, that makes the structure of the fragments (especially the spatial relation between them) likely to be more conserved. I used a threshold of twelve residues to distinguish "local" matches from non-local matches. This choice is backed up by Fernandez-Fuentes et al. [2010], who developed a library of so called "smotifs" which shares some of the same motivations as our building block library. With some slight modifications, the building block library can be considered a superset of the smotif library. A smotif consists of two fragments that are either alpha helices or beta sheets. They are connected by a region of random loop structure that has a maximum length of *twelve* residues. They justify this choice because they have shown that "Smotifs with loop fragments having lengths up to twelve residues, together with their bracing secondary structure elements are exhaustively sampled in the Protein Data Bank".

So, the resulting features are:

seq_separation_align The gap size in the target sequence, i.e. $m_b(i_2) - m_e(i_1) - 1$ for $m_b(i_1) < m_b(i_2)$.

seq_separation_avg The average gap size across all instances, i.e. $\text{avg}_{BBI \in BB} o_b(BBI, i_2) - o_e(BBI, i_1) - 1$ for $o_e(BBI, i_1) < o_b(BBI, i_2)$. → fig. 5.17

seq_separation_stddev The gap size standard deviation across all instances. A low standard deviation together with a high sequence similarity between the instances might be a bad sign because witnesses are closely related and the building block might therefore not be generalizable.

seq_separation_diff This feature tries to measure how similar the sequence separation of the match is to the sequence separation of the instances, i.e. $|\text{seq\_separation\_avg} - \text{seq\_separation\_align}|$. → fig. 5.18

seq_separation_similar With a similar motivation as seq_separation_diff, this features is "1" if seq_separation_align is within $[\text{avg} - \text{stddev}, \text{avg} + \text{stddev}]$, "0" otherwise.

seq_separation_12_align "1" if seq_separation_align ≤ 12, "0" otherwise. → fig. 5.19

seq_separation_12_ratio The fraction of instances that have a gap size ≤ 12. A high fraction indicates a building block that might have significant local interaction and a conserved region between the fragments.

seq_separation_12_ratio_align seq_separation_12_align · seq_separation_12_ratio → fig. 5.20

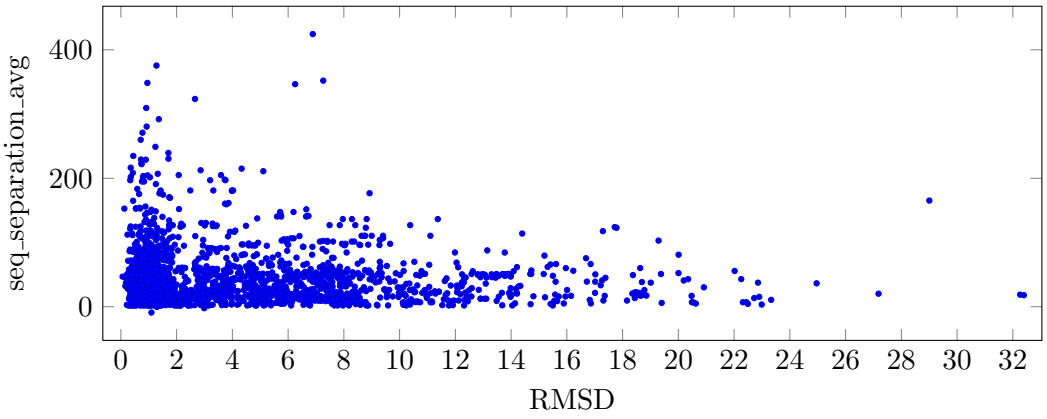


Figure 5.17: Feature seq_separation_avg; see section 5.5

5.6 Hydrophobicity based features

Even though the solution (e.g. surrounding water) is not explicitly modeled in most protein structure prediction approaches, real world proteins are typically submerged in some kind of aquatic solution. Since water molecules are polar, they prefer bonding with other polar molecules. Non-polar molecules disrupt these bonds and thus it is energetically unfavorable if such molecules are in direct contact with water. It has been found that protein folding avoids these contacts by burying non-polar side-chains on the

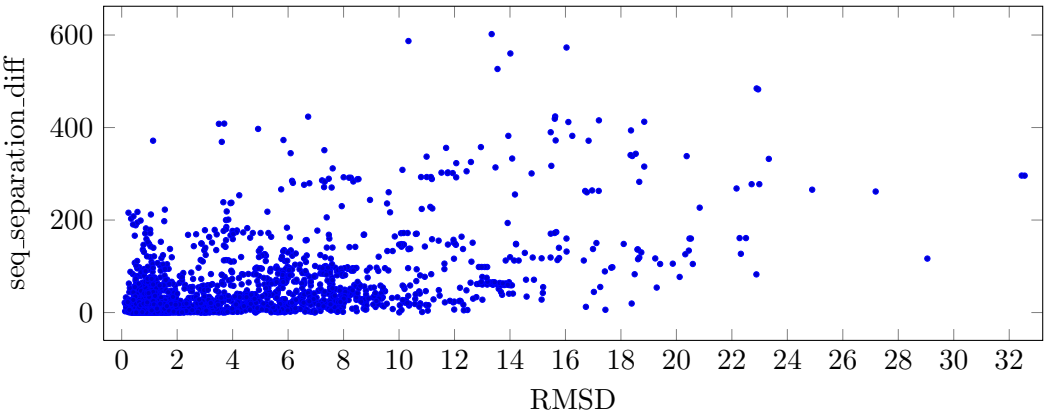


Figure 5.18: Feature seq_separation_diff; see section 5.5

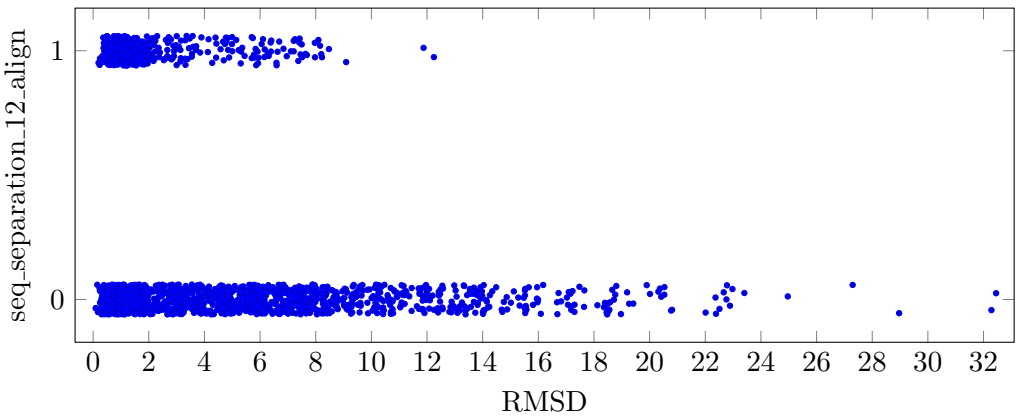


Figure 5.19: Feature seq_separation_12_align; see section 5.5

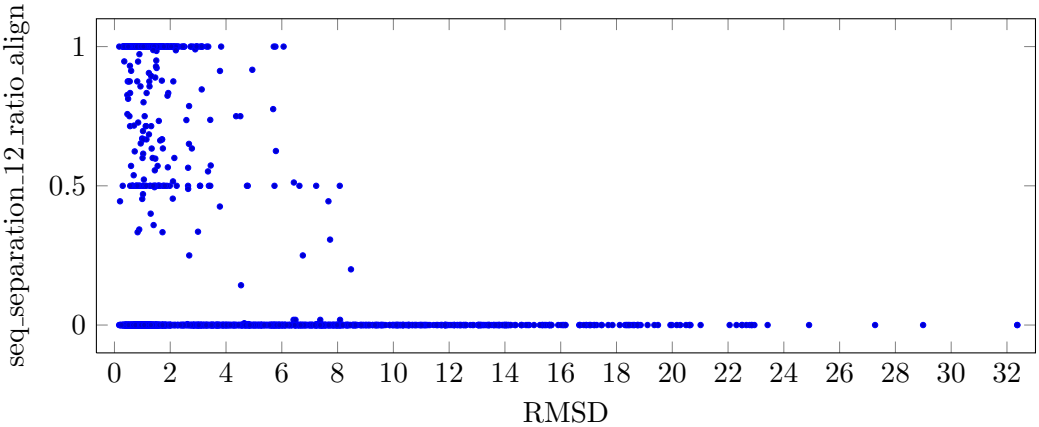


Figure 5.20: Feature seq_separation_12_ratio_align; see section 5.5

f

inside of a protein fold. In consequence, an alpha helix or beta sheet that is located on the surface of a globular protein tends to have a pattern regarding the hydrophobicity of its amino acids. For example, a regular alpha helix has 3.6 amino acids per full 360 degree turn; this means that for an alpha helix on the protein surface one would expect a hydrophobic amino acid roughly every fourth residue. Such patterns can be visualized in so-called Edmundson wheel projections "which are projections of the amino acid side-chains onto planes perpendicular to the long axes of the helices." [Schiffer and Edmundson, 1967]. So-called "hydrophobicity scales" assign each of the 20 amino acids a positive (hydrophobic) or negative (hydrophilic) value. These values can be represented as vector lengths in an Edmundson projection. Figure 5.21 shows a projection that is very likely a surface alpha helix because hydrophobic vectors are grouped together in the north-east and hydrophilic vectors are grouped together on the opposite side when using a 100 ° angle between residues. If a different projection angle is chosen, say 160 °, the angle of a twisted beta strand, the pattern disappears even though the sequence is the same (see fig. 5.22). In fig. 5.21 you can also see a green vector pointing in the north-east direction, which is the mean hydrophobic moment, "the vectorial sum of all the hydrophobicity indices, divided by the number of residues" [Tossi and Sandri, 2001]. The vector is actually also there in fig. 5.22, but has near-zero length.

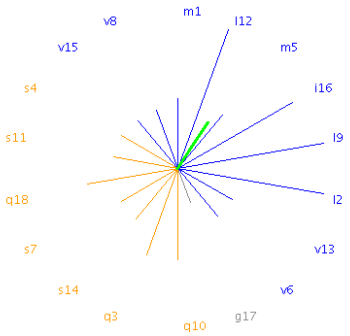


Figure 5.21: Example HydroMCalc plot at 100 °

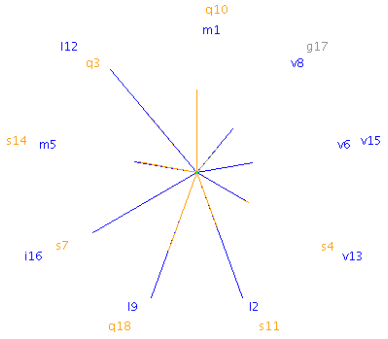


Figure 5.22: Example HydroMCalc plot at 160 °

As you can see, the hydrophobicity can be used to validate secondary structure hypotheses, but this is not the goal here because secondary structure is already part of HHsearch's "Probability". HHsearch in turn uses PSIPRED which is a widely used secondary structure prediction software, so I do not expect to be able to improve on that. Also, amino acid replacements with similar hydropathy are already favored by

amino acid substitution matrices and thereby already somewhat part of the alignment score.

I still think it is worth to model this effect explicitly, because such structures have been shown by Bowie et al. [1990] and others to be highly conserved. That makes a matching hydropathy pattern an even stronger indicator for homology than a high scoring sequence alignment alone and thus increases the likelihood of co-evolution (c.f. lemma 4.1) because a fragment might not only match in local structure but also in placement (i.e. surface facing residues in the witness are surface facing residues in the target as well). Further one could argue that if one of the fragments is on the protein surface, the other fragment is relatively likely to be in the interior of the protein. This makes the other fragment more likely to be evolutionary conserved as well, because "virtually all types of amino acid residues are found to have higher mutabilities on the exterior than in the interior." according to Go and Miyazawa [1980]. This intuitively makes sense because interior/hydrophobic residues have more chances for inter-residue contacts than surface residues.

alpha_helix_min The hydrophobic moment of a sequence is a vector that can be written in polar coordinates with an angle θ and a length l. Let w_i ($i \in M$) be the hydrophobic moment of fragment i's sequence. Let t_i ($i \in M$) be the hydrophobic moment of the corresponding target sequence portion. The similarity between the vectors can be calculated as $\text{sim}(t_i, w_i) = cos(\theta_{t_i} - \theta_{w_i}) \cdot l_{t_i} \cdot l_{w_i}$. Then, this feature is defined as $\min_{i \in M} \text{sim}(t_i, w_i)$, i.e. the worse similarity of both fragments.

alpha_helix_max $\max_{i \in M} \text{sim}(t_i, w_i)$, i.e. the better similarity of both fragments. → fig. 5.23

acc_min DSSP [Kabsch and Sander, 1983] can calculate a "solvent accessibility" value for each residue of a known protein *structure*. This feature is $\min_{i \in M} \sum_{r=o_b(i)}^{o_e(i)} s(r)$ with $s(r)$ being the solvent accessibility at residue r.

acc_max $\max_{i \in M} \sum_{r=o_b(i)}^{o_e(i)} s(r)$

acc_sum $\sum_{i \in M} \sum_{r=o_b(i)}^{o_e(i)} s(r)$ → fig. 5.24

acc_avg acc_sum divided by length_total. A low accessibility is typical for buried fragments. So, a low acc_min means that at least one fragment is buried; low acc_max/acc_sum/acc_avg mean that both fragments are buried.

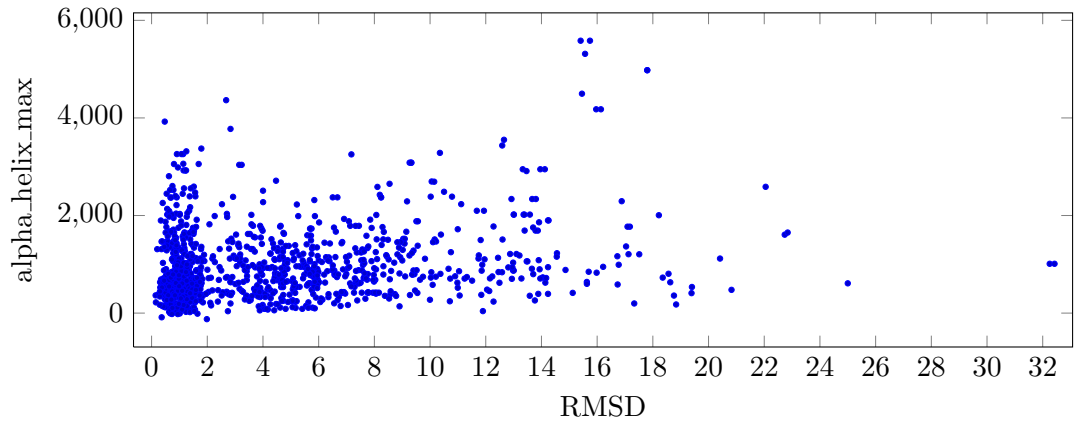


Figure 5.23: Feature alpha_helix_max; see section 5.6

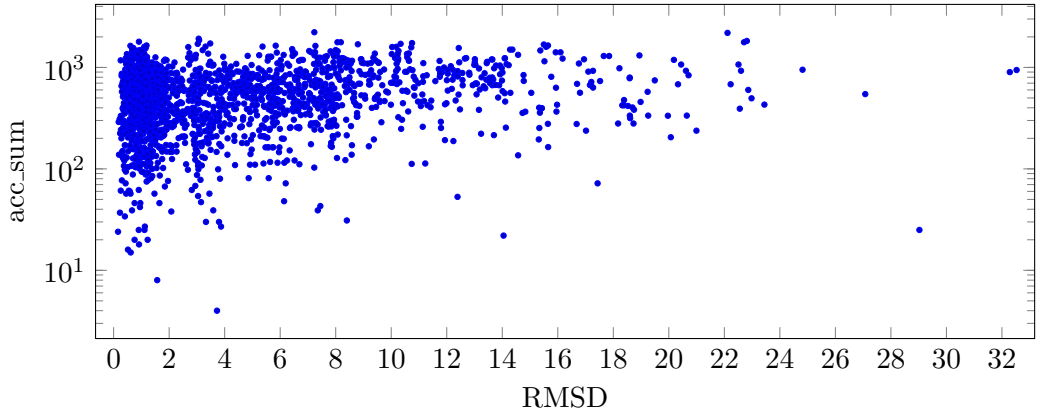


Figure 5.24: Feature acc_sum; see section 5.6

5.7 SCOP based features

The "Structural Classification of Proteins" (SCOP) is a humanly curated and regularly updated hierarchic database of protein domains [Murzin et al., 1995]. The first four levels of the hierarchy are: class, fold, superfamily and family. "Proteins are clustered together into families on the basis of one of two criteria that imply their having a common evolutionary origin: first, all proteins that have residue identities of 30% and greater; second, proteins with lower sequence identities but whose functions and structures are very similar". If it is less clear that two proteins are evolutionary related, but a common ancestry is still probable due to structural and functional similarity, they

are assigned to the same superfamily. Proteins that are in the same fold might not be related at all. Both the delineation of protein domains and the classification are done manually/visually with the help of of some automated tools.

The idea is that a building block where all witnesses are in the same SCOP family could be less preferable than a building block that comes from multiple different (super-)families. This is because the sequence-structure relationship from the former building block is likely highly specific to the function/sequence of its family while the later building block might have a more generalizable sequence-structure relationship. So the feature definition is straightforward:

num_scop_families $|M_f|$ with M_f being the set of SCOP families that are found among all equivalent fragments.

num_scop_superfamilies $|M_{sf}|$ with M_{sf} being the set of SCOP superfamilies that are found among all equivalent fragments. → fig. 5.25

Both features can be interpreted as cleaned up versions of no_of_witnesses (see section 5.3).

The HHsearch user guide [Söding, 2006] lists another idea: "Check relationship among top hits: If several of the top hits are homologous to each other, (e.g. when they are members of the same SCOP superfamily), then this will considerably reduce the chances of all of them being chance hits". To make this work for building blocks, I am using the following algorithm: For all top matches (using 50%/30% "Probability" as a threshold, see "seq_is5030" in section 5.1) I extract their (super-)families.

top_scop_families $|T_f \cap M_f|$ with T_f being the set of SCOP families that are found among the top hits.

top_scop_superfamilies $|T_{sf} \cap M_{sf}|$ with T_{sf} being the set of SCOP superfamilies that are found among the top hits. → fig. 5.26

5.8 Spatial features

Let's define a fragment as a set of atoms where each atom is represented by its Cartesian coordinates. Given a relation $p(w, r)$ that returns the "interaction center"

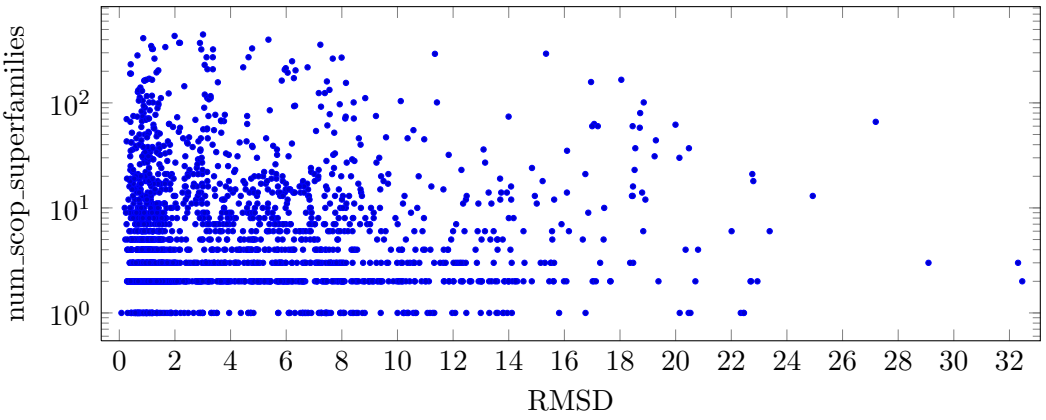


Figure 5.25: Feature num_scop_superfamilies; see section 5.7

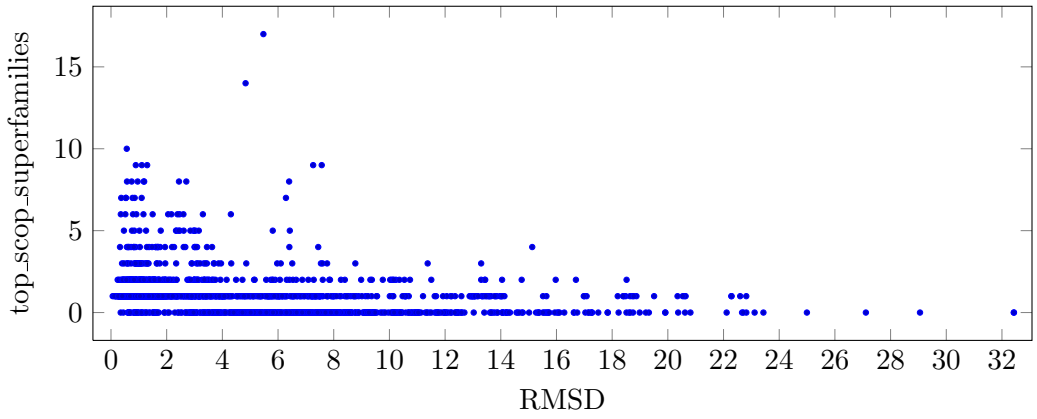


Figure 5.26: Feature top_scop_superfamilies; see section 5.7

(a Cartesian coordinate) for each residue r of a witness w, $I = \{o_b(i_1), \ldots, o_e(i_1)\}$ and $J = \{o_b(i_2), \ldots, o_e(i_2)\}$ one can calculate the distance between these bipartite sets of coordinates as:

$$\text{distance\_min} = \min_{i \in I,\, j \in J} |p(w,i) - p(w,j)|$$

$$\text{distance\_max} = \max_{i \in I,\, j \in J} |p(w,i) - p(w,j)|$$

I calculated the interaction center of a residue according to a definition by Park et al. [1997]: It is located 3 Å from the C_α atom along the vector pointing towards the C_β atom[1].

The closer two fragments are together, the stronger their atomic interactions with each other. Such "contacts" might stabilize the structure; structures with more contacts are more likely to be evolutionary conserved (c.f. section 1.3). Thus, the expectation is that the lower the distance between two building block fragments the higher their conservation.

One might ask the question if fragments aren't per the building block definition in close contact with each other and if distance isn't a superfluous feature then. Well, first of all the building block definition uses a distance cut-off, which if broad enough might still leave room for a correlation. Second of all, the distance cutoff (both algorithmically and numerically) is chosen rather arbitrarily, i.e. without much experimentation. So, if for example one finds many true positives near the cutoff, one might have to think about increasing it.

A low distance_max value is a strong signal for contacts, because it states how close *every* atom from the first fragment is to every atom of the second fragment. Whereas a low distance_min just states how close *one* atom from the first fragment is to one atom of the second fragment. Combined, both values can be used to draw conclusions about the spatial orientation of the fragments: If distance_max is far bigger than distance_min, one can assume the two fragments to be relatively orthogonal to each other and if both values are similar, one would expect them the fragments lying roughly in parallel. This knowledge is useful because parallel fragments not only have more atoms that are able to bond, but also have more bonding partners for each individual atom.

[1] except for Glycine where the interaction center is defined as the location of the C_α atom

Unfortunately, the difference between distance_min and distance_max is correlated with the fragment lengths; distance_avg tries to compensate this[1]:

$$\text{distance\_avg} = \frac{\sum_{i \in I} \min_{j \in J} |p(w,i) - p(w,j)|}{|I|}$$

Given a building block match, all three measures can be calculated for every $eq(BB,\ M)$. This posses the question how to aggregate the individual measures to one feature: minimum, maximum and average come to mind. With *agg* being either min, avg or max:

distance_min:*[agg]* $\text{agg}_{\{eqBB,\ eqM\} \in \text{eq}(BB,\ M)} \text{distance\_min}(eqBB, eqM)$ → fig. 5.27

distance_avg:*[agg]* $\text{agg}_{\{eqBB,\ eqM\} \in \text{eq}(BB,\ M)} \text{distance\_avg}(eqBB, eqM)$

distance_max:*[agg]* $\text{agg}_{\{eqBB,\ eqM\} \in \text{eq}(BB,\ M)} \text{distance\_max}(eqBB, eqM)$ → fig. 5.28

I also tested different strategies that focus more directly on the number of contacts: A contact is assumed if two residues are in close range, for example if the interaction centers of both residues are within a certain threshold. With c being a function that returns "1" if two residues are in contact, "0" otherwise and *threshold* being the contact threshold $\in \{5.5, 7.0\}$:

number_of_contacts*[threshold]*:*[agg]* $\text{agg}_{\{eqBB,\ eqM\} \in \text{eq}(BB,\ M)} \sum_{i \in I} \sum_{j \in J} c(i,j)$

number_of_residues_in_contact*[threshold]*:*[agg]*

$\text{agg}_{\{eqBB,\ eqM\} \in \text{eq}(BB,\ M)} \sum_{i \in I} \text{signum} \sum_{j \in J} c(i,j)$

5.9 Statistical coupling based features

The term "statistical coupling" in relation to protein sequence was coined by Lockless and Ranganathan [1999]. Statistical coupling analysis (SCA) is based on the observation that some pairs of residues covariate stronger than others throughout evolution. "A loss-of-function point mutation in a protein is often rescued by an additional mutation that compensates for the original physical change. According to one hypothesis, such compensation would be most effective in maintaining a structural motif if the two

[1] assuming that $|I| \geq |J|$; accordingly for $|J| > |I|$

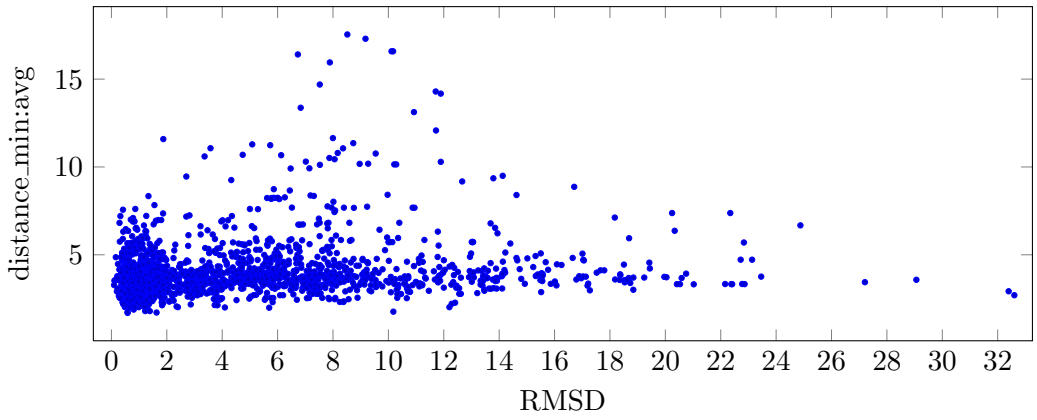


Figure 5.27: Feature distance_min:avg; see section 5.8

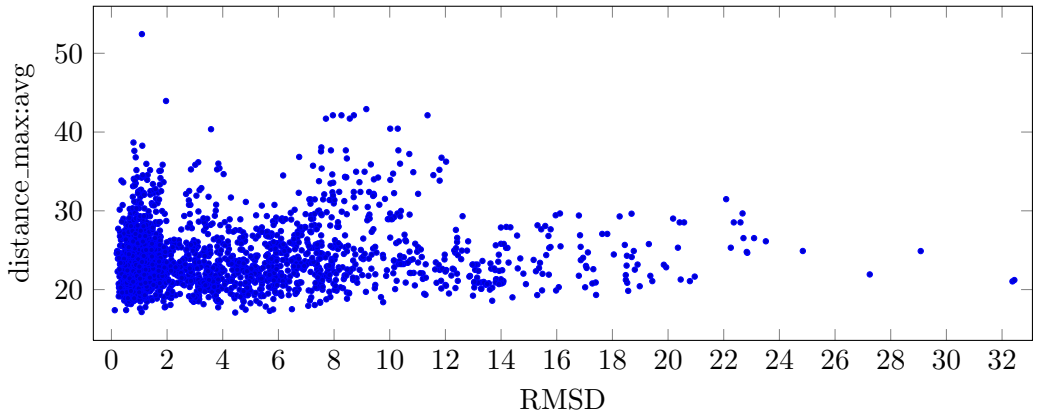


Figure 5.28: Feature distance_max:avg; see section 5.8

mutated residues were spatial neighbors" [Neher, 1994]. The objective is to take a multiple sequence alignment (MSA) of n residues length and turn it into a $n \times n$ matrix that reflects how strongly any two sites (= residue positions) are coupled. For example, if two sites are spatially in contact and one of them is replaced with a smaller side chain, the contact could be kept "intact" by replacing the other one with a bigger side chain. Suel et al. [2002] have found that statistical coupling between sites can be used to predict contacts. However, contacts do not necessarily result in statistically coupled sites [Pazos and Valencia, 2008]. It has also been found that SCA can be used to discriminate correct vs. incorrect folds [Bartlett and Taylor, 2008].

In terms of building block features, one could say that SCA is the sequential counterpart to section 5.8. Maybe SCA can complement the spatial features by finding evolutionary constraints in a broader sense. My hypothesis is that building blocks that have a high sequential coupling between the residues of their fragments are more evolutionary conserved and therefore preferable when it comes to structure prediction. Furthermore, if two fragments have residues that are both in contact *and* sequentially coupled, that might be a strong signal for sequence-structure correlation. I further hypothesize that a strong sequence-structure correlation makes the sequence alignment score more correlated to the structural similarity. The later correlation is usually the great unknown of fragment retrieval by sequence (c.f. the "black box" in section 1.5).

Sadly, all SCA features share the same mayor flaw as the spatial features: they do not depend on the target sequence. Even though we could include the target sequence in the underlying MSA, these MSAs typically contain 3-figure numbers of sequences, so that the impact of the target sequence is negligible. This brings us to another problem: In order to calculate a meaningful coupling matrix, the MSA needs to consist of *non-redundant* sequences only. One typically filters the sequences with a pairwise sequence identity threshold. Additionally, a certain minimum number of sequences needs to remain after the filtering. Estimates for the minimum required number of non-redundant sequences to produce meaningful results range from 30 to 125 [Dickson et al., 2010].

For this thesis, the statistical coupling matrix was generated as follows: Each row of the MSA was generated by concatenating the sequences of two (equivalent) building block instance fragments. The MSA was filtered with an 80 % sequence identity threshold and turned into a sequence profile. Using that profile, for each site a "conservation"

value can be calculated. The "conservation" is a non-linear measure of how much the amino acid distribution differs from a random amino acid distribution. Note, that the random distribution is not flat because some amino acids occur more often than others in nature. Then, for each pair of sites it can be calculated how a perturbation in one site changes the conservation value of the other site. To calculate conservation values and the statistical coupling matrix, I used an implementation by Yip et al. [2008].

sca_msa_size The size of the MSA, i.e. the number of non-redundant sequences across all (equivalent) building block instances. Note that the following features are only defined for sca_msa_size ≥ 30. → fig. 5.29

conservation_*[agg]* The min, max and avg of the conservation value at any of the $n = |I| + |J|$ sites.

sca_avg The SCA matrix is a symmetric $n \times n$ matrix. This feature averages all matrix cells that represent an inter-fragment coupling, i.e. $\frac{1}{|I| \cdot |J|} \cdot \sum_{i \in I} \sum_{j \in J} \mathrm{SCA}_{i,j}$.

sca_stddev The standard deviation of all inter-fragment coupling values.

sca_over_stddev*[factor]* Counts how many inter-fragment coupling values are greater than sca_avg + sca_stddev · factor with *factor* $\in \{1, 2\}$. → fig. 5.30

number_of_sca_contacts*[threshold]*:*[agg]* This feature is an attempt to model the correlation between sequence conservation and structural contacts. Similar to number_of_contacts*[threshold]*:*[agg]*, but only counting contacts where the two sites have a statistical coupling value above sca_avg + sca_stddev. → fig. 5.31

As already mentioned, the main problem with the SCA based features is that they are only defined for sca_msa_size ≥ 30 which usually is only given for $\approx 20\%$ of matches. Another problem is that there are many possible ways to turn a *matrix* into a single representative feature.

5.10 Structural compatibility features

The following idea and most of its code was contributed by Mahmoud Mabrouk. For his thesis (assembly), he was examining whether sequentially overlapping matches were structurally compatible or not. Compatible matches can be used alongside each other

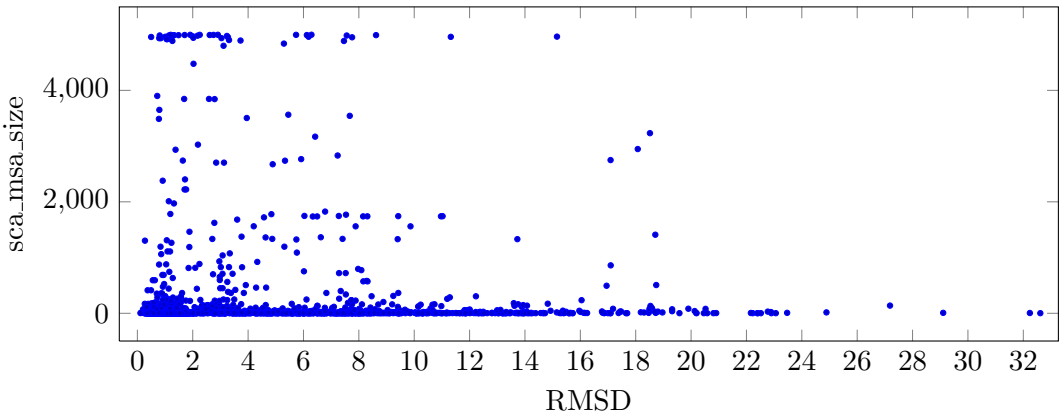


Figure 5.29: Feature sca_msa_size; see section 5.9

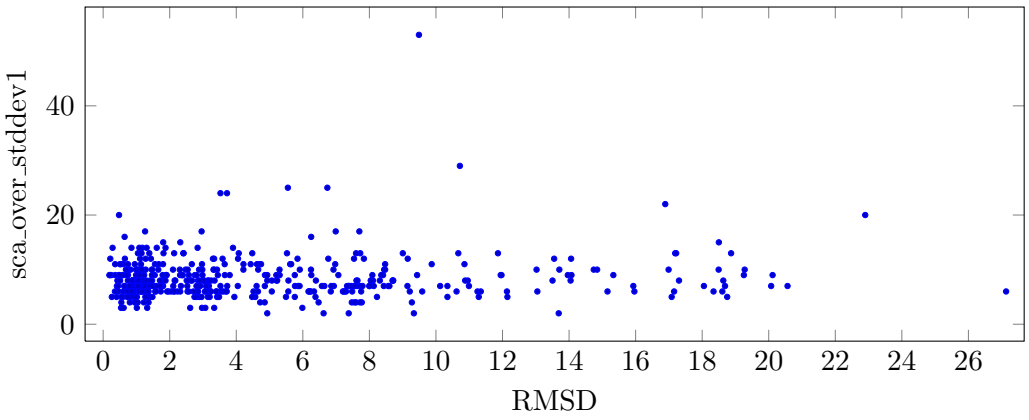


Figure 5.30: Feature sca_over_stddev1; see section 5.9

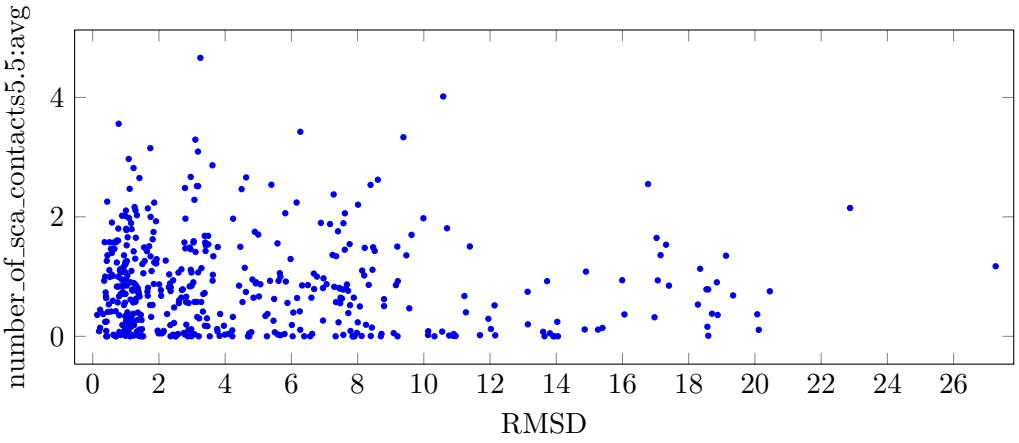


Figure 5.31: Feature number_of_sca_contacts5.5:avg; see section 5.9

to generate prediction constraints; non-compatible matches are mutually exclusive. In practice, compatibility is defined as both matches being superimposable within a certain threshold.

What is especially compelling about this idea is that correct matches are mostly[1] compatible with other *correct* matches. At the same time, incorrect matches are relatively unlikely to be compatible with each other. In consequence, the more compatible matches a building block has, the more likely it is a correct match.

He originally implemented this idea by first doing an all vs. all structure alignment of the overlapping parts of the matches. This essentially gave him a weighted graph where the vertices are the matches and the edge weight is given by their RMSD. He then applied a cut-off to obtain an unweighted graph and mined this graph for maximum cliques. The building blocks in the biggest clique(s) were pretty likely to be correct matches.

The problem with this strategy is that it does not make a statement about most of the building blocks, because they are not part of the top clique(s). However, a correct match could have little to none sequentially overlapping & compatible matches, especially if there is a small number of correct matches. Thus, this approach needs to process a lot of matches; even if those are available for one particular target, an all vs. all structural alignment is quite computationally expensive; so is mining for maximum cliques.

Based on his initial idea, the following features – applicable to *all* matches which have at least one sequentially overlapping match – are defined:

struct_compat_overlaps The number of sequentially overlapping matches (with at least 3 overlapping residues).

struct_compat_rmsd The average RMSD between these matches (all vs. all). → fig. 5.32

struct_compat_compatibility The number of these matches that have at least one such RMSD $< 2.7\,\text{Å}$ divided by struct_compat_overlaps. → fig. 5.33

[1] since we are working with cutoffs, compatibility/correctness is not transitive

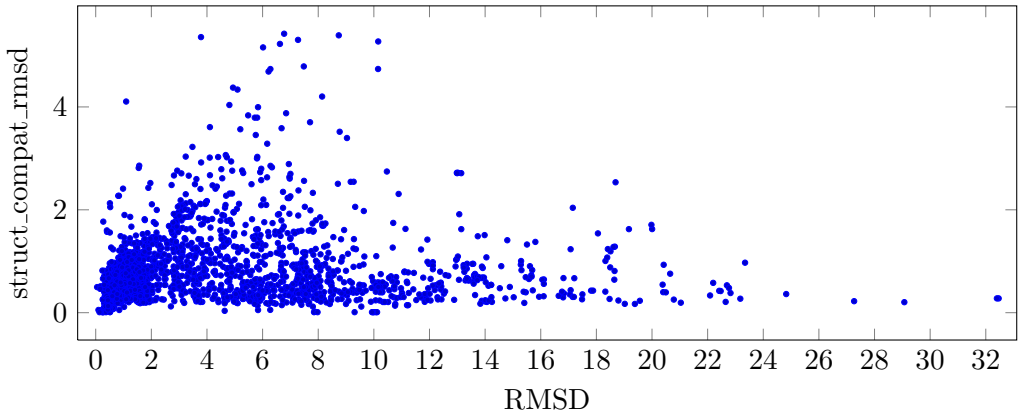


Figure 5.32: Feature struct_compat_rmsd; see section 5.10

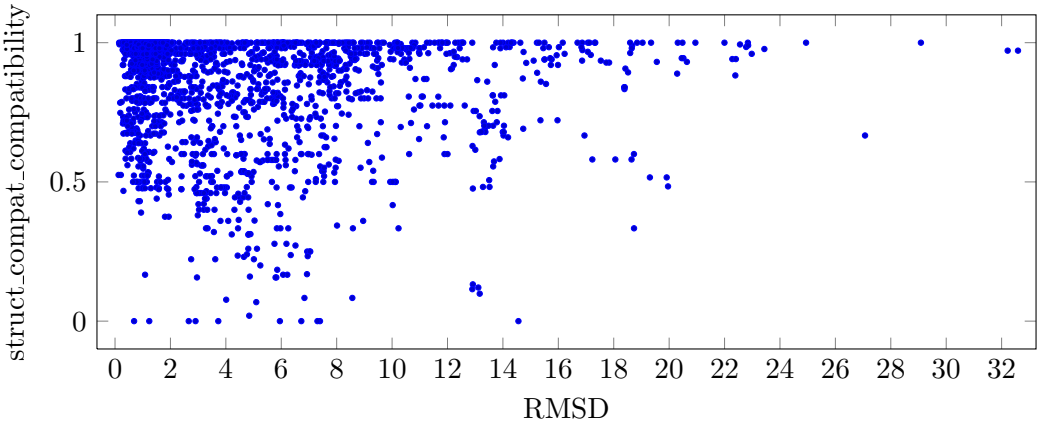


Figure 5.33: Feature struct_compat_compatibility; see section 5.10. This plot is quite the opposite of what I expected; I would have expected it to be reflected across a horizontal line at 0.5. Values of exactly 1.0 are easy to explain: many of them are not significant because struct_compat_overlaps could be 1. The rest of the expected correlation is probably displaced by the following phenomena: The shorter the target protein, the higher the probability of an sequential overlap and the lower the information gain through that. Also, overlapping matches seem to be often homologous to each other. This explains why high compatibilities are spread across the whole RMSD spectrum. It doesn't explain why low compatibilities are mostly found for low RMSDs, though.

5.11 Sequence clustering based features

To recall, HHsearch's "Probability" tries to model the probability $P(A_i \cap B_i) = P(A_i) \cdot P(B_i \mid A_i)$ (c.f. fig. 4.4 for event definition) with mixed results. This is not to say that HHsearch performs bad – a great deal of it's matches are correct – it's just that the ranking of the matches is too noisy.

Because sequence is literally the only information we have about both the target protein and the building blocks, it makes sense to have a closer look. The following idea is based on the fact that some sequence fragments occur (with slight variations) more often than others in the PDBSS. For simplicity, I will explain the idea for *single-fragment* matches first: Let's say we have a fragment where the sequence (or a very similar one) occurs often in the PDBSS, but has many completely different structures; and there is another fragment whose sequence occurs equally often, but always has the same structure. Let's also assume that HHsearch found both matches on the same target protein and that both matches have similar scores. If I had to choose between one of these matches, I suppose the later one would be the better choice because it has proven to be a more robust sequence-structure relationship.

To formalize this idea, one can express it as a conditional probability:

$$P(B_i^* \mid A_i^*) = \frac{P(A_i^* \cap B_i^*)}{P(A_i^*)}$$

Note how there is an asterisk at each event which is meant to express that the alignment position is neglected, i.e. fragment i aligns at least once *anywhere* on the target (A_i^* and B_i^* still need to align at the same position of course). Consequently these probabilities are agnostic of the sequence alignment score; this seems acceptable because the score is already contained in other features.

Unfortunately $P(A_i^* \cap B_i^*)$ and $P(A_i^*)$ are hard to obtain. But assuming that the size N of the PDBSS is big enough and samples the target protein's homologs equally, one could approximate them with the relative frequency:

$$P(A_i^* \cap B_i^*) \approx \frac{|A_i^* \cap B_i^*|}{N}$$

$$P(A_i^*) \approx \frac{|A_i^*|}{N}$$

Where $|A_i^*|$ is the number of witnesses that contain at least one fragment that has a similar sequence and $|A_i^* \cap B_i^*|$ is the number of witnesses that contain at least one fragment that additionally has the same structure as the fragment in question.

For two-fragment building block matches, one can say accordingly:

$$P(B_1^* \cap B_2^* \mid A_1^* \cap A_2^*) = \frac{P(B_1^* \cap B_2^* \cap A_1^* \cap A_2^*)}{P(A_1^* \cap A_2^*)}$$

with

$$P(B_1^* \cap B_2^* \cap A_1^* \cap A_2^*) \approx \frac{|B_1^* \cap B_2^* \cap A_1^* \cap A_2^*|}{N}$$

$$P(A_1^* \cap A_2^*) \approx \frac{|A_1^* \cap A_2^*|}{N}$$

resulting in

$$P(B_1^* \cap B_2^* \mid A_1^* \cap A_2^*) \approx \frac{|B_1^* \cap B_2^* \cap A_1^* \cap A_2^*|}{|A_1^* \cap A_2^*|}$$

As hinted before, the problem is that this approximation uses the law of large numbers and in this context $N \approx 5100$ is probably not large enough. Also, the absolute frequencies are often both 1. This makes $P(B_1^* \cap B_2^* \mid A_1^* \cap A_2^*) = 100\%$ which is of course an artifact. To enable the learner can detect this, I will additionally use the numerator and the denominator as separate features.

bayesian1_denominator_*[mt]* The number of witnesses where there exist two non-overlapping building block fragments f_1 and f_2; The sequence of f_x has to belong to the same sequence cluster as the i_xth fragment of the match.

bayesian1_numerator_*[mt]* The subset of witnesses mentioned before where f_1 and f_2 come form a building block instance that belongs to the same building block as the match.

bayesian1_*[mt]* bayesian1_numerator_*[mt]* divided by bayesian1_denominator_*[mt]*.

bayesian2_denominator_*[mt]* Similar to bayesian1_denominator_*[mt]* but f_x may also come from the same sequence cluster as any fragment that aligned at the exact same target residue range as the i_xth fragment of the match. Ideally, all fragments that align at the same spot should belong to the same sequence cluster. In reality this isn't always the case because HHsearch can find distant sequence similarities and because the target sequence fragment might be equally close to multiple clusters (aliasing).

bayesian2_numerator_*[mt]* Analogous to bayesian1*[mt]*. → fig. 5.34, fig. 5.35

bayesian2_*[mt]* bayesian2_numerator_*[mt]* divided by bayesian2_denominator_*[mt]*. → fig. 5.36

with *mt* being the sequence clustering method and threshold; one of:

I14 Using MCL (see section 3.4) with an inflation parameter of $i = 1.4$.

I16 As before, using an inflation parameter of $i = 1.6$.

T5 Sequences belong to the same sequence cluster if they can be aligned with an E-value ≤ 5 (note that a sequence can belong to multiple clusters according to this definition).

T10 As before, using an E-value threshold of 10.

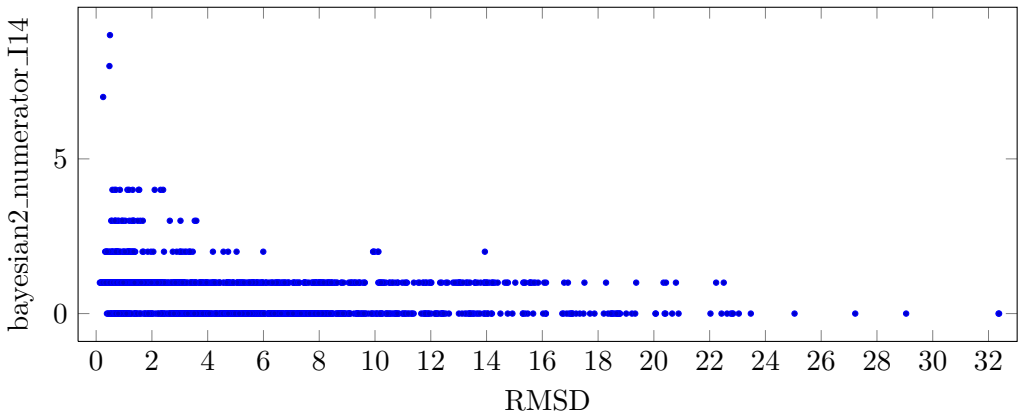


Figure 5.34: Feature bayesian2_numerator_I14; see section 5.11

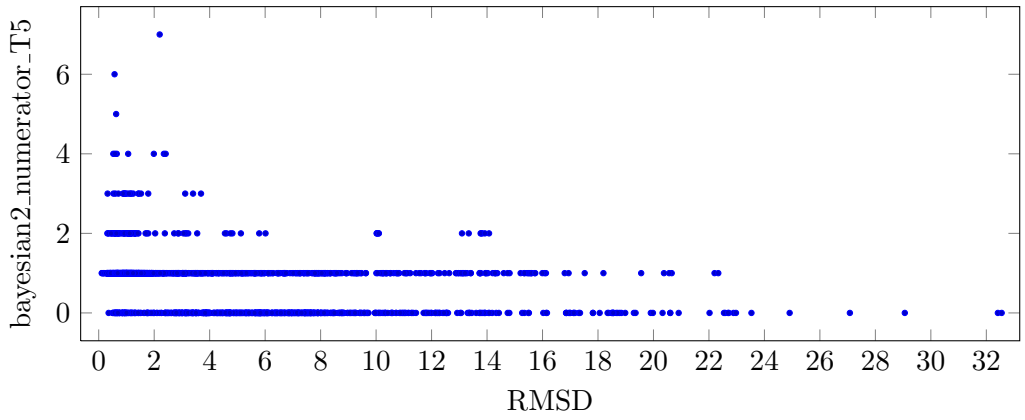


Figure 5.35: Feature bayesian2_numerator_T5; see section 5.11

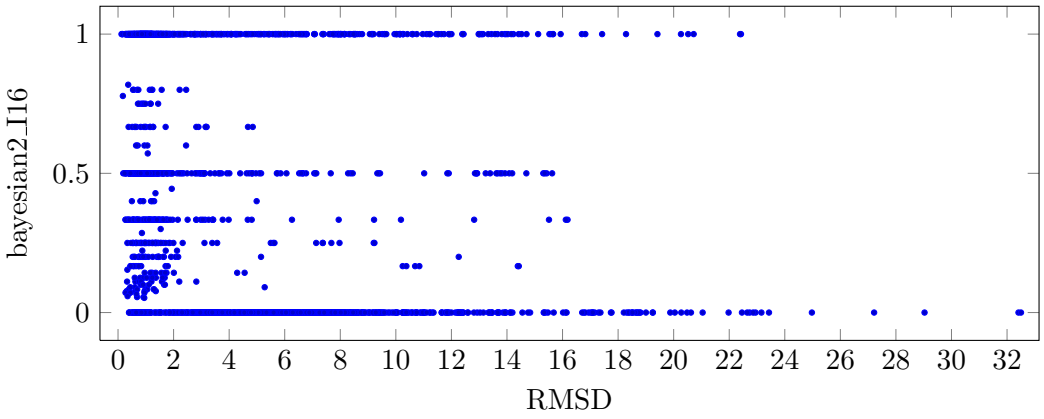


Figure 5.36: Feature bayesian2_I16; see section 5.11

6

Classification

6.1 Training set

In supervised learning one usually partitions the samples (each two-fragment building block match being a sample) into a training set and a test set. The training set is used to train the classifier and the test set is used to measure its performance. To avoid artifacts induced by the partition, one generates multiple different partitions and averages the results.

Under the assumption that all samples are independent, the samples can be randomly assigned to either the training set or the test set. Unfortunately, this is not the case for building block matches: When samples that come from the same target protein are found in both the training and the test set, classifiers perform much better (c.f. Gao et al. [2009]). However, in a blind protein structure prediction situation like CASP, all samples of the current target are unlabeled. Therefore the samples have to be partitioned in a way that samples from the same target are only found in exactly one of the two sets.

Another problem are unbalanced training sets: Many machine learning algorithms act undesirably when fed with a training set that has way more samples from one class ("majority class") then from the other class ("minority class"). If for example 90 % of samples are labeled with the positive class (correct matches), the resulting model might predict the positive class for *any* input. Ideally a training set contains roughly the same number of samples per class.

6.1.1 Leave-one(-target)-out validation

A simple way to do validations given these constraints is to regard the samples from each target protein as a different test set. The training set is then defined as all the remaining samples. To avoid the class imbalance problem, the training set is balanced through majority class under-sampling in a way that for each target protein in the training set there is roughly the same number of positive and negative samples (i.e. the majority class is determined independently for each target protein). Also the number of samples per target protein is limited to a maximum number (currently 40) in order to avoid over-representation of a certain target protein.

The problem with this leave-one-out type of validation is that some target proteins have only very few samples of one or both of the two classes while other target proteins have thousands of samples. Depending on how one calculates/averages the performance measures, this either leads to unjustified under-representation of such target proteins or to hazardously low resolution of the performance measures. For example, performance measures for a target protein that has only two samples can only be 0 %, 50 % or 100 %.

6.1.2 K-fold cross-validation

To address this problem, one can aggregate target proteins into bigger batches: In k-fold cross-validation one divides the dataset into k roughly same sized subsets ("batches"). A new model is then trained k times, each time using a different one of these batches as the test set and the rest of the dataset as the training set.

The partition into batches and the composition of each batch have to be optimized according to multiple criteria: batches should be balanced and have roughly the same size; targets should not be hugely over-represented and map to exactly one batch. At the same time, as few samples as possible should be deleted in order to avoid loosing precious data. Of course there is often no solution that fulfills all these criteria, so I developed a heuristic algorithm that finds an acceptable solution for most cases.

One mayor difference between cross-validation and leave-one-target-out validation is that the test set of the former is balanced while the test set of the later is not necessarily balanced. Thus it is necessary to verify the performance measures obtained for cross-validation, see section 7.1.

6.2 Feature selection

Different learning algorithms have varying susceptibility for noisy and/or redundant features. For example, a k-nearest-neighbor (kNN) classifier that uses an Euclidean distance measure puts the same weight on every dimension. Therefore, a noisy dimension diminishes the weight of each meaningful dimension and redundant dimensions effectively increase the weight of the aspect they are modeling. Also, the calculation of some of the previously mentioned features is quite time consuming, so it would be beneficial to know if some of them don't add any value and thus can be skipped entirely.

Feature selection algorithms can be classified into filter, wrapper and embedded methods. They can also be classified whether they are univariate or multivariate. The former look at features independently, the later also identify features that augment each other when selected together. A wrapper method typically wraps around a cross-validation and optimizes the performance by trying different subsets of features. A filter method finds an optimal feature subset independently from the learner and embedded methods integrate the feature selection into the learning process e.g. decision trees or other learners that include a pruning step.

In this thesis it comes down to using either a filter method or a wrapper method. The problem with most wrapper methods is that they take a long time to finish. With currently around 140 feature variations, a brute force wrapper is out of the question because of $\mathcal{O}(2^n)$. Even a heuristic wrapper method like forward/backward selection takes weeks/months to finish. This issue gets even worse because unfortunately, the very limited amount of available data (only 105 CASP9 targets with matches, some of them with very few matches) prohibits that one can set aside some of the targets solely for the purpose of selecting features with them. Therefore a k-fold cross-validation would need k feature selections, each of them wrapped around a $k-1$-fold cross-validation. This is impractical, even for small k and especially for leave-one-out validation.

In this thesis, minimum-redundancy-maximum-relevance (MRMR) feature selection [Peng et al., 2005] and SVM based recursive feature elimination (SVM-RFE) [Guyon et al., 2002] were tested. MRMR is a filter method that iteratively adds features that correlate best with the label *and* are least redundand to already added features. SVM-RFE interatively trains an SVM and removes the features with the lowest weights[1]. Both

[1] even though SVM-RFE is a wrapper method by definition, it does not directly optimize the

algoritms can be configured to select a certain number of features. The features selected by MRMR consistently performed slightly better than SVM-RFE in a downstream cross-validation.

6.3 Choosing a classifier

The following variables are tested for building block classification.

- Learning algorithm: I tested Support Vector Machines (SVMs; implementation by Rüping [2004]), AutoMLP [Breuel and Shafait, 2010] – a neural network learner with automatic learning rate and size adjustment – and k-nearest-neighbor classification. Parameter optimization showed that SVMs work best on the given dataset with a "dot" (inner product) kernel and the C (complexity) parameter set to 0.01 – 0.1; kNN works best with k between 4 and 7 using "weighted vote".
- Number of features: I tested 1, 5, 10, 15, 20 and *all* features selected by MRMR.
- RMSD threshold: The data is way to noisy to directly predict the RMSD as a floating point value, but it is interesting to know how well the binary classification performs across different RMSD thresholds. I tested threshold between 1 Å and 3.5 Å in 0.25 Å steps.

The performance is measured in classification accuracy, sensitivity and specificity.

accuracy

$$\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}}$$

sensitivity

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

specificity

$$\frac{\mathrm{TN}}{\mathrm{FP} + \mathrm{FN}}$$

TP (true positive) a correct match that was classified as a correct match

FP (false positive) an incorrect match that was classified as a correct match

classification performance via cross-validation and therefore has time characteristics that are typically associated with filter methods

TN (true negative) an incorrect match that was classified as an incorrect match

FN (false negative) a correct match that was classified as an incorrect match

These measures are calculated using a 4-fold cross-validation with around 500 samples per batch. Across all RMSD thresholds, a total of 3200 distinct samples is used. Note that the feature selection happens independently for each training set. The graphs show that an RMSD threshold of 2.75 Å can be classified best. It appears that all learners do well, although SVMs and neural networks perform significantly better. The difference between SVMs and neural networks is small; for 2.75 Å, SVM has better sensitivity while neural networks have better specificity. Interestingly, the SVM does not work at all for the unfiltered features (all examples are classified as negatives; not shown in graphs).

If only a single feature is selected, usually seq_evalues_min or seq_probab_max are chosen. This is interesting because this means that for two fragments, the better matching fragment is more significant than the worse matching fragment. Of course this probably depends on the Evalue thresholds that were used for the training set and might not necessarily hold true when thresholds are chosen more daringly. When 5 features are selected, typically both of these together with 3 non-HHsearch features are selected. More details about well-performing features can be found in section 7.2. It seems that the specificity for 10 or more features stays roughly the same. The sensitivity stays roughly the same for 15 features or more. All classifiers perform better if the features are normalized with the Z-transformation (comparison not shown).

In conclusion, it seems that a reasonable combination would be an SVM learner using the top 15 normalized features selected by MRMR.

For readability sake, the following graphs show only a representative subset of the tested combinations.

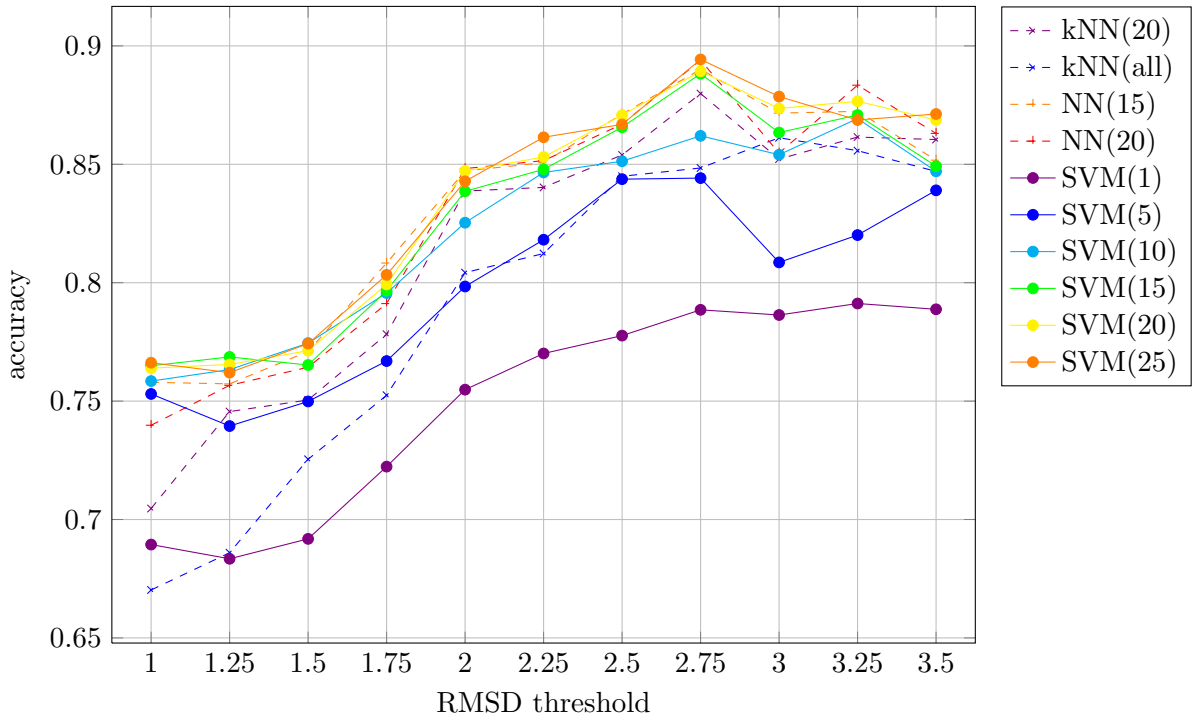


Figure 6.1: Classification *accuracy* across different RMSD thresholds determined by 4-fold cross-validation. Each line represents a different number of features + learner combination.

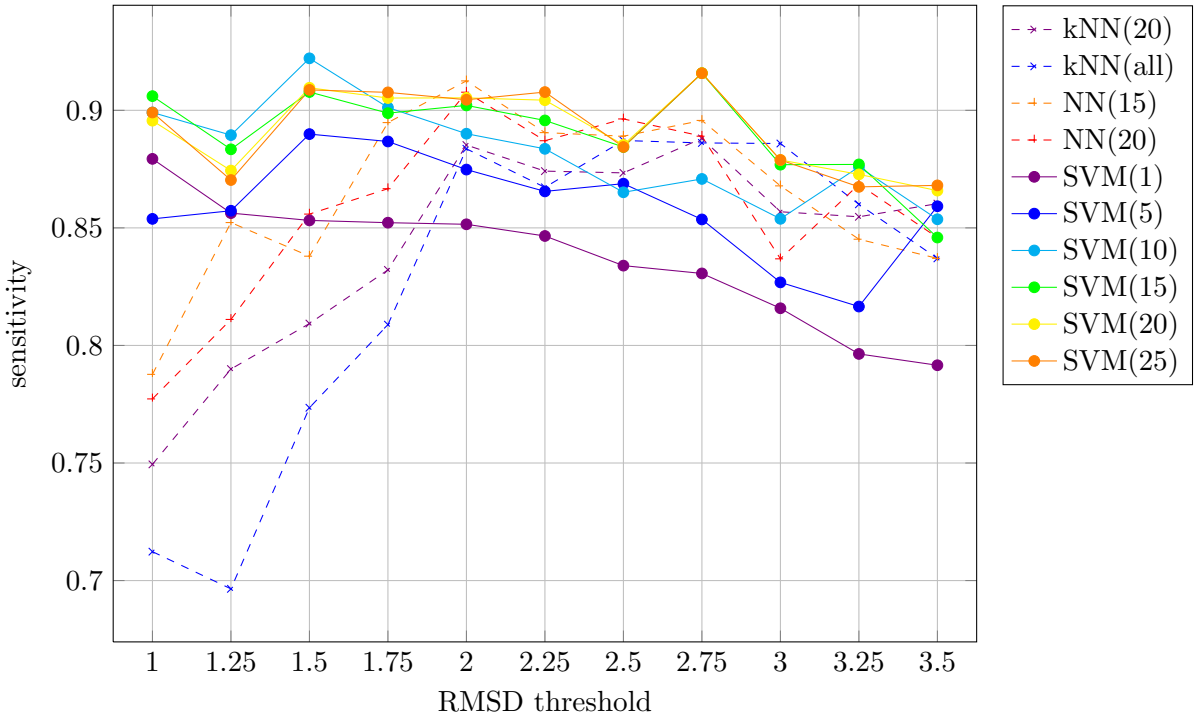


Figure 6.2: Classification *sensitivity* across different RMSD thresholds determined by 4-fold cross-validation. Each line represents a different number of features + learner combination.

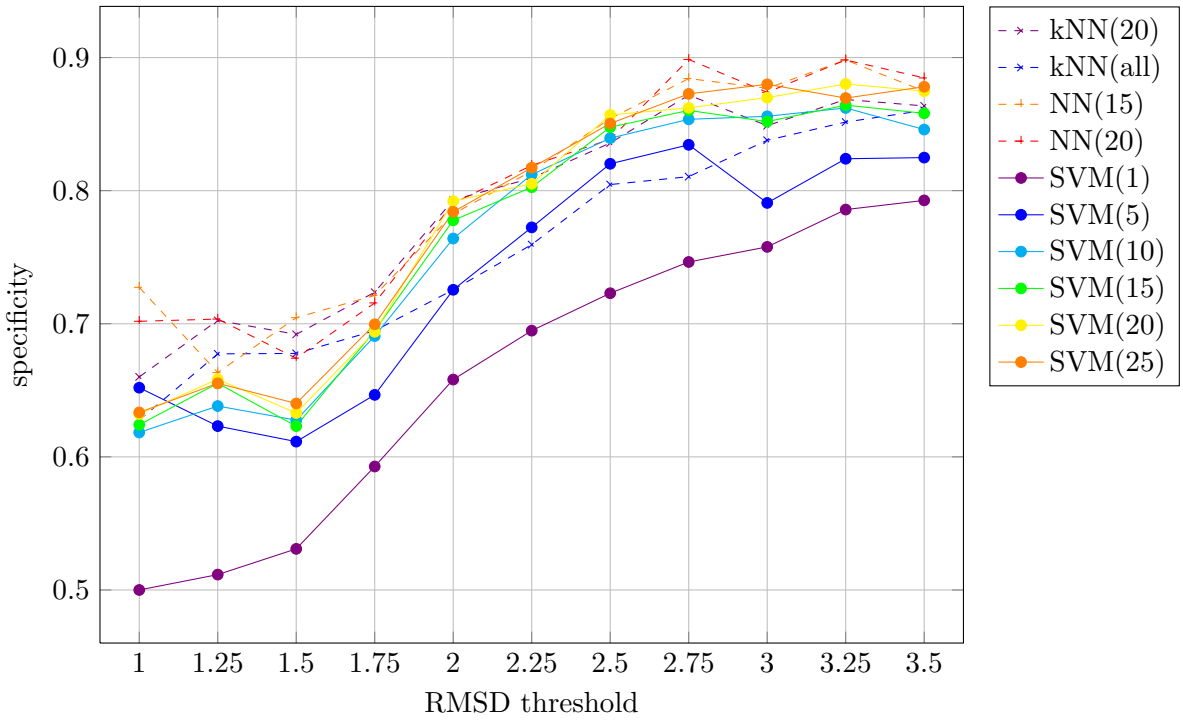


Figure 6.3: Classification *specificity* across different RMSD thresholds determined by 4-fold cross-validation. Each line represents a different number of features + learner combination.

7

Results

7.1 Classification performance

To evaluate the actual contribution that this thesis would do to the CASP pipeline, the accuracy/sensitivity/specificity graphs are not meaningful enough. To understand why, let's look at the following extreme examples: On the one hand there is a target with a handful of non-overlapping building block examples. In this case, a wrong classification of any of the building block matches would have a direct impact on the prediction quality, because a false positive would result in incorrect constraints; a false negative would result in "wasted" constraints. On the other hand there is a target with hundreds of matches – of course overlapping each other to a large degree –, most of them correct matches. In this case, a wrong classification is less likely to have a negative effect on the prediction because only a subset of the matches can be selected so that it is mutually compatible, anyway.

So, in order to determine this thesis' contribution in a meaningful way, one would have to do an actual assembly, i.e. cover the target sequence as optimally as possible with building blocks that where classified as positives. Sadly an optimal assembly algorithm is not yet available at this point. Even if it were, it could potentially be so complex on its own that it adds a lot of "noise" to the measurement. Therefore, I have developed two very simplified assembly models that hopefully enable a more realistic examination of the classifiers' contribution.

For both models, different algorithms that are able to rank matches are tested:

random ranks matches at random,

naive ranks matches by seq_evalues_min ascending (c.f. section 7.2),

prediction (no bail) ranks matches by their confidence[1] of belonging to the positive class (descending),

prediction does the same but filters out confidences below 50 %,

hybrid if the *whole target protein* has not a single match with a confidence above 50 %, this algorithm falls back to "naive". Otherwise, it ranks matches by seq_evalues_min but filters out confidences below 50 %.

The performance of the ranking algorithms is measured residue-wise; the slightly unorthodox[2] confusion matrix looks like this:

TP (true positive) A residue where the selected match is correct.

FP (false positive) A residue where the selected match is incorrect.

TN (true negative) A residue where no match was selected and indeed none of the matches were correct.

FN (false negative) A residue where no match was selected but at least one of the matches were correct.

Note that $\Omega = \mathrm{TP} \cap \mathrm{FP} \cap \mathrm{TN} \cap \mathrm{FN}$ is the set of residues that are covered with at least one match (as opposed to all residues of the target protein).

The two assembly models are as follows:

residue wise selection For each residue, the highest ranking match is selected. This model has two mayor problems: For once, selected building blocks are not necessarily mutually compatible. The other problem is that only the "prediction" and the "hybrid" ranking algorithm can produce negatives with this model.

In order to make the differences between the ranking algorithms more visible, I filtered Ω so that it only contains residues that have at least one negative match. Because otherwise, due to the generally good quality of the retrieved matches,

[1] produced by the SVM implementation; derived from the distance of the sample to the SVM hyperplane

[2] note how residues that have both correct and incorrect matches are not inherently positive/negative but their ground truth is determined by the selected match

ranking algorithms that won't predict negatives have an advantage. On the other hand, this preprocessing favors ranking algorithms that *do* predict negatives. Therefore, the truth probably lies somewhere in the middle.

forward selection assembly Matches are selected iteratively: Initially, every match is a candidate. In each iteration, the best ranking match is selected from the list of candidates. Then, all matches that are incompatible with the current selection are removed from the list of candidates. A match is incompatible with the current selection if it overlaps any of the already selected matches sequentially, but not structurally (c.f. section 5.10). The algorithm terminates when the list of candidates is empty. For the calculation of the confusion matrix, only the first selected match per residue matters. Note how this model can elegantly reassemble more-than-two fragment matches, because all two-fragment subsets of the same match are of course compatible with each other.

This model solves/alleviates both of the previous model's problems; the random and the naive classifier can now produce unmatched residues because all matches at one particular residue might be incompatible with previously selected matches.

This is also key to understanding the disadvantage of this model: If any of the ranking algorithms selects a big incorrect match, this could mean that many sequentially overlapping correct matches are pruned (because they are most likely structurally incompatible with the incorrect match). While this is of course quite realistic (assuming that mutually compatible matches are a requirement!), it might add noise.

Classification performances using the different ranking algorithms and assembly models can be seen in table 7.1. Using the "forward selection assembly" model, the "hybrid" algorithm achieves better precision than the "naive" algorithm for 18 targets; it achieves a worse precision for only five targets. Upon closer inspection, four of these five targets only have a "naive" precision lower than 40 %, anyways. Only in one case, the precision goes down from 94.4 % to 85 %. Whereas 15 of the 18 targets with better performance go up to 100 %. So, in total, the "naive" algorithm achieves 100 % precision for 46 targets and the "hybrid" algorithm achieves it for 61 targets. Of course, the "hybrid" algorithm also results in less covered residues, but on average only 1.06 less true positive residues per target than the "naive" algorithm.

ranking \ model	residue wise unfiltered Ω	residue wise filtered Ω	forward selection assembly
random	69.2 %	45.8 %	63.0 % / 53.8 %
naive	72.0 %	52.1 %	67.5 % / 59.4 %
prediction (no bail)	71.3 %	51.4 %	65.0 % / 53.7 %
prediction	76.9 % / 80.2 %	65.7 % / 77.9 %	71.0 % / 63.3 %
hybrid	80.5 % / 79.7 %	70.9 % / 76.6 %	76.2 % / 65.8 %

Table 7.1: Average precision/accuracy across CASP9 targets. Note that *target proteins* that do not have any correct match are excluded because the classification doesn't matter for these anyway (not to be confused with *residues* that do not have a correct match).

Looking at targets where classification accuracy is poor, it is apparent that free modeling (FM) targets (according to a classification in Kinch et al. [2011]) are a mayor concern. In fact, any match on a FM target gets classified as incorrect. This might be because the training set consist of only ≈ 10 % FM targets and therefore both the feature selection and the classifier are geared towards template based modeling (TBM). From the five targets with worse precision (see above), one is a FM target, the other ones are TBM targets.

7.2 Best performing features

Apart from the the actual contribution in the project context, the bigger picture of this thesis is to present features that can classify building blocks. Even though the plots in chapter 5 show that some of the features seem to work well, the plots don't give much insight as to how they perform relative to each other. The plots especially don't show when two features are redundant, i.e. two features might both work well in classifying almost the same subset of samples.

The best performing features were determined by performing a leave-one-target-out cross-validation with an RMSD threshold of 2.7 Å. MRMR was configured to select 15 features for each training set. Due to the fact that each training set is slightly different, each training set can result in a slightly different set of features. Table 7.2 shows the most frequently selected features across all training sets (each training set spanning 104 targets, resulting in a total of ≈ 1700 samples).. As mentioned before, the

Feature name	Frequency in %
frags_reverse	100
witness_same_instance	100
seq_separation_12_ratio_align	100
seq_separation_diff	100
seq_probab_max	100
seq_probab_prod	100
seq_evalues_min	100
length_match_ratio	99
seq_ranks_max	92.4
bayesian2_numerator_I16	58.1
bayesian2_numerator_I14	56.2
bayesian2_numerator_T5	40
seq_probab_min	39
equivgroups_pdbs	23.8
sca_msa_size	21
no_of_witnesses	17.1
seq_ranks_min	13.3
num_scop_families	6.7
acc_max	3.8
hbonds_max	3.8
seq_separation_12_ratio	2.9
seq_evalues_max	2.9
equivgroups	2.9
num_scop_superfamilies	1.9
bayesian2_I16	1
frags_adjoining	1
num_structure_exceptions:max	1

Table 7.2: Best performing features as selected by MRMR.

Feature name	Frequency in %
seq_evalues_max	84.6
distance_max:avg	84.6
seq_probab_min	76.9
distance_max:max	69.2
distance_max:min	53.8
struct_compat_overlaps	53.8
seq_probab_prod	53.8
frags_reverse	53.8
seq_separation_stddev	38.5
is_largest_local	38.5
seq_evalues_min	38.5
sca_msa_size	30.8
hbonds_max	30.8
hbonds_avg	30.8
distance_min:min	30.8
seq_separation_diff	23.1
bayesian2_denominator_T5	23.1
frags_adjoining	23.1
struct_compat_compatibility	23.1
witness_rmsd_avg	23.1
seq_ranks_max	23.1

Table 7.3: Best performing *free modeling* features as selected by MRMR.

classifier performs well for TBM targets, but poorly for FM targets. Therefore another cross-validation was conducted, this time using only FM targets. Table 7.2 shows the most frequently selected features across all FM training sets. It shall be noted that the data for this test was sparse (each training set spanning only 12 targets, resulting in a total of only $\approx$ 200 samples). Furthermore, even with the feature selection and the classifier geared towards FM targets, precisions only went up to 8 % – 16 % for 4 of the targets.

7.3 Conclusions

- Many of the shown feature plots show some kind of relationship between the feature and the RMSD, but all of them have a large degree of noise.

- The proposed classifier provides a significant improvement in both of the assembly models and "raw" classification accuracy/sensitivity/specificity. In consequence, the classifier is a valuable contribution no matter if the structure prediction requires mutually compatible constraints or not.

- The classifier only performs well for TBM targets. Free modeling targets can't be classified with reasonable results at this point. For the tested FM targets, the proposed "hybrid" ranking algorithm would not have had a negative impact, because it falls back to "naive" ranking if no matches are classified as correct. Due to the fact that CASP10 will most likely have only a small portion of FM targets, it seems reasonable to use *all* CASP9 targets as a training set for the CASP10 run.

- Our setup for CASP10 calculates 12 top performing features for each building block match. Using CAPS9 targets as the training set, the matches are classified as correct and incorrect with an SVM learner. The assembly modules currently lack a method of generating mutually compatible constraints. Therefore, forward selection assembly (with "hybrid" ranking) is used in the CASP pipeline.

- The best performing features across *all* targets contain a lot of features that are essentially modeling the number of witnesses of some sorts (such as no_of_witnesses, sca_msa_size, bayesian2_numerator_*[mt]* and equivgroups_pdbs). For FM targets, these features play little to no role. A similar pattern can be observed for sequence separation based features, witness_same_instance and frags_match_ratio. All of these differences intuitively make sense, because they bias towards matches that come from very similar templates.

- For FM targets, some of the more sophisticated features get some attention, for example distance_max:*[agg]* and struct_compat_overlaps.

- For TBM targets, it seems to be a good tactic to use the alignment "E-value"/"Probability" of the *better* of two fragments. This is somewhat surprising because conservatively one would think that a multi-fragment alignment is only as good as its worst fragment. For FM targets, this conservative assumption is indeed more appropriate. This discrepancy is probably due to TBM targets generally yielding higher scoring alignments and co-occurrence of fragments playing an increased role compared to FM targets.

- The only features that rank among the best features for both FM and TBM are frags_reverse and the ones derived from HHsearch's scores.

- The SCOP based features (section 5.7) perform worse compared to other features that model the number of witnesses. Apparently, the generalizability of a building block plays a secondary role (especially for TBM). Even though top_scop_families has a promising plot it is probably redundant to other features.

- SCA (statistical coupling analysis) based features (section 5.9) and hydrophobicity patterns (section 5.6) do not work well because they are both noisy. Furthermore, SCA based features are only defined for a fraction of the building blocks.

- Low solvent accessibility (section 5.6) i.e. interiority of a building block seems to be correlated with the RMSD, but plays only a minor role compared to other features.

- The size of the fragments plays no significant role even though bigger fragments are more statistically significant.

- The sequence clustering based features (section 5.11) work fairly well. Interestingly, the the feature selection chose the numerators more often than the fraction itself. The numerators are essentially just another variation of number_of_witnesses. There is not definitive answer as to which clustering method/threshold works best. In fact, sometimes the same feature with a different clustering method/threshold is chosen twice in the top 15 features. This is particularly surprising because one would expect those features to be redundant and MRMR tries to keep feature redundancy to a minimum.

- Witness density features (section 5.4) do not play much of a role. Together with the good performance of features that model the number of witnesses it can be said that the building block grouping strategy discussed in section 2.3 seems to be appropriate *for the current database size*. However, sequence profiles of building blocks can neither be used for retrieval nor as features (section 5.2) because they are too noisy.

- Structural compatibility (section 5.10) features are not worthwhile the way they are currently calculated. Figure 5.33 even has a very unintuitive plot.

- Target independent features like fragment length, spatial features, SCA based features, is_largest_local, witness_density, no_of_witnesses perform indeed worse than target dependent features (c.f. section 4.2). The only exception to this rule is distance_max:*[agg]* for FM targets.

7.4 Future work

- The current way to achieve a balanced training set is naive under-sampling. Even though the under-sampling happens according to certain constraints (see section 6.2), there are many more sophisticated tactics in the literature of how to deal with highly skewed training data. It would be interesting to try for example the "Neighborhood Cleaning Rule" [Laurikkala, 2001]; this tactic removes majority samples that lead to a misclassification of minority samples by looking at the three nearest neighbors of each sample.

- The current dataset only includes CASP9 targets. As that the retrieval does not yield any samples for some of the targets, only 105 targets remain for experimentation. Only 12 of them are FM targets which not enough to draw any conclusions. The next step would be to include CASP8 targets or any other targets that are not included in the PDBSS.

- A bigger dataset would also have the advantage of being able to set aside a part of the training set solely for a feature selection that is wrapped around a cross-validation. One could then filter the features first with MRMR and run a forward feature selection on the remaining ones.

- Currently, when none of the samples for one particular target are classified as correct, the tactic is to fall back to ranking samples by E-value (see "hybrid" ranking, section 7.1). Another tactic would be to subsequently increase the predicted RMSD threshold until some matches are classified as correct.

- There are certain TBM targets that work perfectly and others that do not work at all. It would be interesting to see if the working/failing targets have something in common. Apart from visually inspecting their structure and the building block matches, one could color the samples in the feature plots depending on which target a sample belongs to. This could expose features that work only for a subset of the targets and are noisy for others.

- Many of the decisions – especially in section 7.1 – are based on the hypothesis that 100 % precision is more important than coverage of as many residues as possible with building block matches. This is based on the assumption that the downstream energy minimization would be significantly disturbed by wrong or mutually incompatible constraints (c.f. section 1.4). A suitable experiment to test this hypothesis would be to do protein structure predictions for many or all of the CASP9 targets; for each target one would make three prediction runs: each time either using the unfiltered building block matches, the filtered building block matches or only the matches that remain after forward selection assembly.

- As mentioned in section 5.3, there is a bug in the building block database generation process that prevent certain types of building block from ending up in the database. Even frags_reverse should theoretically not be dependent on this bug, it would be interesting to see whether it still remains one of the best performing features after this bug is fixed. It would also be interesting to investigate how circular permutated proteins and frags_reverse interact.

References

Stephen Altschul. The statistics of sequence similarity scores, 1999. URL http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html. 10

Christian B. Anfinsen. Principles that Govern the Folding of Protein Chains. *Science*, 181(4096):223–230, July 1973. ISSN 0036-8075. URL http://dx.doi.org/10.1126/science.181.4096.223. 1

Gail J. Bartlett and William R. Taylor. Using scores derived from statistical coupling analysis to distinguish correct and incorrect folds in de-novo protein structure prediction. *Proteins: Structure, Function, and Bioinformatics*, 71(2):950–959, 2008. ISSN 1097-0134. URL http://dx.doi.org/10.1002/prot.21779. 12, 62

Spencer Bliven and Andreas Prlić. Circular permutation in proteins. *PLoS computational biology*, 8(3):e1002445, 03 2012. URL http://dx.doi.org/10.1371/journal.pcbi.1002445. 46

James U. Bowie, Neil D. Clarke, Carl O. Pabo, and Robert T. Sauer. Identification of protein folds: Matching hydrophobicity patterns of sequence sets with solvent accessibility patterns of known structures. *Proteins: Structure, Function, and Bioinformatics*, 7(3):257–264, 1990. ISSN 1097-0134. URL http://dx.doi.org/10.1002/prot.340070307. 55

Thomas Breuel and Faisal Shafait. Automlp: Simple, effective, fully automated learning rate and size adjustment. In *The Learning Workshop*, 4 2010. 74

J. M. Bujnicki. Protein-structure prediction by recombination of fragments. *Chembiochem*, 7(1):19–27, January 2006. ISSN 1439-4227. URL http://dx.doi.org/10.1002/cbic.200500235. 4

REFERENCES

Christopher Bystroff and David Baker. Prediction of local structure in proteins using a library of sequence-structure motifs. *Journal of Molecular Biology*, 281(3):565 – 577, 1998. ISSN 0022-2836. URL http://dx.doi.org/10.1006/jmbi.1998.1943. 12

C. Chothia and A. M. Lesk. The relation between the divergence of sequence and structure in proteins. *The EMBO journal*, 5(4):823–826, April 1986. ISSN 0261-4189. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1166865/. 2

B.A. Cunningham, J.J. Hemperly, T.P. Hopp, and G.M. Edelman. Favin versus concanavalin a: Circularly permuted amino acid sequences. *Proc Natl Acad Sci U S A*, 76(7):3218–3222, 1979. 46

Pawel Daniluk and Bogdan Lesyng. A novel method to compare protein structures using local descriptors. *BMC Bioinformatics*, 12(1):344+, 2011. ISSN 1471-2105. URL http://dx.doi.org/10.1186/1471-2105-12-344. 5

M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. *In Atlas of Protein Sequences and Structure*, 5:345–352, 1978. 10

Russell J. Dickson, Lindi M. Wahl, Andrew D. Fernandes, and Gregory B. Gloor. Identifying and Seeing beyond Multiple Sequence Alignment Errors Using Intra-Molecular Protein Covariation. *PLoS ONE*, 5(6):e11082, June 2010. URL http://dx.doi.org/10.1371/journal.pone.0011082. 62

Narcis Fernandez-Fuentes, Joseph M. Dybas, and Andras Fiser. Structural characteristics of novel protein folds. *PLoS computational biology*, 6(4):e1000750+, April 2010. ISSN 1553-7358. URL http://dx.doi.org/10.1371/journal.pcbi.1000750. 51

Xin Gao, Jinbo Xu, Shuai Cheng Li, and Ming Li. Predicting local quality of a sequence-structure alignment. *J. Bioinformatics and Computational Biology*, 7(5): 789–810, 2009. URL http://dx.doi.org/10.1142/S0219720009004345. 13, 36, 71

Mitiko Go and Sanzo Miyazawa. Relationshio between mutability, polarity and exteriority of amino acide residues in protein evolution. *International Journal of Peptide and Protein Research*, 15(3):211–224, 1980. ISSN 1399-3011. URL http://dx.doi.org/10.1111/j.1399-3011.1980.tb02570.x. 55

Stephan Günnemann, Ines Färber, Brigitte Boden, and Thomas Seidl. Subspace Clustering Meets Dense Subgraph Mining: A Synthesis of Two Paradigms. In *IEEE International Conference on Data Mining*, pages 845–850, 2010. URL `http://dx.doi.org/10.1109/ICDM.2010.95`. 23

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1-3):389–422, March 2002. ISSN 0885-6125. doi: 10.1023/A:1012487302797. 73

M. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, University of Waikato, 1999. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.4643`. 31

S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, November 1992. ISSN 0027-8424. URL `http://dx.doi.org/10.1073/pnas.89.22.10915`. 10

Enoch S. Huang, Ram Samudrala, and Jay W. Ponder. Ab initio fold prediction of small helical proteins using distance geometry and knowledge-based scoring functions. *Journal of Molecular Biology*, 290(1):267 – 281, 1999. ISSN 0022-2836. URL `http://dx.doi.org/10.1006/jmbi.1999.2861`. 13

Torgeir R. Hvidsten, Andriy Kryshtafovych, and Krzysztof Fidelis. Local descriptors of protein structure: A systematic analysis of the sequence-structure relationship in proteins using short- and long-range interactions. *Proteins: Structure, Function, and Bioinformatics*, 75(4):870–884, 2009. ISSN 1097-0134. URL `http://dx.doi.org/10.1002/prot.22296`. 5

W. Kabsch and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, December 1983. ISSN 0006-3525. URL `http://dx.doi.org/10.1002/bip.360221211`. 3, 45, 55

Ilona Kifer, Ruth Nussinov, and Haim J. Wolfson. Protein structure prediction using a docking-based hierarchical folding scheme. *Proteins: Structure, Function, and Bioinformatics*, 79(6):1759–1773, 2011. ISSN 1097-0134. URL `http://dx.doi.org/10.1002/prot.22999`. 5

REFERENCES

Lisa N. Kinch, Shuoyong Shi, Hua Cheng, Qian Cong, Jimin Pei, Valerio Mariani, Torsten Schwede, and Nick V. Grishin. Casp9 target classification. *Proteins: Structure, Function, and Bioinformatics*, 79(S10):21–36, 2011. ISSN 1097-0134. URL `http://dx.doi.org/10.1002/prot.23190`. 4, 82

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.4175`. 3

Rachel Kolodny, Patrice Koehl, Leonidas Guibas, and Michael Levitt. Small libraries of protein fragments model native protein structures accurately. *Journal of molecular biology*, 323(2):297–307, October 2002. ISSN 0022-2836. URL `http://view.ncbi.nlm.nih.gov/pubmed/12381322`. 2

Jorma Laurikkala. Improving identification of difficult small classes by balancing class distribution. In *Proceedings of the 8th Conference on AI in Medicine in Europe: Artificial Intelligence Medicine*, AIME '01, pages 63–66. Springer-Verlag, 2001. ISBN 3-540-42294-3. 87

Uta Lessel and Dietmar Schomburg. Similarities between protein 3-d structures. *Protein Engineering*, 7(10):1175–1187, 1994. URL `http://dx.doi.org/10.1093/protein/7.10.1175`. 3

Steve W. Lockless and Rama Ranganathan. Evolutionarily Conserved Pathways of Energetic Connectivity in Protein Families. *Science*, 286(5438):295–299, October 1999. ISSN 0036-8075. URL `http://dx.doi.org/10.1126/science.286.5438.295`. 60

Flavia Moser, Recep Colak, Arash Rafiey, and Martin Ester. Mining Cohesive Patterns from Graphs with Feature Vectors. In *SIAM International Conference on Data Mining*, pages 593–604, 2009. 23

Alexey G. Murzin, Steven E. Brenner, Tim Hubbard, and Cyrus Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247(4):536–540, April 1995. URL `http://dx.doi.org/10.1016/S0022-2836(05)80134-2`. 56

E. Neher. How frequent are correlated changes in families of protein sequences? *Proceedings of the National Academy of Sciences of the United States of America*, 91(1):98–102, January 1994. ISSN 0027-8424. URL http://dx.doi.org/10.1073/pnas.91.1.98. 62

Tams Nepusz, Rajkumar Sasidharan, and Alberto Paccanaro. Scps: a fast implementation of a spectral method for detecting protein families on a genome-wide scale. *BMC Bioinformatics*, 11:120, 2010. URL http://dx.doi.org/10.1186/1471-2105-11-120. 24

Britt H. Park, Enoch S. Huang, Michael Levitt, and Beckman Laboratories For. Factors affecting the ability of energy functions to discriminate correct from incorrect folds. *J. Mol. Biol*, 266:831–846, 1997. 13, 59

Florencio Pazos and Alfonso Valencia. Protein co-evolution, co-adaptation and interactions. *The EMBO Journal*, 27(20):2648–2655, September 2008. ISSN 1460-2075. URL http://dx.doi.org/10.1038/emboj.2008.189. 62

Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, August 2005. ISSN 0162-8828. URL http://dx.doi.org/10.1109/TPAMI.2005.159. 73

Stefan Rüping. mysvm - a support vector machine, 2004. URL http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/index.html. 74

M. I. Sadowski and D. T. Jones. Benchmarking template selection and model quality assessment for high-resolution comparative modeling. *Proteins: Structure, Function, and Bioinformatics*, 69(3):476–485, 2007. ISSN 1097-0134. URL http://dx.doi.org/10.1002/prot.21531. 11

M. Schiffer and A. B. Edmundson. Use of helical wheels to represent the structures of proteins and to identify segments with helical potential. *Biophys. J.*, 7:121–135, Mar 1967. 54

Kim T. Simons, Charles Kooperberg, Enoch Huang, and David Baker. Assembly of protein tertiary structures from fragments with similar local sequences using simulated

annealing and bayesian scoring functions. *Journal of Molecular Biology*, 268(1):209 – 225, 1997. ISSN 0022-2836. URL http://dx.doi.org/10.1006/jmbi.1997.0959. 13

Kim T. Simons, Ingo Ruczinski, Charles Kooperberg, Brian A. Fox, Chris Bystroff, and David Baker. Improved recognition of native-like protein structures using a combination of sequence-dependent and sequence-independent features of proteins. *Proteins*, 34:82–95, 1999. 12

Johannes Söding. Quick guide to HHsearch, nov 2006. URL ftp://ftp.tuebingen.mpg.de/pub/protevo/HHsearch/HHsearch1.5.01/HHsearch-guide.pdf. 40, 57

Gurol M. Suel, Steve W. Lockless, Mark A. Wall, and Rama Ranganathan. Evolutionarily conserved networks of residues mediate allosteric communication in proteins. *Nature Structural & Molecular Biology*, 10(1):59–69, December 2002. ISSN 1072-8368. URL http://dx.doi.org/10.1038/nsb881. 5, 62

Alex Tossi and Luca Sandri. Hydromcalc, 2001. URL http://www.bbcm.univ.trieste.it/~tossi/HydroCalc/HydroMCalc.html. 54

Stijn van Dongen. faqs and facts about the mcl cluster algorithm, 2010a. URL http://micans.org/mcl/man/mclfaq.html. 25

Stijn van Dongen. Work flows and protocols for mcl and friends, 2010b. URL http://micans.org/mcl/man/clmprotocols.html. 25

Kevin Y. Yip, Prianka Patel, Philip M. Kim, Donald M. Engelman, Drew McDermott, and Mark Gerstein. An integrated system for studying residue coevolution in proteins. *Bioinformatics*, 24(2):290–292, January 2008. URL http://dx.doi.org/10.1093/bioinformatics/btm584. 63

Yang Zhang and Jeffrey Skolnick. The protein structure prediction problem could be solved using the current pdb library. *Proceedings of the National Academy of Sciences of the United States of America*, 102(4):1029–1034, 2005. URL http://dx.doi.org/10.1073/pnas.0407152101. 4