Classifiers for Discrimination of Significant Protein Residues and Protein-Protein Interaction Using Concepts of Information Theory and Machine Learning

Dissertation
zur Erlangung des wissenschaftlichen Doktorgrades
"Dr. rerum naturalium"
an der Georg-August-Universität Göttingen

vorgelegt von
Roman Yorick Asper
aus Bielefeld

Göttingen, October 2011

Dekan: Professor Dr. Dieter Hogrefe
1. Referent: Professor Dr. Stephan Waack
2. Korreferent: Professor Dr. Carsten Damm

Tag der mündlichen Prüfung: 26.10.2011

Contents

List of Figures

Chapter 1

Introduction

The field of bioinformatics has been a major influence for our work in recent years, and after the conclusion of our project Colombo ([WM06]) which dealt with the prediction of genomic islands, we changed our focus to the field of protein analysis. In particular we focused on prediction of important residues in protein chains, especially protein-protein interaction. We thought our theoretical approach would yield as good results on this issue as it had done with the Colombo project.

1.1 The Problem of Protein-Analysis

The biggest problem of protein analysis is the massive amount of proteins which leads to an even greater amount of possible protein-protein interactions. With today's methods and technology, analysing a protein or an interaction of proteins is possible in a laboratory. This is an expensive and time consuming effort and thus it is not feasible for all possible protein combinations. Therefore, the research and analysis of proteins in laboratories is mostly restricted to specific projects, for example the creation of a new pharmaceutical agent. These projects are sometimes even more confined due to being built on the basis of already known proteins and only slight variations are tested.
The field of bioinformatics offers theoretic methods to predict the chances of success of experiments. These predictions can be used to determine whether an experiment should actually be performed in a laboratory or not. Though there has been quite a gain in knowledge in recent years about the inner workings of proteins and protein-protein interaction, the accuracy of these predictions is still problematic. A lot is still unknown, which makes it difficult to create accurate prediction models for the behaviour of proteins.

1.2 Designing a Theoretical Model for Predicting Important Residues

In the field of prediction of important residues, *prediction models* have become more common and are an important branch of research due to the numerous tasks where these models are of assistance, for example pharmaceutics, research of behaviour and evolution of cells.
For our field of interest, a prediction model for important residues consists of two major parts:

- The prediction algorithm itself, which actually decides, whether a residue is important or not, or in case of protein-protein interaction, if a residue is part of the interaction process.

- The data model upon that the predictions are based. This includes the classifiers which are distinctive characteristics of proteins and these are used to separate important residues from unimportant residues.

The data model is the part which has to be done first. We decided to use already existing classifiers and also to develop new ones that promise good results based on our expertise.

1.3 Developing Classifiers for Protein-Analysis

The first avenue was motivated by the fact that classifiers based on thermodynamical entropy generate good results, but only a few of those exist. We decided to use an approach with the Shannon entropy, which is used in the field of information theory and has only recently begun to be seen in the field of protein residue prediction. These two entropy concepts are similar yet have some distinctive differences. We felt that for our purposes the Shannon entropy is better suited as we want to use the level of the information in the proteins and not physical characteristics to classify them. Thus in our setting the Shannon entropy promises better results than the thermodynamical entropy. This approach focuses on identifying important residues and on creating a ranking of the residues.
The second avenue was a structural approach combined with machine learning which focused on detecting protein-protein interaction. Previous contributions to this problem by our working group provided good preliminary results ([Bro08]). We decided to built upon that concept and use PAC-learning to generate a classifier for protein-protein interaction detection. The classifier is based on structural differences between interaction sites and non-interaction sites on the proteins.
Modelling these two classifiers was the main contribution of this thesis.

1.4 Structure of the Thesis

The remainder of this thesis is structured as follows.
Chapter 2 gives a brief overview about what proteins are and how protein-protein interaction is defined.
Chapter 3 gives introductions to the mathematical foundations upon which our classifiers are based. This includes information theory and the Shannon entropy as well as the basics of machine learning and the PAC-learning algorithm used in this work.
In Chapter 3 the data we used as an input for our models is shown and how the data is preprocessed for our needs.
Chapter 4 describes the significant protein residue classifier based on entropy as well as the related mathematical model. It also includes a quality assessment and a comparison to related works.
Following in Chapter 5 is the classifier for protein-protein interaction based on structural patterns of the protein backbone. It also presents a conclusion about the quality of the method.
Chapter 6 is a brief comparison between the two classifiers and their respective results.
In Chapter 7 the thesis is concluded with an outlook for possible future project.

Chapter 2

Foundations of Protein-Analysis

This chapter provides a brief introduction to proteins, their main characteristics and an overview of the interaction between proteins. Additionally there is a brief summary about protein alignments and multiple sequence alignments (MSA) as these are important for the classifiers presented in Chapter 5 and Chapter 6.

2.1 Protein

A protein is basically a compound of one or more chains of amino acids which are connected through *peptide bonds*. The amino acids can be differentiated by their chemical and physical characteristics. Additionally the general occurrences of the amino acids differs from each other ([WM09]) for each protein chain. Peptide bonds are covalent chemical bonds between two molecules. When a peptid bond occurs between two amino acids a part of the amino acid is discarded, usually a water molecule. The remainder of the amino acid is called *residue*. For the remainder of this work we consider the terms amino acid and amino residue to be synonymous.
The chain is folded into a globular form depending on the interactions and bonds between the residues. Based on the form and the composition of the residues the chain can develop various functions.
For further use we declare:

Definition 2.1
$\mathfrak{A}_{set}$ is the set consisting of the 20 *proteinogenic* amino acids. Proteinogenic means protein building.

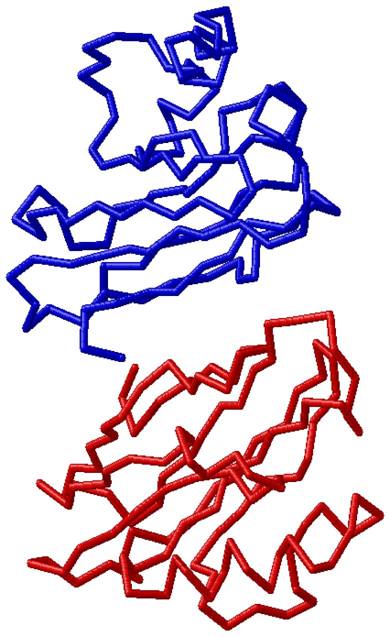


Figure 2.1: The backbone of the protein AMPA chains A and C

As mentioned before the protein forms a 3D structure through the aforementioned peptid bonds. Therefore, the structure of a protein is an important information, beside the chain of amino acids and the amino acid composition. The protein structure is divided into four distinct structure types.

- The *primary structure* is the sequence of the residues. It is also called the *backbone*. Figure 2.1 shows the backbone of the protein azurin mutant Phe114Ala from Pseudomonas aeruginosa (AMPA).
- The *secondary structure* is composed of repeating regular substructures of the backbone which are stabilised by hydrogen bonds. The most common substructures are the α-helix and the β-sheet. These can be seen in Figure 2.2 where the α-helices are red and the β-sheets are yellow. Again the protein AMPA is used as example.
- The *tertiary structure* is the spatial alignment of all atoms to each other.
- The *quaternary structure* is the alignment of several protein chains into one functional protein complex.

These structures are not rigid, but can shift fluidly depending on the function the protein has to perform. This makes it harder to predict the function and behaviour of a protein since it depends on the environment and the current circumstances.

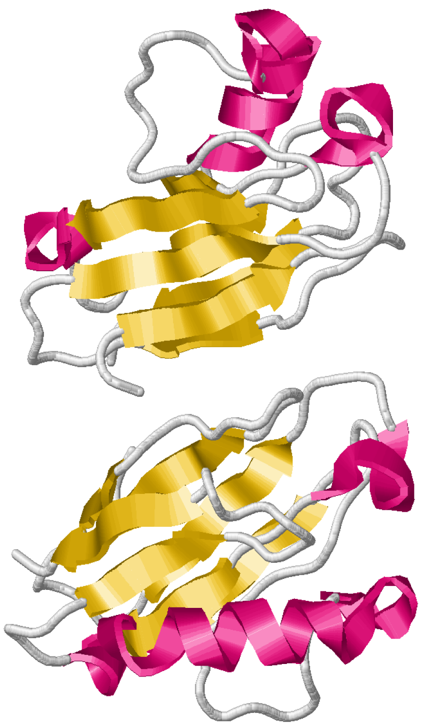


Figure 2.2: The secondary structure of the protein AMPA chains A and C, α-helices (red) and β-sheets (yellow)

Proteins are grouped into *families*. A family includes all proteins that are evolutionary related and they typically share a common ancestor. A protein family usually shares similar structures and functional components so-called *motifs* in the residue chain.
One example is the BCL-2 protein family which is responsible for the cell suicide process (apoptosis), a vital process for the organism. If a cell is damaged or infected, it gets a signal to dismantle itself which is regulated by the BCL-2 proteins ([AC98]).

2.1.1 Terminology

Two important terms which are also used in the course of this thesis are *Van-der-Waal radius* and *Angstrom*. The Van-der-Waal radius is the radius of an imaginary hard sphere around an atom and is used for spatial measurements between atoms.
Angstrom is a unit of length measuring 1 Angstrom = 10^{-10} meter and is denoted as Å.
Amino residues in the backbone are commonly written as a three letter code. For example tyrosine is encoded as TYR. A single letter coding also exists, which would encode tyrosine as Y.

2.2 Protein-Protein-Interaction

The definition of an interaction between two or more protein chains is that the chains have physical contact and form bonds between residues which leads to a protein complex.
A prominent example for these bonds are the *hydrogen bonds* [NT97]. Hydrogen bonds occur when a hydrogen atom bonds with an electronegative atom from another molecule, these are non covalent bonds and in general weaker than covalent bonds like for example the peptide bonds.
We assume a broader stance, which includes the functionality of proteins into the definition of an interaction. Interactions between proteins also means that they belong to the same *molecular machine*. These proteins do not necessarily have physical contact but have a functional contact. Molecular machines are for example responsible for coordinating the information flow on a cellular level and for the cell replicating process. Nearly all important functions of a cell are controlled or performed by protein-protein-interactions([WM09]).

2.2.1 Homo- and heterodimer

A dimer is a molecule which consists of two sub units, the monomers. There are two types of dimers; the homodimers, where the two sub units are identical monomers and the heterodimers where the monomers differ from each other.
If we apply this to proteins, a homodimer is a protein complex of two interacting residue chains, where the backbone of the chains is identical. Note that only the primary structure (the amino acid sequence) is identical, secondary and tertiary structure might differ.
Vice versa a heterodimer is a protein complex of two interacting residue chains where the backbones of the protein chains are different from each other.
In the following work we will use homodimer and heterodimer always as a denomination for a protein-protein compound.

2.2.2 Structural Regions of Proteins

A protein can be divided into 2 regions: the core and the surface.

Core and surface are calculated by a method that determines how accessible each residue is by a solvent, usually water, the solvent accessible surface (SAS) method ([LR71]). If the percentage of the solvent accessible area of a residue is below a certain threshold, the residue belongs to the core, otherwise it belongs to the surface. An example of the regions of a protein can be seen in in Figure 2.3. A subset of the surface is the interface. These are the residues on a chain which interact with other protein chains in a protein complex. The interface is defined as those residues that have a certain maximal distance to a residue from another chain in a protein complex. Depending on the experiments this distance is usually between 0.3 - 0.8 Å.

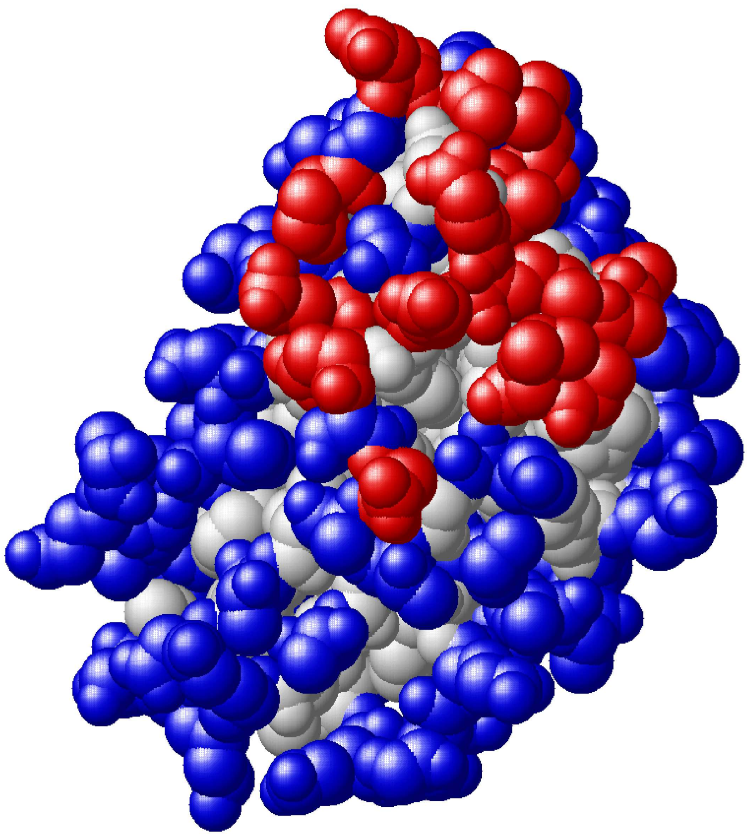


Figure 2.3: Regions of the protein AMPA, core = grey, surface = blue and interface = red (Chain A)

Hotspots

There exists no common definition of what hotspots are. The one most often used is that hotspots are the true binding sites of a protein-protein-interaction. In other words hotspots are the interface residues that are essential to the protein binding process. If these hotspots were mutated or cut out off the chain, the interaction is impeded or even completely blocked. Compared to the size of the interface the amount of hotspot residues is relatively small. Only about 5% of the interface residues are considered to be hotspots ([BT98]).
The standard method to identify hotspots experimentally is to mutate the residue in question into the amino acid alanine and test how this affects the binding of the protein chains ([MW01]). This is a tedious and expensive progress as each residue has to be tested individually. As a consequence, the database for true hotspots is very small and incomplete.

2.2.3 Current Methods for Analyzing

Since the analysis of proteins is an important field of research, several experimental methods, to analyse protein-protein-interactions, exist. We present one example of an experimental method.
Additionally, over the last years quite a few theoretical methods have been developed. An overview about the different theoretical concepts and methods will be given later in this section.

Experimental Method

One of the standard methods is the *co-immunoprecipitation* [Yac07].
Co-immunoprecipitation works on a solution that contains multiple proteins and contains at least one known protein. An antibody for this known protein is given into the solution where it binds with the known protein and is pulled out. If that known protein is part of a tightly bound protein complex, a chance exists that other members of that complex are pulled out as well, which can then be identified and analysed. This process can be repeated with antibodies for the newly identified proteins until the solution is completely analysed.
This method is not without flaws, for example the requirement of at least one known protein in the solution; not to mention that it is both expensive and time consuming.

Theoretical Methods

We follow the example of the work of Zhou et al. [ZQ07] who made an assessment about protein-protein interface predictors.
Zhou et al. divide the predictors into classes based on their prediction method and not on which protein characteristics they use. Protein characteristics used are for example the different distribution of amino residues for interfaces and their respective chemical characteristics.
All prediction methods have in common that they have to train over a data set of known protein-protein-interactions. The first division is made to differentiate between numerical methods and probabilistic methods.
Due to the diversity of methods used, combined with the diversity of available protein characteristics, a few combined approaches already exist. These "metamethods" show a good starting point for future endeavours ([dVB06], [SD04], [QZ07]).

Theoretical Methods: Numerical Let d_r be the data relevant for a residue r on a protein chain. Numerical methods then employ a function $F(d_r, c)$, where c are some coefficients which have been learned through the training.
The value of $F(d_r, c)$ then determines, if r is rated as an interface residue or not.

- *Linear Regression Methods*
  These methods model F as a linear function in $\mathbb{R}$ and employ a threshold t for the rating of r. This is a simple approach, but in general lacks in the performance of predictions ([KA07], [LC06]).

- *Scoring Function Methods*
  These functions are more complex and are based on empirical energy functions which are used to calculate the energy potential of molecules. These models have a better discrimination than the linear regression approach, but the model itself needs a much higher knowledge in physics to be transparent ([BJ06], [dVB06], [HB06], [LBT05], [LZ06], [MJ06]).

- *Support Vector Machine Methods*
  SVMs map the training data set as a vectors in $\mathbb{R}$ and then calculate a hyperplane, which has the best seperation for the interface and non-interface examples. Any new example is then checked against this hyperplane. The overall accuracy is better than with linear regression but the classification process is a lot less transparent ([BA05], [BW05], [CB06], [KT04], [RL05], [WH06], [WL06] [ZST^{+}11], [Zel11]).

- *Neural Network Methods*
  A neural network is structured in layers that consist of nodes. A standard variant has an input layer, an intermediate layer and an output layer. These nodes are connected through functions which represent dependencies of the system, these functions are learned in the training process.

  Similar to SVMs the accuracy of the neural networks is based on the cost of the transparency of the method ([OR07a], [OR03], [CZ05], [FC02], [PM07], [ZS01]).

Theoretical Methods: Probabilistic Again let d_r be the data relevant for a residue r on a protein chain. We assume that $d_r = d_{r_1}, .., d_{r_n}$ are the individual data of which d_r is compromised. Probabilistic methods calculate the conditional probability $p(R|d_r)$ for R being either interface or non-interface. Thus are two different distributions gained from the training data. For an unknown residue x it is determined if $p(R, x)$ fits better into the interface or non-interface distribution and thus x is ranked.

- *Naive Bayesian Methods*
  The naive bayes method assumes that all individual data d_{r_i} are independent of each other and thus calculates $p(R|d_r)$ accordingly ([NS04]).

- *Bayesian Network Methods*
  In this method the individual data are not necessarily assumed independent, if d_{r_i} and d_{r_j} are known to be dependant one each other they contribute to $p(R|d_r)$ with their joint probability $p(d_{r_i}, d_{r_j}|R)$. This method thus needs knowledge of the dependencies between the indivdiual data ([BW06]).

- *Hidden Markov Model Methods*
  A hidden markov model involves a chain of states and a chain of observations. Each state emits one observation, but only the observations are visible. In case of proteins the states would be interface or non-interface combined with the relevant data and the observations would be the residues. Therefore, during the training process the HMM method tries to match the real observation with an artificial observation made with guessed states. The best guessed states are then used to rate unknown residues ([FM06]).

- *Conditional Random Field Methods*
  Conditional random fields (CRFs) are a probability framework for labeling and segmenting structured data, such as sequences, trees and lattices. The underlying idea is that of defining a conditional probability distribution over label sequences given a particular observation sequence, rather than a joint distribution over both label and observation sequences. Definition taken from [Wal04] ([LL07]).

2.3 Coevolution and Alignments

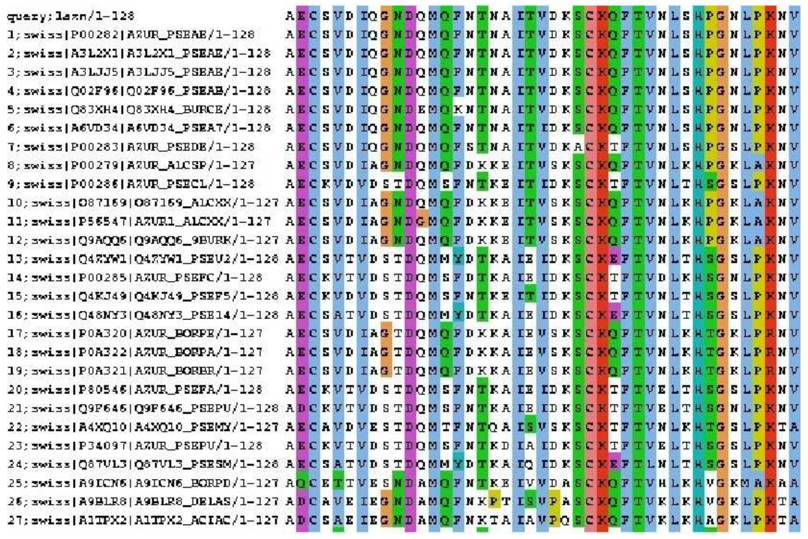
An important part of protein research is the comparison of proteins. The differences and similarities are evaluated to draw conclusions about the structural, functional, and evolutionary relation between two protein chains.
In this area the notion of *coevolution* is a very important concept. Coevolution means that two biological objects are so closely connected, that, if one evolves, the other has to evolve as well. In proteins this is described as *correlated mutations* where we have a pair of amino acids and, if one changes, the other has to change as well due to the selective pressure these amino acids have on each other.

2.3.1 Sequence Alignments

By aligning two different protein chains we hope to determine how related these two proteins are. We often encounter a protein chain that is slightly evolved or mutated to fit evolutionary needs of a molecular structure. Thus it has effectively become a new protein, but still retains most functionality and structure of the original protein. An alignment of these two sequences can show exactly which parts of the protein have been mutated and which parts have been retained. If we align a known protein and an unknown protein, the comparison can give some indication of the functionality of the unknown protein if similarities between the two proteins show up.
There are optimal algorithms to align two protein chains with each other like the Needleman-Wunsch algorithm [NW70], and the Smith-Waterman algorithm [SW81]. These two algorithms have a run time of $O(n^2)$ which is rather high, considering that these algorithms are commonly used to compare one protein to a large database of protein chains to find a good match. Heuristics have been developed to address this run time problem, like the BLAST algorithm [AL90].
If one protein is shorter than the other, the shorter protein has to be lengthened to be accurately matched with the longer protein. This is done by inserting gaps in the shorter protein chain.

2.3.2 Multiple Sequence Alignments

If it is possible to align two protein chains with each other, we can also align multiple sequences with each other.
These multiple sequence alignments (MSAs) are typically used to align a protein family. The MSA can then be evaluated to identify regions that have been conserved throughout the whole protein family or that show a traceable sequence of correlated mutations. Figure 2.4 shows a segment of a MSA of the protein azurin mutant Phe114Ala from Pseudomonas Aeruginosa (AMPA).



```
query;1azn/1-128                         AECSVDIQGNDQMQFNTNAITVDKSCKQFTVNLSHPGNLPKNV
1;swiss|P00282|AZUR_PSEAE/1-128          AECSVDIQGNDQMQFNTNAITVDKSCKQFTVNLSHPGNLPKNV
2;swiss|A3L2X1|A3L2X1_PSEAE/1-128        AECSVDIQGNDQMQFNTNAITVDKSCKQFTVNLSHPGNLPKNV
3;swiss|A3LJJ5|A3LJJ5_PSEAE/1-128        AECSVDIQGNDQMQFNTNAITVDKSCKQFTVNLSHPGNLPKNV
4;swiss|Q02F96|Q02F96_PSEAB/1-128        AECSVDIQGNDQMQFNTNAITVDKSCKQFTVNLSHPGNLPKNV
5;swiss|Q83XH4|Q83XH4_BURCE/1-128        AECSVDIQGNDEMQKNTNAITVDKSCKQFTVNLSHPGNLPKNV
6;swiss|A6VD34|A6VD34_PSEA7/1-128        AECSVDIQGNDQMQFNTNAITIDKSCKQFTVNLSHPGNLPKNV
7;swiss|P00283|AZUR_PSEDE/1-128          AECSVDIQGNDQMQFSTNAITVDKACKTFTVNLSHPGSLPKNV
8;swiss|P00279|AZUR_ALCSP/1-127          AECSVDIAGNDQMQFDKKEITVSKSCKQFTVNLKHPGKLAKNV
9;swiss|P00286|AZUR_PSECL/1-128          AECKVDVDSTDQMSFNTKEITIDKSCKTFTVNLTHSGSLPKNV
10;swiss|O87169|O87169_ALCXX/1-127       AECSVDIAGNDQMQFDKKEITVSKSCKQFTVNLKHPGKLAKNV
11;swiss|P56547|AZUR1_ALCXX/1-127        AECSVDIAGNDGMQFDKKEITVSKSCKQFTVNLKHPGKLAKNV
12;swiss|Q9AQQ6|Q9AQQ6_9BURK/1-127       AECSVDIAGNDQMQFDKKEITVSKSCKQFTVNLKHPGKLAKNV
13;swiss|Q4ZYW1|Q4ZYW1_PSEU2/1-128       AECSVTVDSTDQMMYDTKAIEIDKSCKEFTVNLTHSGSLPKNV
14;swiss|P00285|AZUR_PSEFC/1-128         AECKVTVDSTDQMSFDTKAIEIDKSCKTFTVDLKHSGNLPKNV
15;swiss|Q4KJ49|Q4KJ49_PSEF5/1-128       AECKVDVDSTDQMSFNTKEITIDKSCKTFTVNLTHSGSLPKNV
16;swiss|Q48NY3|Q48NY3_PSE14/1-128       AECSATVDSTDQMMYDTKAIEIDKSCKEFTVNLTHSGSLPKNV
17;swiss|P0A320|AZUR_BORPE/1-127         AECSVDIAGTDQMQFDKKAIEVSKSCKQFTVNLKHTGKLPRNV
18;swiss|P0A322|AZUR_BORPA/1-127         AECSVDIAGTDQMQFDKKAIEVSKSCKQFTVNLKHTGKLPRNV
19;swiss|P0A321|AZUR_BORBR/1-127         AECSVDIAGTDQMQFDKKAIEVSKSCKQFTVNLKHTGKLPRNV
20;swiss|P80546|AZUR_PSEFA/1-128         AECKVTVDSTDQMSFNTKAIEIDKSCKTFTVELTHSGSLPKNV
21;swiss|Q9F646|Q9F646_PSEPU/1-128       ADCKVTVDSTDQMSFNTKAIEIDKSCKTFTVELTHSGNLPKNV
22;swiss|A4XQ10|A4XQ10_PSEMY/1-127       AECAVDVESTDQMTFNTQAISVSKSCKTFTVNLKHTGSLPKTA
23;swiss|P34097|AZUR_PSEPU/1-128         AECKVTVDSTDQMSFNTKDIAIDKSCKTFTVELTHSGSLPKNV
24;swiss|Q87VL3|Q87VL3_PSESM/1-128       AECSATVDSTDQMMYDTKAIQIDKSCKEFTLNLTHSGSLPKNV
25;swiss|A9ICN6|A9ICN6_BORPD/1-127       AQCETTVESNDAMQFNTKEIVVDASCKQFTVHLKHVGKMAKAA
26;swiss|A9BLR8|A9BLR8_DELAS/1-127       ADCAVEIEGNDAMQFNKPTISVPASCKQFTVKLKHVGKLPKTA
27;swiss|A1TPX2|A1TPX2_ACIAC/1-127       ADCSAEIEGNDAMQFNKTAIAVPQSCKQFTVKLKHAGKLPKTA
```

Figure 2.4: A segment of a possible MSA of the protein AMPA chain A

Opposed to the sequence alignment of a single pair of protein chains there exists no algorithm which can calculate an optimal solution for aligning multiple sequences. Similar to the pair alignment, gaps are used to prolongate short proteins.
Several heuristics exist, for example, the Mafft algorithm, and the ClustalW algorithm [TW06]. The results of the heuristic algorithms depend on the choice of the algorithm and the choice of the protein database. Due to these factors there can be a variance in the resulting MSAs.

Chapter 3

Foundations for Applied Mathematical Concepts

3.1 Information Theory

Information theory provides quantifications of uncertainty in predictions of the value of random variables, as well as measurements for the information of random variable distributions. Claude E. Shannon developed this branch of applied mathematics to calculate the boundaries of data compression and transmission. Although this implies that information theory is merely a subset of communication theory, it intersects with several other fields, for example physics, computer science, mathematics, which make information theory much more than a simple subset.
This section gives an overview about the most important definitions and theorems later used in our work. As this is only an overview, we omit the proofs of the theorems; all proofs, as well as a detailed introduction to information theory can be found in [CT91].

We use the convention

$$0 \cdot \log(0) := 0$$

which is justified by

$$\frac{1}{n} \cdot \log(\frac{1}{n}) \to 0 \text{ for } n \to \infty$$

In addition, $log(x)$ always means $log_2(x)$ throughout this work.

The *entropy* of a discrete random variable X with alphabet $\mathfrak{X}$ is a measure of uncertainty, when predicting its value.

Definition 3.1
Let X be a discrete random variable over the alphabet $\mathfrak{X}$, and $p(x)$ the probability mass function with $p(x) = Pr(X = x), x \in \mathfrak{X}$. The *entropy* $H(X)$ is defined as

$$H(X) = -\sum_{x \in \mathfrak{X}} p(x) \log p(x).$$

The entropy over the alphabet $\mathfrak{X}$ is maximized, if $p(x)$ is the uniform distribution.

Theorem 3.2
Let $\mathfrak{X}$ be the alphabet and $|\mathfrak{X}|$ be the number of elements over which X is distributed, then $H(X) \leq \log(|\mathfrak{X}|)$ with equality, if and only if X is uniformly distributed over $|\mathfrak{X}|$.

We have a definition for the entropy of a single random variable, which is not sufficient for our research about proteins, because we need to be able to compare pairs of residues. Therefore we extend the entropy definition to a pair of discrete random variables X, Y over the alphabet $\mathfrak{X}$.

Definition 3.3
Let X, Y be a pair of discrete random variables with joint probability mass function $p(x, y)$, the joint entropy $H(X, Y)$ is defined as

$$H(X,Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log p(x,y)$$

In addition to the joint entropy, we also need an entropy definition of a random variable Y given another variable X.

Definition 3.4
Let X, Y be a pair of discrete random variables with conditional probability mass function $p(y|x)$, the conditional entropy $H(Y|X)$ is defined as

$$\begin{aligned} H(Y|X) &= \sum_{x \in \mathcal{X}} p(x) H(Y|X = x) \\ &= -\sum_{x \in \mathcal{X}} p(x) \sum_{y \in \mathcal{Y}} p(y|x) \log p(y|x) \\ &= -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log p(x|y) \end{aligned}$$

Entropy, joint entropy, and conditional entropy are closely connected, which is shown in the following theorem.

Theorem 3.5

$$H(X,Y) = H(X) + H(Y|X)$$

The most important concept from information theory for our work is the *mutual information.* It is a measure for the amount of information random variables X, Y contain about each other.

Definition 3.6
Let X, Y be a pair of discrete random variables with joint probability mass function $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$, the mutual information $MI(X;Y)$ is defined as

$$MI(X;Y) = \sum_{x \in \mathfrak{X}} \sum_{y \in \mathfrak{Y}} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

With our knowledge of Y we can reduce the uncertainty in X by the mutual information $MI(X;Y)$.
Mutual information can be rearranged by

$$MI(X;Y) = H(X) + H(Y) - H(X,Y) \tag{3.1}$$

This relationship between $MI(X;Y)$, $H(X)$, $H(Y)$, $H(X;Y)$ and $H(Y|X)$ is best demonstrated by a Venn diagram presented in Figure 3.1. As can be seen, the mutual information is the intersection of $H(X)$ and $H(Y)$.
The mutual information is a special case of the relative entropy or *Kullback Leibler distance.*

Definition 3.7
The relative entropy between two probability mass functions p(x) and q(y) is defined as

$$D(p\|q) = \sum_{x \in \mathfrak{X}} p(x) \log \left(\frac{p(x)}{q(x)} \right)$$

In order to achieve a consistency for the case $q(x) = 0$, we define:

$$p \cdot \log \left(\frac{p}{0} \right) := \infty$$

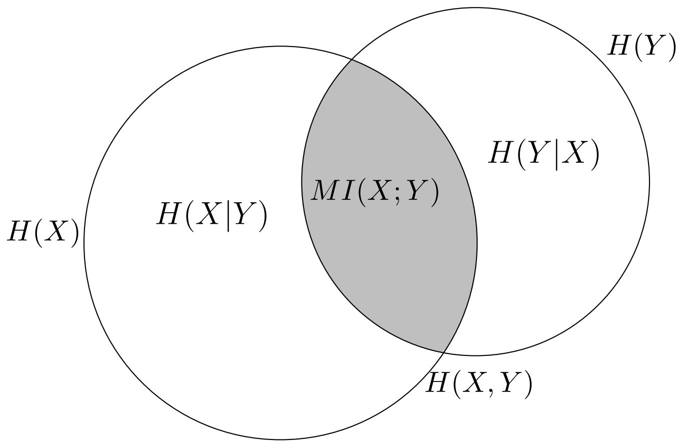


Figure 3.1: Relationship between mutual information and the entropy of two random variables

3.2 Machine Learning

The concept of machine learning is based on having a system analyse data and gaining knowledge about the underlying process that generated the data.
In general this means that the computer tries to detect patterns in the analysed data and uses this knowledge to make predictions about unseen data. The weight and accuracy of the patterns depends on the data and the predictions they are used for. This overview has the purpose of showing the involved concepts and the algorithm that was used for the patch classifier in Chapter 6. It derived from [BW10], where a more detailed description of the concepts and algorithm can be found.

3.2.1 Foundations of Machine Learning

Let $\mathfrak{X}_n \subseteq \mathbb{R}^n$ be the *input space.* Then the *learning universe* is defined as $\mathfrak{U}_n := \mathfrak{X}_n \times \{0,1\}$. A pair $U = (X, Y)$ with a random $X \in \mathfrak{X}_n$ and a random classification $Y \in \{0,1\}$ induces a distribution $\mathbb{P}_U$ on $\mathfrak{U}_n$.
Our aim is to predict the classification Y of the observation X of length n with the use of a hypothesis $h : \mathfrak{X}_n \to \{0,1\}$.

Definition 3.8
The risk of a hypothesis h is defined through

$$r(h) := \mathbb{P}(h(X) \neq Y)$$

Definition 3.9
The *posterior probability* of the classification on the observation $x \in \mathfrak{X}$ is defined as

$$\eta(x) := \mathbb{P}(Y = 1|X = x)$$

Definition 3.10
The classifier

$$g^*(x) = \begin{cases} 1, & \text{if } \eta(x) > \frac{1}{2}; \\ 0, & \text{otherwise} \end{cases}$$

is called the Bayes classifier, the risk $r(g^*)$ the Bayes risk.

A hypothesis h is calculated from an independent and identically distributed (i.i.d.) *learning sample*

$$\mathbf{U}_m = (U_1, U_2, ..., U_m),$$

where U is independent of $\mathbf{U}_m$, and $U_i = (X_i, Y_i)$, for $i = 1, 2, ..., m$ induces the same distribution as U on $\mathfrak{U}_n$. These hypotheses are random and parameterized by the length m, as they are transformations of the learning sample. They are denoted as $\hat{H}_m$, which is also called a discrimination rule.
The risk of a discrimination rule is given by:

$$r(\hat{H}_m) := \mathbb{P}(\hat{H}_m(X) \neq Y|\mathbf{U}_m).$$

Definition 3.11
A discrimination rule $\hat{H}_m$ is consistent, if $r(\hat{H}_m) - r(g^*)$ converges stochastically to zero for $m \to \infty$.

Mammen and Tsybakov [MT99], [Tsy04] showed that the optimal convergence rates are determined by the complexity of the class G^* of possible Bayes classifiers and a *margin* parameter.

Definition 3.12
The distribution $\mathbb{P}_U$ has *margin parameter* $0 < k \leq 1$, if there is a *margin constant* $d > 0$ so that for every hypothesis h

$$\mathbb{P}(h(X) \neq g^*(X)) \leq d \cdot (r(h) - r(g^*))^k$$

3.2.2 PAC Learning with Noise

Valiant introduced the Probably Approximately Correct model of learning (PAC learning) [Val84], which is a form of *concept learning.* A concept specifies how to divide vectors from the $\mathbb{R}^n$ into positive and negative examples, in general the input space is not restricted to $\mathbb{R}^n$. A learning algorithm is responsible for inferring an unknown target concept g^* out of a known concept class C_n. Rather than using Valiant's model, the functional scenario of PAC learning considered in Haussler et al. [HW91] is used.

We also need the concept of a *representation class* H_n; these are the hypotheses which comprise the class G_n^* of possible Bayes classifiers.
In the PAC learning literature it is standard to measure the accuracy of hypotheses by the error rather than by the risk.

$$err(\hat{H}_m) \geq r(\hat{H}_m) - r(g^*) \tag{3.2}$$

A *functional learning algorithm* is specified in the following way:
A learning algorithm A of a concept class C_n by a representation class H_n of C_n takes the learning sample U_m, the desired accuracy $\epsilon \in (0,1)$ and the confidence $\delta \in (0,1)$ as input. Additionally, the representation size s of the target concept g^* is known. The output is a hypothesis $\hat{H}_m \in H_n$ that ϵ approximates the target concept with probability at least $1 - \delta$.

$$err(\hat{H}_m) := \mathbb{P}(\hat{H}_m(X) \neq g^*(X) | U_m) \leq \epsilon. \tag{3.3}$$

The PAC model introduced by Valiant only works on noise-free data, i.e., $Y = g^*(X)$. In order to make the algorithms robust and applicable to real-life application, which are rarely noise free, a noise model is required, i.e., $Y = g(X) \oplus S$. Noise models were introduced by Valiant ([Val85]) and further analyzed by Kearns and Li ([KL93]). We make use of the general noise model presented in [BW10].
The random variable S is called the random noise, it may be dependant on the observation X. The random noise rate $\nu(X)$ is defined as:

$$\nu(x) := \mathbb{P}(S = 1 | X = x)(x \in \mathfrak{X}). \tag{3.4}$$

Given a concept class C_n the classification noise model N_n consists of the set N_{g^*} of random noise rates $\nu(X)$ for every target concept $g^* \in C_n$.
An upper bound for the *expected noise rate*

$$\nu := \mathbb{E}_\nu(X) \tag{3.5}$$

can be given as $\nu \leq \nu^{(b)} < \frac{1}{2}$ [BW10].
With this noise model a definition for a PAC-learner can be given:

Definition 3.13
A functional learning algorithm A is called an efficient PAC learner of a concept class C_n by a representation class H_n in the noise model N_n, if

- for any $\epsilon, \delta \in (0, 1)$, for any length n, for any target concept size bound s, and for any expected noise rate bound $\nu^{(b)} < \frac{1}{2}$, a minimal sample length $\mathbf{m_A}(\epsilon, \delta, n, s, \nu^{(b)})$ exists so that for all $\mathbf{m} \geq \mathbf{m_A}$, for any distribution $\mathbb{P}_X$ of the input element $X \in \mathfrak{X}_n$, for any target concept $g^* \in C_n$ of size at most s, and any noise rate $\nu(X) \in N_{g^*}$ whose expectation is less than or equal to $\nu^{(b)}$ it returns a hypothesis $\hat{H}_m$ such that with probability at least $1 - \delta$ equation 3.3 holds.
- the minimal sample length $\mathbf{m_A}(\epsilon, \delta, n, s, \nu^{(b)})$ is polynomial in $1/\epsilon, \ln(1/\delta), n, s$ and $1/(\frac{1}{2} - \nu^{(b)})$.
- its running time is polynomial in
  $m, 1/\epsilon, \ln(1/\delta), n, s$ and $1/(\frac{1}{2} - \nu^{(b)})$.

3.2.3 Learning in the Presence of Classification Noise

Kearns developed the statistical query (SQ) model [Kea98] to be able to devise efficient noise-tolerant learning algorithms in extension of Valiant's model.
The SQ model is based on a measurable and efficiently computable query function:

$$\chi : \mathfrak{X}_n \times \{0,1\} \to [a,b],$$

where $a < b$ are real numbers. A statistical query $[\chi, \tau]$ with the so-called tolerance τ the learning algorithm requests for an estimate $\hat{e}_{\chi,g^*}$ of the expected value

$$e_{\chi,g^*} := \mathbb{E}_{\chi}(X, g^*(X)) \tag{3.6}$$

such that $|e_{\chi,g^*} - \hat{e}_{\chi,g^*}| \leq \tau$.
For $y_0 \in \{0,1\}$, a conditional statistical query (CSQ) $[\chi, \tau/y_0]$ is a request for an estimate $\hat{e}_{\chi,g^*,y_0}$ of the conditional expectation

$$e_{\chi,g^*,y_0} := \mathbb{E}(\chi(X, y_0)|g^*(X) = y_0) \tag{3.7}$$

with the additive error bound τ.
Thus, learning algorithms in the SQM are defined as follows:

A SQ learning algorithm A of a concept class C_n by a representation class H_n has the accuracy ϵ, the length of the observation n as input and the size s of the target concept $g^* \in C_n$ is known. The algorithm has access to an oracle $STAT(\mathbb{P}_X, g^*)$. This oracle can answer queries $[\chi, \tau]$ for expected values defined in equation 3.3. The output is a hypothesis $h \in H_n$.

Thus according to Kearns [Kea98] efficient consistency in the SQ model is defined as:

Definition 3.14
A SQ learning algorithm A is called an efficient and consistent learner of a concept class C_n by a representation class H_n, if for any ϵ, any n, any target size bound s, any distribution $\mathbb{P}_{\mathbb{X}}$ of the input element $X \in \mathfrak{X}$, and any target concept $g^* \in C_n$ of size s

- the output $h \in H_n$ satisfies

$$err(h) := \mathbb{P}(h(X) \neq g^*(X)) \leq \epsilon \tag{3.8}$$

- the reciprocal of the tolerance τ is bounded from above, for every statistical query, by the tolerance bound $\mathbf{tb}(\epsilon, n, s)$, which is a polynomial in $1/\epsilon$, n, and s;
- the evaluation time of every query and function used is polynomial in $1/\epsilon$, n, and s;
- the running time is polynomial in $1/\epsilon$, n, and s.

3.2.4 Combining PAC and SQ Algorithms

We will now combine the concepts of PAC and SQ models to create an algorithm that has access to the learning sample U_m like it is in the case in PAC learning as well as an oracle $STAT(\mathbb{P}_X, g^*)$ as it is used in the SQM. This combination of the two concepts is taken from [BW10].
Similar to the target concept size, the margin constant d, and the margin parameter k are known to the learning algorithm.
The specification for a PAC+SQ learning algorithm A is as follows. A learning algorithm of a target concept class C_n by a representative class H_n has as input the learning sample $\mathbf{U}_m$ and the parameters accuracy $\epsilon \in (0, 1)$ and confidence $\delta \in (0, 1)$. As mentioned above the learning algorithm knows the target concept size s of $g^* \in C_n$, the parameter α, the margin constant d, and the margin parameter k. It also has access to an oracle $STAT(\mathbb{P}_X, g^*)$ to make (conditional) statistical queries $[\chi, \tau]$ and $[\chi, \tau, y_0]$. χ is a query function, τ is the tolerance of the query and $y_0 \in \{0, 1\}$ is a classification. The output is a hypothesis $\hat{H}_m \in H_n$.

The definitions of an efficient and consistent PAC respective SQ learner are combined to define an efficient PAC+SQ learner.

Definition 3.15
A learning algorithm A that follows the above specification is an efficient PAC+SQ learning algorithm a concept class C_n by a representation class H_n, if

- for any $\epsilon, \delta \in (0, 1)$, for any length n, for any target size bound s, and for any expected noise rate bound $\nu^{(b)} < \frac{1}{2}$, there is a minimal sample length $\mathbf{m}_A(\epsilon, \delta, n, s, \nu^{(b)})$ and a tolerance bound $\mathbf{tb}_A(\epsilon, n, s)$.
  For every $m \geq \mathbf{m}_A(\epsilon, \delta, n, s, \nu^{(b)})$, for any distribution $\mathbb{P}_X, X \in \mathfrak{X}_n$, for any target concept $g^* \in C_n$ of size at most s, and for any noise rate $\nu(X)$ with an expectation of $\nu \leq \nu^{(b)}$) with probability at least $1 - \delta$

  $$\mathbb{P}\big(err(\hat{H}_m \leq \epsilon)\big) \geq 1 - \delta,$$

  where the reciprocal of the tolerance of every (conditional) statistical query made is bounded from above by $\mathbf{tb}(\epsilon, \delta, n, s)$;
- the minimal sample length is polynomial in $1/\epsilon$, $\ln(1/\delta)$, n, s, and $1/(\frac{1}{2} - \nu^{(b)})$;
- the tolerance bound is polynomial in $1/\epsilon$, n and s;
- the evaluation time of each query function χ is polynomial in n;
- the overall running time is polynomial in m, $1/\epsilon$, $\ln(1/\delta)$, n, s, and $1/(\frac{1}{2} - \nu^{(b)})$.

Finally, we require a definition on how to rate the complexity of the query stage of a learning algorithm.

Definition 3.16
The query complexity $\mathbf{qc}_A(\epsilon, \delta, n, s, \nu^{(b)})$ of an efficient PAC+SQ learning algorithm is defined as the supremum of the number of query calls the algorithm has to make, over all values of the learning sample compatible with the parameters n, s and $\nu^{(b)}$.

The query space of an algorithm is the set of query functions Q_n used on observations of the length n.

3.2.5 The Orthogonal Noise Model

No general way to perform SQs in the PAC+SQ model with CN is known, making it inoperable. However, within the orthogonal noise model it can be shown that SQs are operable.

Definition 3.17
The conditional expected noise rate for $y_0 \in \{0, 1\}$ given $\{g^*(X) = y_0\}$ is defined as

$$\nu_{y_0} := \mathbb{E}(\nu(X)|g^*(X) = y_0)$$

We limit the definition of orthogonal noise rates to the equivalence between noise rates and query functions. For a more detailed definition of orthogonal noise rates and the proofs of the following theorems see [BW10].

Definition 3.18
Let C_n be a target concept class and let Q_n be a set of query functions. Then a noise rate $\nu(X)$ is orthogonal to Q_n given a target concept $g^* \in C_n$, if and only if

$$\mathbb{E}(\nu(X) \cdot \chi(X, 1)|g^*(X) = y_0) = \nu_{y_0} \cdot \mathbb{E}(\chi(X, 1)|g^*(X) = y_0)$$

$\wedge$

$$\mathbb{E}(\nu(X) \cdot \chi(X, 0)|g^*(X) = y_0) = \nu_{y_0} \cdot \mathbb{E}(\chi(X, 0)|g^*(X) = y_0)$$

At this point, we can state the central theorem for orthogonal noise rates which gives the possibility to simulate any PAC+SQ learning algorithm.

Theorem 3.19
Let H_n be a representation class of a concept class C_n. Every efficient PAC+SQ learning algorithm of C_n by H_n having query space Q_n can be simulated by an efficient PAC+SQ learning algorithm of C_n by H_n in the noise model $Q_n^\perp$ orthogonal to Q_n.

The estimate of a conditional query $[\chi, \tau, y_0]$ is given by the following theorem.

Theorem 3.20

Let χ be a query function, $g^* \in C_n$ a target concept and $\nu(X)$ an orthogonal noise rate. Then for $y_0 \in \{0,1\}$

(3.9)
$$\mathbb{E}(\chi(X,y_0)|g^*(X)=y_0) = \frac{(1-\nu_{\bar{y}_0})\mathbb{E}[\mathbb{1}_{\{Y=y_0\}}\chi(X,y_0)] - \nu_{\bar{y}_0}\mathbb{E}[\mathbb{1}_{\{Y=\bar{y}_0\}}\chi(X,y_0)]}{\mathbb{P}(Y=y_0)-\nu_{\bar{y}_0}}.$$

The function $\mathbb{1}$ is defined as

$$\mathbb{1}_{\{cond\}} = \begin{cases} 1 & \text{if } cond \text{ is } true \\ 0 & \text{if } cond \text{ is } false. \end{cases}$$

It is reasonable to assume that the noise rate is orthogonal to the current query space given a target concept. Otherwise, it would mean that the noise itself contains relevant information. In general, it is unlikely that relevant information is only inferred from differences in noise rates.

Before we can state a theorem about simulating a PAC+SQ learner we need a few more definitions.

Let T be a class of measurable functions from the learning universe $\mathfrak{U}_n$ to a closed interval $[a,b]$, where $c := b-a$.

Definition 3.21

The growth function $S_T(m)$ is defined to be

$$S_T(m) = \sup_{u_1,u_2,\dots,u_m} \left|T_{u_1,u_2,\dots,u_m}\right|.$$

The VC dimension $vc-dim(T)$ of a class T is the largest m such that $S_T(m)=2^m$.

With this we can formulate the following theorem which is one of the main results in [BW10].

Theorem 3.22

Let B be an efficient PAC + SQ learner of a concept class C_n by a representation class H_n having effective query spaces $Q_{\epsilon,n,s}$, minimal sample size $\mathbf{m_B}(\epsilon,\delta,n,s,\nu^{(b)})$, and tolerance bound $\mathbf{tb_B}(\epsilon,n,s)$, and let

$$\mathbf{m_B} := \mathbf{m_B}((\epsilon/(2d))^{1/k}, \delta/4, n, s, \nu^{(b)}), \qquad \mathbf{tb_B} := \mathbf{tb_B}((\epsilon/(2d))^{1/k}, n, s),$$

where k and d are the margin parameter and the margin constant.

Then this algorithm B can be simulated by an efficient PAC learner A of C_n by H_n in the noise model $Q_n^\perp$ orthogonal to the query space Q_n of B such that the minimal sample size $\mathbf{m_A}(\epsilon,\delta,n,s,\nu^{(b)})$ of A can be bounded from above by

$$O\left(\mathbf{m_B} + \frac{\mathbf{tb_B}^2\big(vc-dim\big(Q_{(\epsilon/(2d))^{1/k},n,s}\big)+\ln(1/\delta)\big)}{(1/2-\nu^{(b)})^2} + (d/\epsilon)^{2/k}\ln\frac{\mathbf{tb_B}}{(1/2-\nu^{(b)})\delta}\right),$$

given B uses unconditional queries only, and by

$$O\left(\mathbf{m_B} + \frac{\mathbf{tb_B}^2\big(vc-dim\big(Q_{(\epsilon/(2d))^{1/k},n,s}\big)+\ln(1/\delta)\big)}{(1/2-\nu^{(b)})^2\epsilon^2} + (d/\epsilon)^{2/k}\ln\frac{\mathbf{tb_B}}{(1/2-\nu^{(b)})\epsilon\delta}\right)$$

otherwise.

3.2.6 A PAC+SQ Simulator for Haussler's Covering Method

The goal for our purposes is to efficiently learn Boolean conjunctions of clauses with a PAC+SQ learning algorithm, this can then be used as foundation for our hypotheses for the patch classifier in Chapter 6. We define a clause as an efficiently computable 0/1-valued function on the input space. The preceding Section has shown that this learning algorithm can be simulated by an operable PAC+SQ learning algorithm.
A target concept g^* is a conjunction of at most s elements from the set:

$$C_n := \{c_1(X), c_2(X)...c_n^\gamma(X)\}$$

of n^γ clauses, where n is the length of the random observation $X \in \mathfrak{X}$ and γ is a positive constant. The amount of s is called the size parameter of the target concept. The target concept class is denoted by $C_n - Mon$.
A solution for this was given by Haussler [Hau88] by using a covering method though this solution was based on Valiant's noise free model. Kearns [Kea98] adapted this solution for the constant classification noise model. Based on Kearns algorithm Brodag et al.[BW10] developed the following algorithm.
The algorithm is divided into two parts *the prune phase* and *the cover phase.*
The prune phase starts with a query for every clause $c \in C_n$ to get an estimate of the conditional failure probability $\mathbb{P}(c(X) = 0, g^*(X) = 1)$ of c given $g^*(X) = 1$ within accuracy $\Theta(\epsilon^2/s)$. Only the clauses are kept as candidates whose estimates of the conditional failure probability fulfills an $O(\epsilon^2/s)$. We refer to these clauses as *survivors of the prune phase.*
This ensures that

- at most s clauses forming the target concept g^* are survivors of the prune phase;
- no matter how we select the clauses from the set of survivors of the prune phase to form the hypothesis $\hat{H}_m$, we have

  $$\mathbb{P}\{\hat{H}_m = 0, g^*(X) = 1\} = O(\epsilon) \tag{3.10}$$

  as long as their number is an $O(s/\epsilon)$.

The cover phase needs only negative examples and does not make any statistical queries. Therefore, the query space used is determined by the prune phase and equals:

$$Q_n = \{c_i(x) \wedge y | i = 1, 2, ...n^\gamma\}.$$

A sample of negative instances of length m is drawn according to the distribution $\mathbb{P}(X = \bullet | Y = 0)$ and used as an input. Then the subset of input covered by the negations of all candidates retained in the prune phase is computed. This set is then covered by the means of a greedy algorithm which needs $r = \Theta(s \log(1/\epsilon))$ iterations. Only the clauses that are part of the cover are used to form the hypothesis $\hat{H}_m$.

This PAC+SQ learner for boolean conjunctions of s clauses taken from a query space of cardinality n^γ has sample complexity $O((\log(1/\delta) + \log n)s/\epsilon^2)$, and tolerance bound $O(s/\epsilon^2)$. This is an improvement compared with the adaption of Haussler's covering method by Kearns ([Kea98]). According to Theorem the overall sample size dependence differs on s which is only s^2, whereas Kearns has s^3. On ϵ Kearns has a dependence of ϵ^{-4}. The here presented algorithm has a dependence of ϵ^{-4} only if the margin parameter $k \geq \frac{1}{2}$ else it has $\epsilon^{\frac{-2}{k}}$.
A more detailed analysis can be found in [BW10]. The algorithm was implemented by Steffen Herbold [HW11].

Chapter 4

Data Selection and Pre-Processing

In this chapter we provide insights about the protein information we require for our classifiers. We also describe the methods used to gain this information out of the available protein information databases.
The foundation we needed was a reliable database upon which we can base our predictions. As the patch classifier is for protein-protein-interactions we need a database with already known protein-protein-interactions. We further limited our research to interactions of homodimer proteins. In general, homodimers are considered easier to predict than heterodimers. Therefore, if something does not work on homodimer data, we assume it does not work for heterodimer data either. Another reason for this limitation is the availability and quality of the respective MSAs for the proteins. Homodimer interactions have only one MSA as the chains are identical. This minimizes the risk of having an ill suited MSA for an interaction.
We decided to use the database from the Nussinov group exclusively as the foundation for our research([KN04]). There are several reasons for this decision.
The Nussinov database (NDS) provides information on two-chain protein-protein-interfaces. These are the type of interfaces we are looking for. Another factor was that the Nussinov database is derived from the publicly available protein data bank PDB ([BB00]). Thus we can easily access the underlying protein data, which allows the reconstruction of the results of the Nussinov database.
The most important reason for picking the Nussinov database is that the protein-protein-interactions are divided into clusters of non-redundant datasets. The non-redundancy reduces the threat of *overfitting*, which makes the NDS well suited for machine learning. Overfitting is the process when predictions are based on noise instead of the underlying process. This can happen when certain information is overrepresented in the model.
This database contains about 25 000 protein-protein-interactions, partitioned into 3799 non-redundant clusters, which makes it a solid size for our learning algorithms.

4.1 Generating Interface Information Files

For our research we need to pre-process the data we have from the Nussinov database (NDS) and the associated protein files taken from the PDB. Although these two databases contain all the relevant information we need, there is also a lot of unnecessary information. Especially in the PDB files, as these contain all known information about a species and not only the parts we need for our research.
The relevant data for our project are the primary protein structure and parts of the tertiary and quatenary structure. This information can be extracted or calculated from the PDB files by using the NDS as reference to which exact parts are needed.
To simplify this process we created Interface Information Files (IFFs). In these files we store all the relevant information for our projects, hence, we can speed up the actual data handling. As this information does not change, the IFFs have to be generated just once as a pre-processing step. A fictional IFF example is shown in Appendix A. To handle the PDB files we use an implementation based on the biochemical algorithms library (BALL)[HK10].

4.1.1 Protein Chains

The first information we need are the chains or backbones of the two interacting proteins. By accessing the NDS we get the name of the species and the identifiers of the two interacting chains. The name of the protein lets us retrieve the relevant file from the PDB.
With the two chain identifiers we are able to extract the backbone information from the PDB file and write it to the IFF in form of single letter amino coding. This coding allows easier handling of the chain information in the actual implementations.

4.1.2 Surface and Core

The second part of the IFF is the distinction between the surface and core regions of the protein chain. This is mainly for the patch classifier as we need to separate surface and core in order to create learning samples.
By calculating the solvent accessible surface, we determine how accessible the theoretical atom surface of a residue is by a specified solvent. This is done with an algorithm introduced by Lee and Richards ([LR71]), which uses water molecules as standard solvent. The algorithm has been improved by Shrake and Rupley ([SR73]). It is implemented in the BALL library [HK10] in a more efficient runtime version from Eisenhaber et al. [ES95].

In the course of this thesis we use the following definition:

Definition 4.1
A residue on a protein chain is considered to be on the surface of the chain, if the solvent accessible surface area (SASA) is above 15%.

As the chains and their spatial structure information are embedded in the quaternary structure information, we first have to dissolve the protein complex. Therefore we have to extract the spatial structure information of a single chain out of the protein complex via BALL first and then use our SAS algorithm for this single chain. Otherwise, only those regions of the chains would be identified as surface that are also surface regions of the whole protein complex.

4.1.3 Neighborhood on the Chain

Additionally we require information about spatial neighborhoods of amino acids on the same chain in order to identify spatially linked areas for our patch classifier.
With BALL we calculate the distance between two residues on the same chain using the major carbon atoms as fix points. This is the same method used by Nussinov et al. ([KN04]) to describe nearby residues. The threshold we use is the same as the one used by Nussinov.

Definition 4.2
Two residues on a protein chain are neighbors if the distance of the major carbon atoms is below 6 Å.

4.1.4 Interaction Pairs

The last bit of information to be gathered from the PDB files is the notation of the actual interacting residue pairs between the two chains.
Again, using BALL the distance between two residues on the different chains is calculated by determining the distance between any two atoms of each residue ([KN04]). Again the threshold below is adopted from Nussinov.

Definition 4.3
Two residues on different protein chains are interacting, if the distance between any two atoms of each residue is below the sum of their corresponding Van-Der-Waals radii plus 0.5 Å.

4.2 Multiple Sequence Alignments

Apart from the PDB files we require information about the MSAs for the respective chains of the interacting proteins.
We use the dictionary of secondary structure of proteins (DSSP) database ([KS83]) and get the MSA to each chain of our homodimers designated by the NDS. The MSAs for each chain are identical, therefore, we simply pick one at random if both are available. Each MSA is then filtered in a pre-processing step. To avoid having too much redundant information in the MSAs, we remove all species from the MSA that are too similar to the original protein chain. We also remove all species from the MSA which have almost no similarity to the original protein. We measure similarity with the *Hamming distance* ([WM09]). The Hamming distance between two strings is the number of positions at which the corresponding symbols are different. For two proteins the Hamming distance is the number of positions where the corresponding amino residues differ. We then normalize the distance by the length of the protein chains to have a comparable measurement.
We use the threshold of 90% as upper bound for similarity and a threshold of 30% as lower bound.

4.3 Datasets

From the original Nussinov database we took the 3799 representatives of the non-redundant clusters as our base set.
A few representatives had to be removed from the set due to different reasons. For example, that no MSA is available for a particular protein chain, or that the PDB file contains flawed information which makes an automated approach of generating the IFF impossible.
Furthermore, we split the representatives into heterodimer and homodimer.
The last step is to generate a separate learning set $\mathfrak{L}$ and a testing set $\mathfrak{T}$. This is done by assigning all entries from the complete list of homodimers randomly to either set, with a chance of 80% that an entry is assigned to the learning set.
We use the $\mathfrak{L}_{\mathfrak{hom}}$ set for calculating and generating thresholds and refinement methods. The $\mathfrak{T}_{\mathfrak{hom}}$ is used as a testing device to evaluate the results from our methods independent of their input. Thereby we guarantee that the result is not tailored to one specific data set.
As a result we have two sets of data about homdimers, see table 4.3.

Set	Amount of entries
$\mathfrak{L}_{\mathfrak{hom}}$	1175
$\mathfrak{T}_{\mathfrak{hom}}$	277

Table 4.1: The two homodimer data sets

Chapter 5

Entropy Based Ranking of Significant Protein Residues

The idea for this classifier is based on the work of Merkl et al. [MZ08]. We use the same concept of finding important residues through the means of entropy and then rank these residues depending on their interactions between each other. However, instead of just picking the 75 residues with the highest entropy like Merkl et al. do, we have devised a mathematical model to detect all potentially significant residues, based on different entropy variants.
This chapter is divided into four parts. The first part is about observing the characteristics of significant protein residues through the means of entropy, and designing a mathematical model to represent these characteristics based on our learning data.
In the second part we develop a method to amplify the characteristics of significant protein residues with the information gained in the first part. In the third part we present a concept for ranking the significant protein residues.
The fourth part presents an assessment and comparison with similar methods on certain case files.

5.1 Observing and Modelling Coevolution

Throughout the chapter we will use the term of *substitution* of residues and pairs. We define substitution as the transition from one amino residue to another amino residue in different rows of a MSA in a fixed position. A transition between two identical amino residues is also called a substitution. For example, if the first amino acid in the original protein of the MSA is Glycine (G) and the first amino acid of the second protein in the MSA is Leucine (L), then we have a substitution between G and L.

We want to use the concept of mutual information to quantify coevolution of two protein chain residues. Although this has so far not been proven conclusively, there are strong indications that suggest, that correlation and coevolution are connected [GC00]. We assume that coevolution has taken place when we can find a measurable correlation for a residue pair on a protein chain. The correlation is traceable through residue substitutions in the corresponding MSA.

5.1.1 Quality of Mutual Information and Normalized Mutual Information

Let l, k with $l \neq k$ be two columns and m be the number of sequences of a MSA. If we regard co-occurring amino acids in columns l, k as random variables, it allows us to apply the concepts of information theory.

Let X and Y be random variables with alphabet $\mathfrak{A}$. With X being the observations of amino acids in column l, and Y being the observations of amino acids in column k, the empirical joint probability $\hat{p}(X = x, Y = y)$ is calculated as.

$$\hat{p}(x_i, y_j) = \frac{\#(x_i, y_j)}{m},$$

where $\#(x_i, y_j)$ is the number of observations of amino acids (x_i, y_j) over all MSA sequences m in columns l and k. The marginal distribution $P(X = x)$ of column l, is calculated as

$$\hat{p}(x_i) = \frac{\#(x_i)}{m},$$

where $\#(x_i)$ is the number of observations of amino acid (x_i) over all MSA sequences m in columns l. The marginal distribution of $P(X = x)$ of column k is calculated accordingly. The mutual information of columns l, k is $MI(X;Y)$.

Normalization of the Mutual Information

The magnitude of mutual information (MI) values depends on the observed alphabet size and the degree of correlation between the two columns. Therefore, it is necessary to find a suitable scaling that removes the alphabet size as an impacting factor. For example we have two different column pairs (u, v) and (w, z) which have an absolute correlation between each of their respective columns. Then both pairs are uniformly distributed in $P(X, Y)$ and, therefore, $MI(X;Y) = \log(|\mathfrak{X}|)$, where $\mathfrak{X}$ is the observed alphabet in X. If the observed alphabet size of pair (u, v) is larger than that of (w, z), (u, v) has a higher mutual information.
Martin et al. [DW05] compared different entropy based quantifiers for coevolution prediction among functional important residues. The conclusion of that work was, that normalized mutual information (NMI) values offer better results than methods that are based on pure (MI) values.

There are different approaches on how to normalize the MI values. A first variant utilizes the joint entropy to normalize the MI.

$$NMI_{joint}(X;Y) := \frac{H(X) + H(Y) - H(X,Y)}{H(X,Y)} \tag{5.1}$$

A second variant uses the sum of the entropy components $H(X) + H(Y)$ to normalize the MI.

$$NMI_{sum}(X;Y) := \frac{H(X) + H(Y) - H(X,Y)}{H(X) + H(Y)} \tag{5.2}$$

Because $NMI_{sum}(X;Y)$ has results in the range of $[0, 0.5]$, the $NMI_{sum}(X;Y)$ values are multiplied with 2, to attain the standard range of $[0, 1]$.
Merkl et al. use the following variant:

$$U(X;Y) := 2 \cdot \frac{H(X) + H(Y) - H(X,Y)}{H(X) + H(Y)} = 2 \cdot NMI_{sum}(X;Y) \tag{5.3}$$

$MI(X;Y)$ can be seen as intersection of $H(X)$ and $H(Y)$, as seen in Chapter 3.1. NMI_{joint} and U, represent a ratio of this intersection between $H(X)$, $H(Y)$ and $H(X,Y)$. In the case of U values, this intersection is represented twice.
The choice of the normalization influences the whole model and is consequently a very important factor. Consider a pair of columns where one column consists to 80% of the amino residue A, the other column to 89% of the amino residue E. Then the pairing A-E would appear quite often, not necessarily due to co-evolution, it could be pure chance, since the possibility of that pairing is so high. In our model we want to reduce the impact of pure chance pairings by picking our normalization accordingly. We chose four different normalization variants and tested them on different samples to be able to evaluate, which suits the model best.
Let $\mathfrak{X}$ be the observed alphabet in column l and $\mathfrak{Y}$ be the observed alphabet in column k. For the maximal entropy of columns l and k follows:
$H(X) \leq \log|\mathfrak{X}|$ and $H(Y) \leq \log|\mathfrak{Y}|$. The designator a stands for the alphabet. Therefore:

$$NMI_{max\_a} := \frac{H(X) + H(Y) - H(X,Y)}{max\{\log|\mathfrak{X}|, \log|\mathfrak{Y}|\}}$$
$$NMI_{min\_a} := \frac{H(X) + H(Y) - H(X,Y)}{min\{\log|\mathfrak{X}|, \log|\mathfrak{Y}|\}}.$$

In concordance with the first two NMI variants, the observed alphabet size can still be utilized. The next two NMI variants use the actual entropy of one column.
The third variant uses the entropy of the column with the larger alphabet:

$$NMI_{max} := \frac{H(X) + H(Y) - H(X,Y)}{max\{H(X), H(Y)\}}.$$

The fourth variant then uses the entropy of the column with the smaller alphabet:

$$NMI_{min} := \frac{H(X) + H(Y) - H(X,Y)}{min\{H(X), H(Y)\}}.$$

All these variants as well as the NMI_{joint} and the $U - values$ share one problem. If we have a fully conserved column, the entropy is zero, thus we would have a division by zero in our calculation. Due to this and as mentioned above, a fully conserved column is more likely to simulate co-evolution by pure chance, none of these columns are considered in the model.

NMI Value Assessment

A comparison of the different normalization variants is shown in Table 5.1. The first row shows examples of different residue column pairs, the following rows are the different NMI variants. Examples $a-e$ are taken from [DW05]. For the majority of the examples the NMI values are pretty similar, the greatest differences can be found in examples c and f.
Example c is a case, where the general view of coevolution and correlation influences which normalization variant is preferred. The column pair has only a moderate MI value due to AE and CE having the same probability. If viewed as correlation, it can be interpreted as a strict binding, where A and C require E as partner. We want to support this correlation view, as it is unlikely to have happened by pure chance. We prefer a normalization variant that rates this example higher. The variants U, NMI_{min_a} and NMI_{min} fulfill this condition.
Example f is the case, where two columns are both strongly conserved and appear to be completely correlated. The variants U, NMI_{joint},NMI_{max} and NMI_{min} rate this example with the maximum score of one, because $MI(X;Y) = H(X) = H(Y) = H(X,Y)$. NMI_{max_a} and NMI_{min_a} rate this example with a medium score which fits our interpretation of correlation better. We feel that although there could be co-evolution in these two columns, it could also be the product of pure chance, and thus we do not want to rely on this uncertain information. The variants NMI_{max_a} and NMI_{min_a} fulfill this condition. However, we cannot rule out that it is a sign of co-evolution, hence we do not want to have a score which is very low.

	a	b	c	d	e	f	g	h
	AE AE AE CM CM CM	AE AE CM CM DF DF	AE AE CE CE DF DF	AE AE CE DM DF DF	AE AM CE CM CE CM	AE AE AE AE AE CM	AE AE AE AE AM CM	AE AE CE AF DF DF
MI	1	1.58	0.92	1	0	0.65	0.32	0.54
NMI_{joint}	1	1	0.58	0.52	0	1	0.25	0.25
U	1	1	0.73	0.69	0	1	0.4	0.44
NMI_{max_a}	1	1	0.58	0.63	0	0.65	0.32	0.34
NMI_{min_a}	1	1	0.92	0.63	0	0.65	0.32	0.54
NMI_{max}	1	1	0.58	0.69	0	1	0.34	0.37
NMI_{min}	1	1	1	0.69	0	1	0.34	0.54

Table 5.1: Normalization comparison

All things considered, we are left with two scores that rate the examples, that represent our interpretation of correlation best: the NMI_{min_a} variant and the NMI_{min} variant. As Merkl et al. use the U variant, we are examining this variant as well .

Comparison of U, $\mathbf{NMI_{min\_a}}$ and $\mathbf{NMI_{min}}$

We have identified the best candidates for normalization and we want to compare the respective results for actual proteins and MSAs.
We chose to compare the overlapping high value residues of all three scores. We expect that there is an overlap, because the variants are similar to each other. Also the significant residue pairs should not be completely different depending on the score. A complete overlap is unlikely, because the variants express different views on how coevolution is expressed through correlated pairs.
We chose 200 MSAs for the comparison and in each MSA the 75 highest NMI scoring pairs were picked. We picked 75 for comparability to the work of Merkl et al. [MZ08]. It is an arbitrarily picked threshold that is not based upon a valid mathematical model, but for this comparison the number of residue pairs is sufficient.
The largest overlap is between NMI_{min} and NMI_{min_a}; it averages 38 pairs per MSA for these two variants. This is due to the similar equations and the similar model these variants express. Between U and NMI_{min} we have an overlap of 22 pairs and between U and NMI_{min_a} an overlap of 19 pairs. The use of the actual entropy as normalization in U, and NMI_{min} instead of the theoretical maximum entropy as in NMI_{min_a} is a possible explanation for the slightly higher overlap between the U and NMI_{min} variants.

The overlap between all three variants is 12 pairs for each MSA. This shows that some residue pairs are always scored high, regardless of the model, while the rest depends on the model, which meets our expectations on the models.

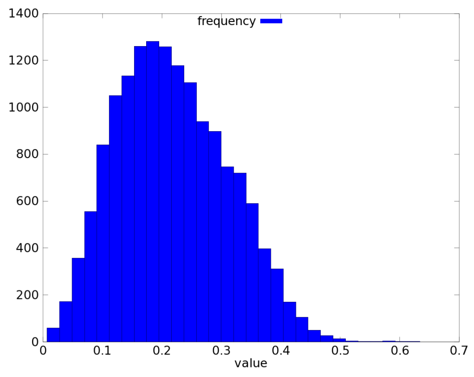
5.1.2 Conserved Columns and Gap Handling

As explained above, fully conserved columns would lead to a division through zero in our variants, thus we exclude them from our calculations. We are also excluding columns that are highly conserved with the threshold of 95% conservation. This is also done because fully conserved columns or almost fully conserved columns yield no or very little information through entropy.
The column pairs have, depending on the MSA, a number of gaps which have to be dealt with. A simple choice would be treating the gaps as a 21st amino residue to not lose those rows with gaps. This procedure has quite a few drawbacks though. If one assumes a gap is where there is even less correlation than between two random residues, we would not lose the rows, but falsify our information with wrong data. If we have a row where both residues are gaps, we assume a correlation where no information is available. Hence, we decided to simply eliminate all rows with gaps from our calculation. If the number of gaps exceeds 25% of the column length, the whole pair is ignored due to the lack of information. This threshold is in compliance with the work of Merkl et al.

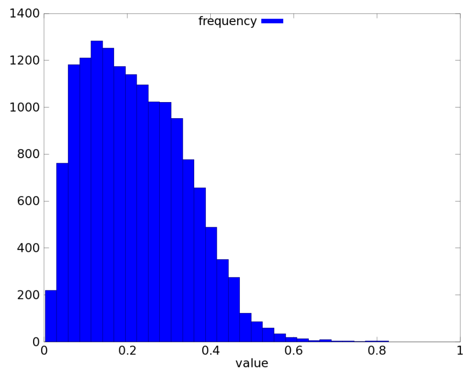
5.1.3 Identification of Statistically Significant Values

We have observed a correlation between the average level of the NMI values and the number of species in the MSA, or the depth of the MSA. Larger MSAs tend to have lower NMI values than smaller MSAs. So we need a MSA independent threshold to determine what is a significant value. Using the MSA depth itself is not an option as the correlation between depth and NMI values is not consistent enough. This observation adds to the assumption that there are several levels of coevolutionary pressure between residues. Although a high level of coevolutionary pressure exists between functional important residues, we assume that a general level of coevolutionary pressure exists between the majority of residues.
If we regard the NMI values of a given MSA as random variables, we are enabled to use statistical analysis. To show an exemplary NMI value distribution, a frequency histogram of the three different NMI variants for the protein and MSA Pyrrolidone carboxyl peptidase from thermococcus litoralis (PCPTL) is shown in Figure 5.1.
To be able to find an independent threshold, we need to make assumptions about the distribution of the NMI values. This requires a null hypothesis, which is the synonym for the general model for the data. In this case it stands for the coevolutionary pressure between non-significant residues as opposed to the model of coevolutionary pressure between significant residues.

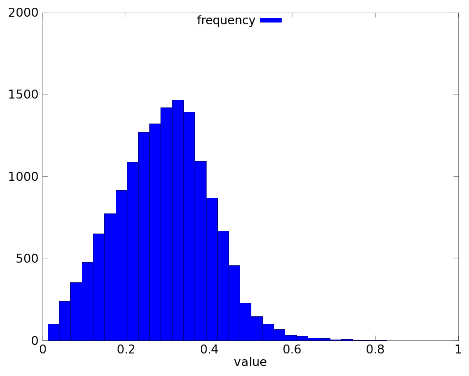
We tested several approaches to build a null hypothesis, with non-satisfactory results. Thus, we decided upon a combination of two of these approaches to build our null hypothesis. These are detailed in the following two sections.



(a) U



(b) NMI_{min_a}



(c) NMI_{min}

Figure 5.1: Histograms of U, NMI_{min_a} and NMI_{min} values for the protein PCPTL

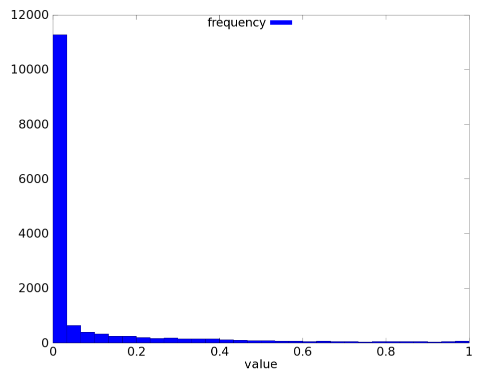
Artificial Column Based Approach

The first approach is to generate artificial columns depending on the given MSA and calculate a another set of $\overline{NMI}$ values, with the assumption that these artificial columns have no significant value. We decided that we want to create the null hypothesis out of the MSA itself instead of using the approach with a known distribution. This approach is better suited to compensate the differences between the individual MSAs, and there is no known distribution which models the NMI values distribution.

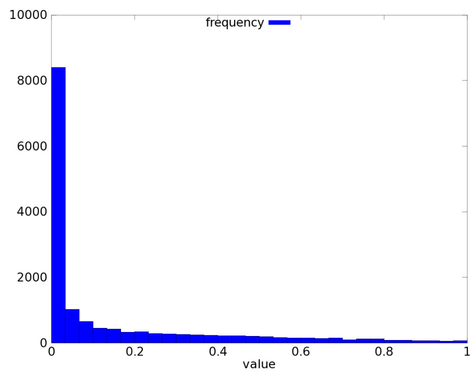
Therefore, the basic approach is to create new columns in dependance of the given MSA and then calculate new NMI values. For each column pair and their NMI value we need the respective p value. A p value describes the probability that the observed NMI value can be obtained by choosing a random column pair, assuming the pair belongs to the null hypothesis.
Let $\overline{NMI}_j$ for $j = 1, \ldots, n_0$ be the resulting null values from the artificial columns. Then p values can be calculated for all observed NMI_i for $i = 1, \ldots, n$ of a given MSA as:

$$p_i = \frac{\#\{\overline{NMI}_j \geq NMI_i, j = 1, \ldots, n_0\}}{n_0}$$

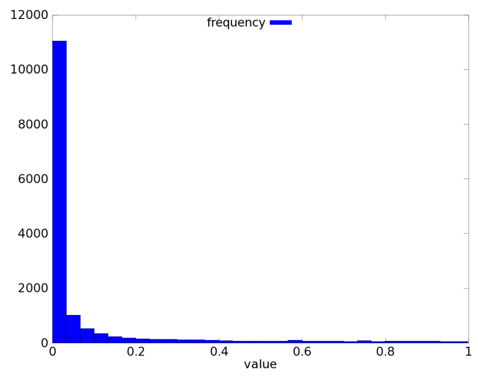
These p_i are used to make assumptions about significantly high NMI values. What we need is an exact method to create these new artificial columns. One procedure is to permute one of the columns. This would destroy any link between the columns [BSB+10]. This is a standard procedure, if a model should represent none or only a very low coevolution. The new $\overline{NMI}$ values however are so low that any column pair with at least a bit of correlation would be determined significantly higher than this distribution, which makes this procedure unsuitable for our needs. These permuted column pair have too low values of $\overline{NMI}$, which are reflected in their corresponding p values (Figure 5.2).
Another procedure is to create artificial columns by using probabilities taken from a given pair of columns. This would create high correlations and the resulting $\overline{NMI}$ values tend to be similar to the original NMI.
Neither of these two procedures identifies significant values as we envision them. The first procedure rates almost everything as significant. The second approach has an almost uniform distribution of p values making it impossible to distinguish the significant values.



(a) U



(b) NMI_{min_a}



(c) NMI_{min}

Figure 5.2: Histograms of p values, resulting from the permutation null hypothesis for U, NM_{min_a} and NM_{min}, using the protein PCPTL as an example

Transformed Beta Approach

The second approach assumes a well known distribution and calls all values significant that do not fit this distribution or deviate from the expected value by a large margin. As stated above, we believe that there is a general coevolutionary pressure between residues. This means that we have to model our null hypothesis to reflect this background pressure which is specific to a given MSA.
We have observed in the permutation approach that not only the p values are generally very low, but also the range is much lower than the range of the original p values. The reason for this is that the permutation approach almost exclusively reflects a non existent coevolution, while the original approach reflects all variants of coevolutionary pressure.

To counter this effect we use an approach in two steps. The first step is to create a matrix TM which reflects the substitution probabilities in a given MSA. Here we count all substitutions including the gaps, as we want to make the artificial columns reflect the build of the given MSA. We simply count the substitution in a column from residue t to residue $t+1$. Thus we preserve the probabilities of the substitutions and the order of the MSA sequences. The algorithm 1 creates this matrix. The probabilities are calculated by

$$P(X = x_i|X = x_j) = \frac{P(X = x_i, X = x_j)}{P(X = x_j)}$$

Algorithm 1 Amino acid substitution counting

$code(a, b)$ returns an integer representation for the amino acid or gap in position a of a sequence b.

```
Initialize an integer array M[21][21]
for all columns $l$ of the MSA do
    for sequences $s = 2, \dots, m$ of the MSA do
        $x \leftarrow code(l, s)$
        $y \leftarrow code(l, s-1)$
        M[x][y]++
    end for
end for
```

To create a column pair of the null hypothesis we choose one actual column and create an artificial one based on the matrix TM. Creating a column with our TM starts with drawing the first amino acid uniformly from $\mathfrak{A}$. The following $m-1$ residues are drawn by a Markov process with probability $p(x_j|x_{j-1})$ for $j = 2, \dots, m$, where m is the depth of the MSA. It is unlikely to get a pair with a high coevolution value, but the $\overline{NMI}$ values are generally higher than those from the permutation model. And the $\overline{NMI}$ values are lower than the original NMI values.
The second step is to combine the artificial $\overline{NMI}$ values with the real empirical expected value.
For this, we need a distribution for our NMI values We observed that most of the NMI histograms resemble a *beta distribution* [DH04]. Hence, it stands to reason to use a beta distribution in order to model the null hypothesis.

The beta distribution is defined over $[0,1]$, with parameters α and β and probability density function

$$f(x;\alpha,\beta) = \frac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}.$$

The beta distribution has an expected value of

$$\mu = \frac{\alpha}{\alpha+\beta}$$

and a standard deviation of

$$\sigma = \sqrt{\left(\frac{\alpha\beta}{(\alpha+\beta+1)(\alpha+\beta)^2}\right)}.$$

After the creation of the artificial columns and the calculation of the $\overline{NMI}$ values we can now determine empirically the expected values and standard deviations for the NMI and the $\overline{NMI}$ values.
Let $\hat{\mu}$ be the average of real NMI values and $\hat{\sigma}$ their standard deviation. Respectively let $\overline{\mu}$ be the average of the $\overline{NMI}$ values and $\overline{\sigma}$, their standard deviation. If we assume an underlying beta distribution, we can calculate parameters α and β as follows:

$$\alpha = \hat{\mu}\left(\frac{\hat{\mu}(1-\hat{\mu})}{\overline{\sigma}^2} - 1\right)$$

and

$$\beta = (1-\hat{\mu})\left(\frac{\hat{\mu}(1-\hat{\mu})}{\overline{\sigma}^2} - 1\right)$$

As $\alpha, \beta > 0$ for all tested MSAs, we can use the *BN algorithm* [AD74] (see algorithm 2) to draw sample values from a beta distribution.
The BN algorithm uses the relation between *Beta* and *Gamma* [DH04] distributions: $B(\alpha,\beta)$ variables can be represented by $\frac{X}{X+Y}$ with $X \sim \Gamma(\alpha,1)$ and $Y \sim \Gamma(\beta,1)$. X and Y are required to be independent of each other.

Lemma 5.1
For $\alpha > 1$, $\beta > 1$ and $0 \leq x \leq 1$, it is:

$$\left(\frac{x}{\alpha-1}\right)^{\alpha-1}\left(\frac{1-x}{\beta-1}\right)^{\beta-1}(\alpha+\beta-2)^{\alpha+\beta-2} \leq \exp\left(-\left(x-\frac{\alpha-1}{\alpha+\beta-2}\right)^2 2(\alpha+\beta-2)\right)$$

The left side of the inequality is proportional to the density function of a $B(\alpha,\beta)$ distribution and the right side is proportional to the density function of a $N(\frac{\alpha-1}{\alpha+\beta-2},(\frac{1}{2\sqrt{\alpha+\beta-2}})^2)$ distribution. The sampling from the normal distribution and the exit condition

$$\ln u \leq A \cdot \ln(x/A) + B \cdot \ln((1-x)/B) + L + 0.5 \cdot \sigma^2$$

of the BN algorithm are consequences of Lemma 5.1.

Algorithm 2 BN

"Generate u" means sampling from a uniform distribution in $[0,1]$ and "Generate s" means sampling from a standard normal distribution.

```
Require: $\alpha > 1$ and $\beta > 1$ as input.
  $A \leftarrow \alpha - 1$
  $B \leftarrow \beta - 1$
  $C \leftarrow A + B$
  $L \leftarrow C * \ln C$
  $\mu \leftarrow A/C$
  $\sigma \leftarrow 0.5 \cdot \sqrt{C}$
  repeat
    repeat
      Generate $s$
      $x \leftarrow s \cdot \sigma + \mu$
    until $x \in [0,1]$
    Generate $u$
  until $\ln u \leq A \cdot \ln(x/A) + B \cdot \ln((1-x)/B) + L + 0.5 \cdot \sigma^2$
  return $x$
```

We can draw new $\overline{NMI}$ values from a beta distribution with the usage of the parameters α and β. This procedure transforms the artificial values with the real empirical expected values which was the goal as stated above. A similar modus operandi, using an artificial standard deviation and real expected values is used to create z scores for MI values and permuted MSA columns by [BSB$^+$10]. This modus was used by Bremm et al. to create z scores for MI values and permuted MSA columns.

An example histogram for the p values resulting from this approach is shown in Figure 5.3.

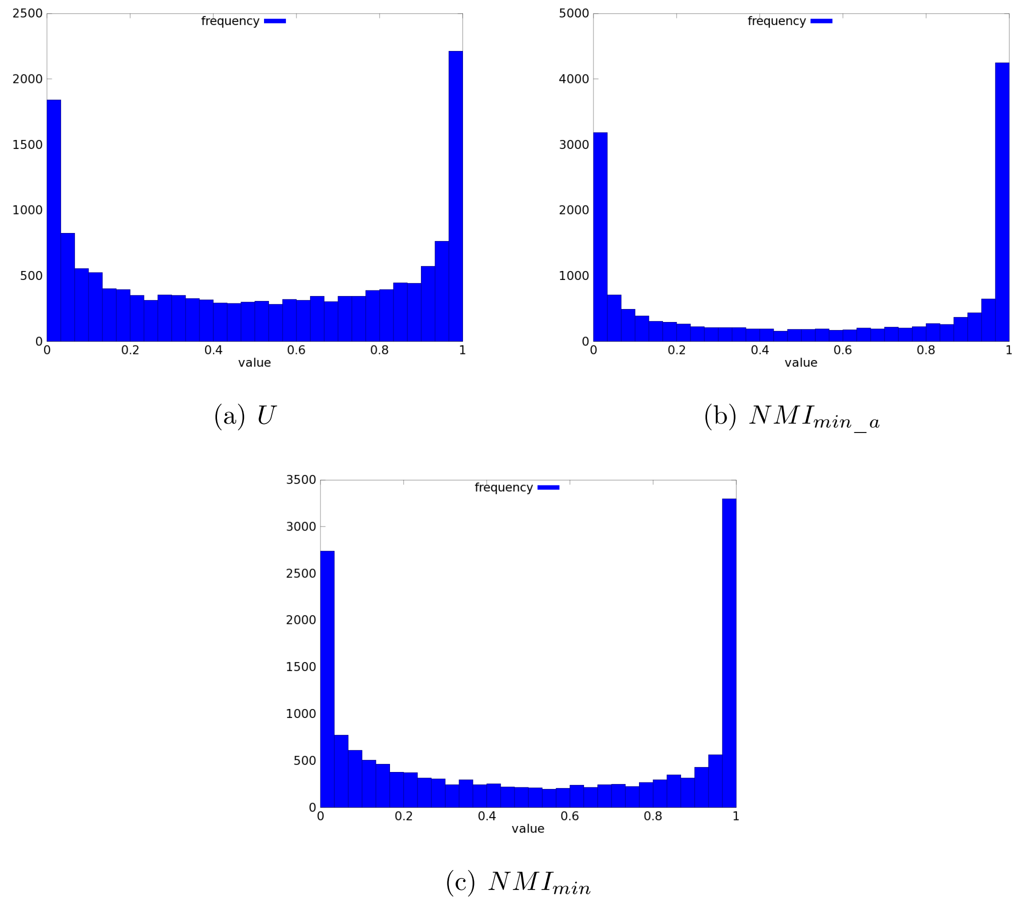


Figure 5.3: Histograms of p values, resulting from the transformed beta null hypothesis for U, NMI_{min_a} and NMI_{min}, using the protein PCPTL as an example

False Discovery Rate

The standard approach for hypothesis testing, involving p values and a significance level α, is too inflexible though to be used with success on our problem. We need a more dynamic approach to determine a threshold for the significant residue values.
Our goal is to find a threshold which separates as many true significant features from the rest while only including a small amount of non significant features. This ratio of wrongly classified features is the *false discovery rate* (FDR). Thus, we want to determine our threshold for significant p through an estimation based on a fixed false discovery rate.

The work of Storey and Tibshirani [ST03] gives such an estimation which we have used for our work.
Let S be the number of all features which are rated as significant. Then we designate T as the subset of S with true significant values and F as the subset of S with non significant values. Based on these variables the false discovery rate FDR is:

$$FDR = \frac{F}{F+T} = \frac{F}{S}$$

If we have a threshold τ established, a feature is called significant, if the corresponding p value is equal or less than τ. We designate n as the amount of all features.
We have a different possibility to calculate our FDR in dependency to τ as well as our sets S and F:

$$FDR(\tau) = E\left(\frac{F(\tau)}{S(\tau)}\right)$$

with

$$F(\tau) = \#\{null p_i \leq \tau, i = 1, \ldots, n\}$$

and

$$S(\tau) = \#\{p_i \leq \tau, i = 1, \ldots, n\}.$$

This calculation only works though, if we already know whether a feature is significant or not. Hence, it is useful to quantify the performance of a chosen threshold; but we want to reverse this process and chose the threshold through a fixed FDR.

For large n it can be shown that:

$$FDR(\tau) = E\left(\frac{F(\tau)}{S(\tau)}\right) \approx \frac{E(F(\tau))}{E(S(\tau))}$$

Let n_0 be the amount of truly non significant features and τ a random but fixed threshold in the range of $[0, 1]$. $E(S(\tau))$ is the same as $S(\tau)$ and can be calculated by applying τ to all features. The problem is to estimate $E(F(\tau))$, as under normal circumstances we do not know which features are non-significant.
We can exploit the fact that the p values belonging to the null hypothesis are uniformly distributed. This fact can be used to estimate the ratio of non-significant features $\pi_0 = \frac{n_0}{n}$. As we expect at least some p values to be significant, the distribution of all p values should not be uniform. The features which are significant and thus, do not belong to the null hypothesis have very low p values near or even equal to zero.

If we have ideal circumstances, a histogram of the complete p values should show a peak near zero and be completely uniform for the rest. The part where the distribution is uniform, can be used to estimate π_0. For this we also need a tuning parameter $\lambda \in [0, 1]$:

$$\hat{\pi}_0(\lambda) = \frac{\#\{p_i > \lambda; i = 1, \ldots, n\}}{n(1-\lambda)}.$$

The distribution of the p values only has the necessary uniform portion, if the π_0-values are relatively stable beyond a fixed λ. For our purposes choosing $\lambda = 0.65$ provided the necessary stability.
At this point, we can estimate n_0 through π_0 and λ via the following equation:

$$\hat{n}_0(\lambda) = n\hat{\pi}_0(\lambda) = \frac{\#\{p_i > \lambda; i = 1, \ldots, n\}}{(1-\lambda)}.$$

As we have our estimation of $E(F(\tau)) = \tau \cdot n_0$, we are able to estimate the FDR:

$$\widehat{FDR} = n_0 \frac{\tau}{S(\tau)} = \hat{n}_0 \left(\frac{\tau}{\#\{p_i \leq \tau; i = 1, \ldots, n\}} \right).$$

If we let τ run from zero to one and calculate our FDR estimation, we can then determine our threshold as soon as we have reached a desired FDR.
Although we have devised a method to determine a threshold, we still face one problem. Our method to create a null hypothesis has not the desired form we need to apply the FDR method. As stated above, we expect an almost ideal distribution with one peak near zero, our null hypothesis has two peaks, one near zero and one near one.
The peak near one consists of very low NMI values which are unimportant for our purposes. Therefore, we have to modify the FDR method to make it applicable to this situation. This second peak affects the π_0 estimate, hence, we need to add an adjustment at that point. This is done by splitting the tuning parameter λ into two boundaries, so they encompass the uniformly distributed portion of the p values. We set $\lambda' \in [0, 1]$ as the lower boundary and $\lambda'' \in [0, 1]$ as the upper boundary with $\lambda' < \lambda''$. We can calculate the new π_0:

$$\hat{\pi}_0 = \frac{\#\{p_i \in [\lambda', \lambda'']; i = 1, \ldots, n\}}{n(\lambda'' - \lambda')}$$

To be able to apply this procedure we need a fairly large number n of features, and not all MSAs are able to provide that number. A rough estimation is that we need an MSA to have at least 2500 pairs for this method to work. An example for a MSA which is too short is the MSA for the disulfide mutant of basic pancreatic trypsin inhibitor protein (DMPTI) shown in Figure 5.4.

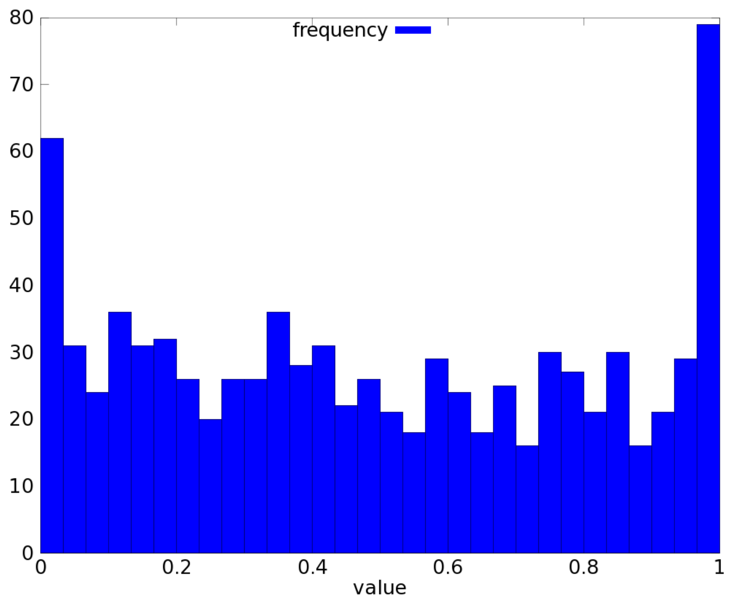


Figure 5.4: Histograms of p values, resulting from the transformed beta null hypothesis, if the sequence length is too short, using the protein DMPTI as an example

If we encounter a MSA which is too short, we revert to the method of simply picking the top 75 pairs as significant. The transformed beta approach has some minor flaws, but has so far proven to be the best approach applied so far. It reflects the desired view of co-evolution better than for example the permutation approach, where almost all pairs are significant due to the very low p values of the null hypothesis.

5.1.4 Selection of MSAs

To ensure a certain quality in the estimation we have set a few conditions for the MSAs that can be used. As the results are used as a means to detect significant residues, we want to avoid as many errors from unreliable data as possible.
The MSAs or the respective protein files need a minimum of 125 sequences, so the NMI values are not calculated on very few residues. Furthermore, we left out protein files, where the length of the chains are different. As final safeguard we removed the protein files where the protein chain is not matched exactly in the MSA.
After applying these criteria, the MSAs of 667 proteins remain. If we are able to use the transformed beta approach (MSA has over 2500 pairs), we choose to set the parameters as follows: $\lambda' = 0.2$ and $\lambda'' = 0.8$ and a desired FDR of 10%.

5.2 Amplifying Significant Residues

We want to devise a method to be able to transform the NMI values to amplify the signal from highly connected and thus significant residues. This will be done through bistochastic matrices for each normalization variant that are calculated from the results of Section 5.1.

Due to the varying sizes of the MSAs the number of significant pairs also varies substantially. In order to prevent overfitting we decided to choose a fixed amount of significant pairs from each MSA. This amount is set to 75 to be in line with our selection of MSAs (see Section 5.1.4), these are randomly picked out of all significant pairs. We want explicitly to not pick the top pairs from each MSA, since we want a representation of the general characteristics of significant pairs and the top pairs are likely extremes which would falsify our results.

From these column pairs we build 3 substitution matrices: one for each NMI variant. As a contrast, and to be able to have a measure of comparison, we also build a matrix out of 75 random pairs from each MSA.

5.2.1 Substitution Matrices

The concept of a pair-to-pair substitution matrix is adapted from the work of Pietrokovski et al. [EP07], the matrices are based on the pair substitutions frequencies. Pietrokovski et al. use sequence weights based on the similarity of the sequences to measure the significance of each substitution ([HHH94]). We omitted this measuring process, because our MSAs are already filtered for similarity and our matrices are based on the 75 chosen pairs and not on the whole MSA.

Creating the Substitution Matrices

Each MSA and pair is counted separately, therefore, we have an accurate profile of the substitutions. The method to count one pair of residues is described below and algorithm 3 implements this method for all MSAs and their respective 75 pairs.

We identify a residue pair r in its respective alignment by the columns which we denote with l and k, $l \neq k$. We have our amino acid alphabet $\mathfrak{A}$ and define the alphabet $\mathfrak{P}$ as the pair alphabet with $x, y \in \mathfrak{A}$ which results in the size of $\mathfrak{P}$ being: $|\mathfrak{P}| = 400$.

We can regard $\mathfrak{P}$ as an undirected weighted Graph $G_{1,k}$ where our set of vertices consists of $\mathfrak{p}_1, \dots, \mathfrak{p}_{400} \in \mathfrak{P}$. We initialize $G_{1,k}$ as a complete graph where all edges have the weight zero. By adding weight to the correct edge we can count pair substitutions.

Considering our MSA M, we denote with m the number of sequences in M. Let $s, t = 1, \dots, m$ be two random sequences in M, with $s \neq t$ and let x, y be two different columns in M then we denote with $\mathfrak{p}_s(x_s, y_s) = \mathfrak{p}_i$ and $\mathfrak{p}_t(x_t, y_t) = \mathfrak{p}_j$ the observed residue pair in each sequence.

We count pair substitutions symmetrically for each column pair. Thus, a pair substitution $\mathfrak{p}_i \to \mathfrak{p}_j$ is accountable for four single substitutions:

$$\mathfrak{p}_s(x_s, y_s) = \mathfrak{p}_i \to \mathfrak{p}_t(x_t, y_t) = \mathfrak{p}_j$$

$$\mathfrak{p}_s(y_s, x_s) = \mathfrak{p}_u \to \mathfrak{p}_t(y_t, x_t) = \mathfrak{p}_v$$

$$\mathfrak{p}_t(x_t, y_t) = \mathfrak{p}_j \to \mathfrak{p}_s(x_s, y_s) = \mathfrak{p}_i$$

$$\mathfrak{p}_t(y_t, x_t) = \mathfrak{p}_v \to \mathfrak{p}_s(y_s, x_s) = \mathfrak{p}_u$$

As we have an undirected Graph, it follows that $\mathfrak{p}_i \leftrightarrow \mathfrak{p}_j$ specifies the same edge as $\mathfrak{p}_j \leftrightarrow \mathfrak{p}_i$. $\mathfrak{p}_u \leftrightarrow \mathfrak{p}_v$ and $\mathfrak{p}_v \leftrightarrow \mathfrak{p}_u$ also specify the same edge. Each of these two edge weights is thus increased by two.
We count the substitutions symmetrically, because we do not know the exact order between the sequences. To be able to predict the correct order would have meant including a phylogenetic algorithm which is used to estimate the order of the sequences by relation to each other. These algorithms are not always exact and thus would add another measure of uncertainty into our model, which we felt was unnecessary [Hil95]. Thus we regard the order of the MSA as random, therefore, each sequence to sequence transition is treated equally and all substitutions are counted symmetrically.
For each sequence $s = 1, \dots, m$ the substitutions are counted for all substitutions into sequences $t = 1, \dots, m$ with $s \neq t$. If there is a gap in any of the observed columns in a sequence x, that pair is ignored for all substitutions.

Definition 5.2
For one MSA M and columns $l, k \in S$ where S is a set of residue pairs, let $A_{(l,k)}$ be the 400×400 adjacency matrix of $G_{(l,k)}$, then

$$C_M := \sum_{(l,k) \in S} A_{l,k}$$

is the pair substitution matrix for MSA M.

We can calculate the sum of the matrices C_M for all MSAs M as:

$$C := \sum_{M=1}^{r} C_M.$$

Because we do this four times for all three NMI variants and once for random pairs we get four matrices as result. These consist of the absolute count of all substitutions in the respective pairs for each NMI of all MSAs. C_U, $C_{NMI_{min}}$, $C_{NMI_{min\_a}}$ and C_{random}.

Algorithm 3 Pair substitution counting

code(a, b) returns an integer representation for the amino acid in position a of a sequence b and -1 if there is a gap.

```
Initialize an integer array M[20][20][20][20]
for all MSAs do
  for all column pairs $l, k$ in set P do
    for all sequences $s$ of the MSA do
      $u \leftarrow code(k, s)$
      $v \leftarrow code(l, s)$
      for all sequences $t$ of the MSA do
        $x \leftarrow code(k, t)$
        $y \leftarrow code(l, t)$
        if $u \neq -1 \& v \neq -1 \& x \neq -1 \& y \neq -1$ then
          M[u][v][x][y]++
          M[v][u][y][x]++
          M[x][y][u][v]++
          M[y][x][v][u]++
        end if
      end for
    end for
  end for
end for
```

The Scoring Function

As we calculated the substitution matrix, we want to rate the chance of a substitution between two pairs. This way we can focus on the important substitutions which indicate coevolution and ignore those that happen randomly.
Considering a substitution matrix C we define q as the amount of all substitutions in C:

$$q := \sum_{i=1}^{400} \sum_{j=1}^{400} c_{ij}$$

We can calculate the empirical probability of pair substitution $\mathfrak{p}_i \rightarrow \mathfrak{p}_j$ by:

$$\hat{p}(\mathfrak{p}_i, \mathfrak{p}_j) := \frac{c_{ij}}{q}$$

From the joint probabilities we can derive the marginal probability $p(\mathfrak{p}_i)$ for $\mathfrak{p}_i$ by:

$$p(\mathfrak{p}_i) := \sum_{j=1}^{400} p(\mathfrak{p}_i, \mathfrak{p}_j)$$

$p(\mathfrak{p}_j)$ for $\mathfrak{p}_j$ is calculated in the same way.
We need to decide if a pair substitution occurred at mere random or if its probability is significant. Hence, we have the null hypothesis for a pair substitution with probability $p(\mathfrak{p}_i) \cdot p(\mathfrak{p}_j)$, which reflects the random occurrence.
To be able to rate a substitution as significant, we need a scoring function, which rates the substitution. We do this by comparing $p(\mathfrak{p}_i, \mathfrak{p}_j)$ to $p(\mathfrak{p}_i) \cdot p(\mathfrak{p}_j)$

Definition 5.3
We define the scoring function $\mathfrak{s}(\mathfrak{p}_i, \mathfrak{p}_j)$ for all pair to pair substitutions $(\mathfrak{p}_i, \mathfrak{p}_j)$ as:

$$\mathfrak{s}(\mathfrak{p}_i, \mathfrak{p}_j) := \log\left(\frac{p(\mathfrak{p}_i, \mathfrak{p}_j)}{p(\mathfrak{p}_i)p(\mathfrak{p}_j)}\right).$$

If our scoring function $\mathfrak{s}(\mathfrak{p}_i, \mathfrak{p}_j)$ has a positive value, it is an indication that the $\mathfrak{p}_i \rightarrow \mathfrak{p}_j$ is significant as opposed to random chance.

5.2.2 The Final Pair Substitution Scoring Matrices

We want to remove all non-significant substitutions from our pair substitutions matrices. Thus, we apply a three step revision process.
We use the joint probabilities for all steps due to the fact that the number of pairs are equal in each matrix, but due to gaps in the columns the numbers can differ. The joint probabilities are independent from the absolute number so they ensure a stable comparable environment.
In the first step, we compare each of the NMI variant matrices to the random matrix. This ensures that we only keep the substitutions which are more likely to appear in the significant pairs than in random pairs. Therefore, each element $M(NMI)_{ij}$ of a significant pair matrix is compared with the $M(random)_{ij}$ respective pair in the random matrix and if $M(NMI)_{ij} < M(random)_{ij}$, we set $M(NMI)_{ij} = 0$. This is done for all $i, j \in 1, 2, ..., m | i \neq j$. A similar process is done by Pietrokovski et al. in [EP07], though in the work of Pietrokovski et al. the random values are subtracted from the significant values.

The second step is to eliminate non *compensatory mutation*. Compensatory mutation is a mutation where not chemically similar amino acids substitute each other. A substitution between chemically similar amino acids is not assumed to be a strong indicator for co-evolution.
To be able to detect chemically similar amino acids we employ the use of *BLOSUM matrices*. BLOSUM matrices were introduced by Henikoff et al. [HH92] and apply a $log - odds - ratio$ function to rate the similarity of amino acid substitutions. They are generated based on the BLOCKS database [PH95], which contains blocks of MSA without gaps. BLOSUM N matrices can be calculated for a range of $N = 50, \dots, 80$, where N is the percentage of equal residues between any two sequences of each block. If the percentage of equal residues among any of these sequence pairs exceeds N, one of these sequences is ignored. A substitution between not chemically similar amino acids has a negative BLOSUM score.
We use the BLOSUM62 (Table 5.2) matrix to define compensatory mutations, since this BLOSUM table is considered to be very robust and generally used for similar applications [WM09].

Ala	4																			
Arg	-1	5																		
Asn	-2	0	6																	
Asp	-2	-2	1	6																
Cys	0	-3	-3	-3	9															
Gln	-1	1	0	0	-3	5														
Glu	-1	0	0	2	-4	2	5													
Gly	0	-2	0	-1	-3	-2	-2	6												
His	-2	0	1	-1	-3	0	0	-2	8											
Ile	-1	-3	-3	-3	-1	-3	-3	-4	-3	4										
Leu	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4									
Lys	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5								
Met	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5							
Phe	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6						
Pro	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7					
Ser	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4				
Thr	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5			
Trp	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		
Tyr	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	
Val	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4
	Ala	Arg	Asn	Asp	Cys	Gln	Glu	Gly	His	Ile	Leu	Lys	Met	Phe	Pro	Ser	Thr	Trp	Tyr	Val

Table 5.2: BLOSUM62 matrix

We employ this in our revision step by checking for each pair substitution, if both sides have a negative sign in the BLOSUM matrix. If this is not the case, the substitution value is set to zero. One example is the pair substitution $Ala \times Ala \rightarrow Val \times Tyr$. Ala and Tyr have a negative sign according to the BLOSUM62 matrix, thus, it is still a valid candidate. But the substitution Ala and Val is neutral, and thus the pair substitution is set to zero. Since the sign is always positive for a substitution of an amino acid with itself, elements $a_{ij}, i = j$ are ignored.

The third step is to apply our scoring function. As we have changed the sum of elements in the NMI variant matrices, we need to recalculate all probabilities for use in our scoring function. If $\mathfrak{s}(\mathfrak{p}_i, \mathfrak{p}_j) < 0$, we set the respective pair substitution to zero. Negative values would indicate that the substitution happens at mere random and is not significant, therefore, we want to eliminate it from our matrix.

5.2.3 Bistochastic Matrices

After the revision of our NMI variant matrices we want to calculate a bistochastic matrix for each of the variants.

Definition 5.4
A matrix $BM \in \mathbb{R}^{n \times n}$ with elements $m_{ij} \geq 0$ is called bistochastic, if the sum of all columns k, $\sum_{i=1}^{n} a_{ik} = 1$ and the sum of all rows l, $\sum_{j=1}^{n} a_{lj} = 1$

To calculate our bistochastic matrices we use algorithm 4 introduced by Sinkhorn [Sin64]. This algorithm uses an arbitrary square matrix AS, which contains only positive elements. It then builds a bistochastic matrix BM by normalizing the rows and columns of AS. A further requirement is that the sum of the rows and column has to be greater than zero.

Algorithm 4 Sinkhorn

```
Require: $n \times n$ matrix $BM$ as input with $a_{ij} > 0$,
  $\sum_{i=1}^{n} a_{ij} > 0$ and $\sum_{j=1}^{n} a_{ij} > 0$ for $i, j = 1, \ldots n$
  repeat
    for i < n do
      for j < n do
        $a_{ij} \leftarrow a_{ij} \div \sum_{j=1}^{n} a_{ij}$
      end for
    end for
    for j < n do
      for i < n do
        $a_{ij} \leftarrow a_{ij} \div \sum_{i=1} a_{ij}$
      end for
    end for
  until $\|BM\| \leq 1 + c$
  return BM
```

c represents the desired accuracy and $\|.\|$ is a matrix norm.

After the revision our pair substitution matrix fulfills all requirements made by the Sinkhorn algorithm. After revisions are complete, all elements are greater or equal to zero and at least the elements of the principal diagonal are greater than zero. Note that these revisions eliminate the majority of entries in the bistochastic matrix. Depending on the variant, 75% - 85% of the entries are eliminated.

5.3 Residue Ranking

We have a method to identify significant pairs and to increase the strength of those signals through the bistochastic matrices. In this section, we want to give the means to separate the pairs and rank the single residues.

5.3.1 Finding Significant Pairs

The procedure to find significant residues has been shown in the Section 5.1, we just upgrade it with our results from Section 5.2.
We transform the joint distribution of amino residue pairs and their MSA columns by multiplication with the bistochastic matrix BM introduced in Section 5.2. This amplifies the signal of the significant residues. The transformation is done corresponding to the normalization variant:

$$(\overline{p}(x_1, y_1), \dots, \overline{p}(x_n, y_n))^T = ((1-\alpha)\mathbb{1} + (\alpha)M)(p(x_1, y_1), \dots, p(x_n, y_n))^T$$

The α is a weight factor that determines, how much of an effect this transformation should have. We calculate the marginal values for the transformed pair NMI variants as:

$$P(X = x_i) = \sum_{j=1}^{n} P(X = x_i, Y = y_j)$$

$$P(Y = y_j) = \sum_{i=1}^{n} P(X = x_i, Y = y_j)$$

We use our transformed beta approach and the false discovery rate as in Section 5.1 to deduce the significant pairs in these transformed NMI values.

5.3.2 From Significant Pairs to Significant Single Residues

This approach was introduced by Merkl in [MZ08] and we use the same method for our purposes. For each NMI variant, we have identified an amount of significant pairs. Let Sig be the set of all significant pairs for a NMI variant. Each significant pair $p \in Sig$ consists of two residues u, v.
We declare G as an undirected Graph with all u, v as vertices in G. We define that two vertices x, y in V_G have an edge between them, if x, y also form a pair in Sig: $u, v \in \{V_G |\, p(u, v) \in Sig\}$.

We define our ranking for each residue r through the degree of the respective vertex of r in G. This score is called $conn(r)$ for a residue r. It reflects, how often the residue r appears in the significant pairs and thus how often it is correlated within the set Sig. It stands to reason that a highly connected correlated residue is more important for the protein than a residue, which appears only once in all of the significant pairs.

5.3.3 Interface Residues

In general protein-protein-interfaces are considered to be an important subset of the significant residues. Thus, it is safe to assume, that interface residues are coevolved and thus detectable through our statistical approach.

NMI Values of Interface Pairs

The first task is to evaluate the general behaviour of interface NMI values. Hence, we calculate our NMI variants for the known interface positions in our database and compare these values against the average NMI values for the respective MSA. This is done without the transformation through the bistochastic matrix. The comparison is shown in Table 5.3.3.
The results show that interface position have a tendency to be rated higher than the average residue pair. This tendency is only slight and thus in itself not enough to be a good discrimination factor.

	# interface pairs > average	# total pairs > average
MI	5184 (62.26%)	7456512(44.14%)
U	5751(69.06%)	7940070(47%)
NMI_{min_a}	5009(60.15%)	7656792(45.32%)
NMI_{min}	54338(65.25%)	7456512(47.32%)

Table 5.3: Number of all pairs and interface pairs greater than the expected value of the corresponding MSA. The total number of interface pairs is 8327, and the total number of pairs is 16894420

If we check the example of the protein cystathionine gamma-synthase from Nicotiana tabacum (CGNT), we have five interface residues among the top 15 ranked residues out of the 83 interface residues overall in the protein CGNT. All the NMI variants share these interface residues in their respective top 15, despite the fact that we only have a 60% overlap in residues overall.
This is not a discriminator to predict interface positions, as it is difficult to separate interface positions from the functional important positions in a protein through correlated mutation. Our method can only detect that these positions are significant, not that they are interfaces, but this example leads to another approach to use this method.

Ranking of Known Interface Residues

To try a different approach, we want to use our method on already known interface positions and rank their NMI values with our score function $conn(r)$. This could help identify the more important residues of an interface. We assume that an interface residue with high $conn(r)$ score had more coevolutionary pressure than an interface residue $\overline{r}$ with a low $conn(\overline{r})$ score.

In general, we have a large overlap between all three NMI variants if we rank the known interface residues with $conn(r)$. As an example, we show chain A of the protein chicken Citrate Synthase complex (CCSC).
All three scores overlap in 11 of the 15 top ranking $conn(r)$ values, which are calculated only from the 78 known interface positions of the protein CCSC. The overlap is visualized in the Figure 5.5.

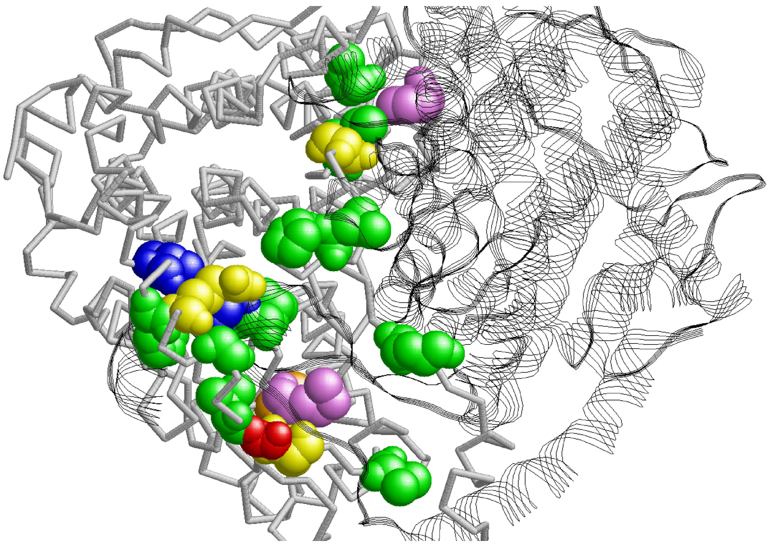


Figure 5.5: This figure shows the protein CCSC, chain A is plotted as gray backbone and chain B as black strands. The 15 highest $conn(r)$ interface residues for U, NMI_{min} and NMI_{min_a} are plotted in spacefill mode. Interface residues among all three NMI values are plotted in green, overlaps between U and NMI_{min} in orange and overlaps between U and NMI_{min_a} in violet. Interface residues without overlap are plotted in red (U), blue (NMI_{min_a}) and yellow(NMI_{min}). The residues are not labeled to provide better visibility.

All three NMI variants overlap in 11 positions. Additionally, U and NMI_{min} overlap in another residue and U and NMI_{min_a} share two more interface residues. U has only one residue without an overlap. NMI_{min_a} has two unique residues and NMI_{min} has three unique residues.

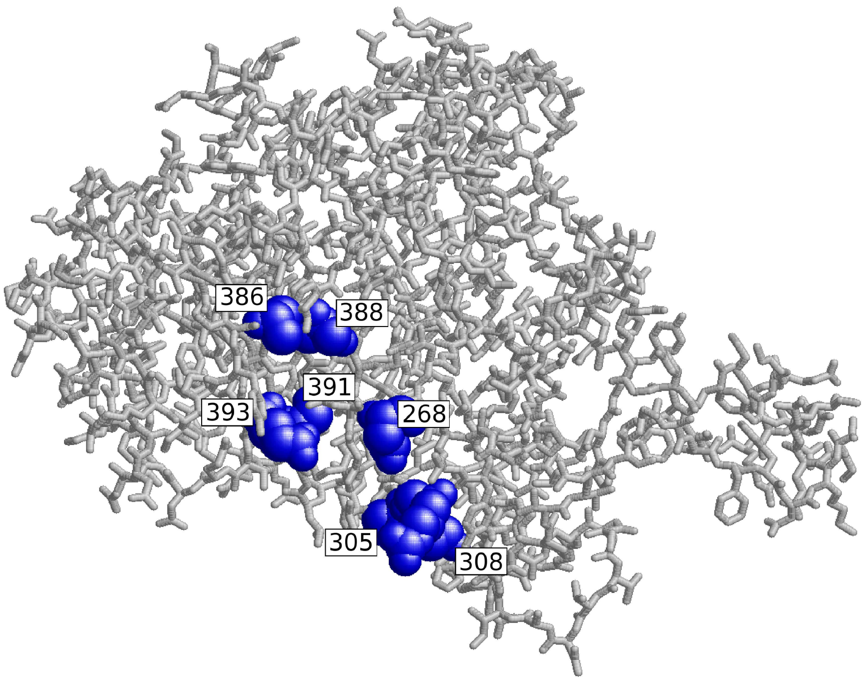


Figure 5.6: Protein CGNT chain A residues with high *conn*(r) scores

5.3.4 Assessment

As we drew some inspiration from the work of Merkl et al. [MZ08] and use the same ranking score, we have to compare our results to that work.
Merkl et al. use the protein CGNT as main example in their work. We focus on chain A of the protein CGNT, as all sequences of the protein CGNT are identical and thus, have identical MSAs. Their method was to calculate the U-values, take the 75 highest rating pairs and then use the *conn*(r) scoring function to rank the residues of the 75 pairs. The seven top *conn*(r) rated residues in [MZ08] are shown in Figure 5.6.
First we use our model without transformation through the bistochastic matrices. The results are shown in Figure 5.7, which depicts the 15 residues with leading *conn*(r) values. Notice that there is no overlap here; this will be explained later in this section.

Additionally, we use our approach with transforming the U-values with the bistochastic matrix first and then using the transformed beta approach. These results are shown in Figure 5.8, which again depicts the top 15 $conn(r)$ residues. Here we have one overlap with Merkl et al. original results at position 393, which is also an interface residue.

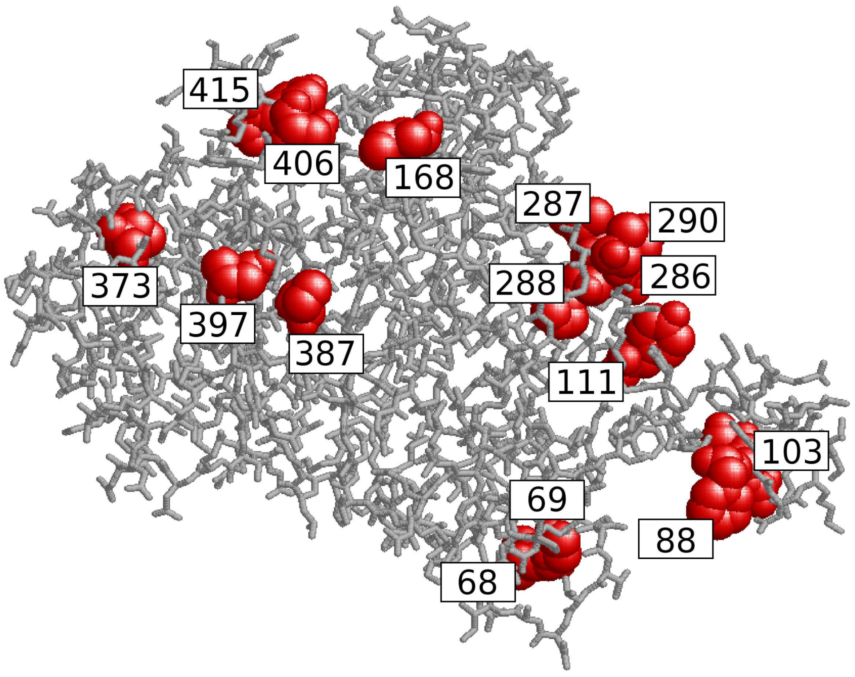


Figure 5.7: Protein CGNT chain A residues with high $conn(r)$ scores $conn(r)$ is calculated using pairs with significantly high U values

The small overlap between our U-values and Merkl et al. can be explained through differences in our methodologies. Through our transformed beta approach we have a much larger amount of significant residues than just the amount of 75 that Merkl et al. use. The $conn(r)$ values are highly dependent on this amount. Thus the general level of $conn(r)$ values is higher in our approach. Additionally, we depict the connections between the full set of significant features and not an extreme subset like in the work of Merkl et al.

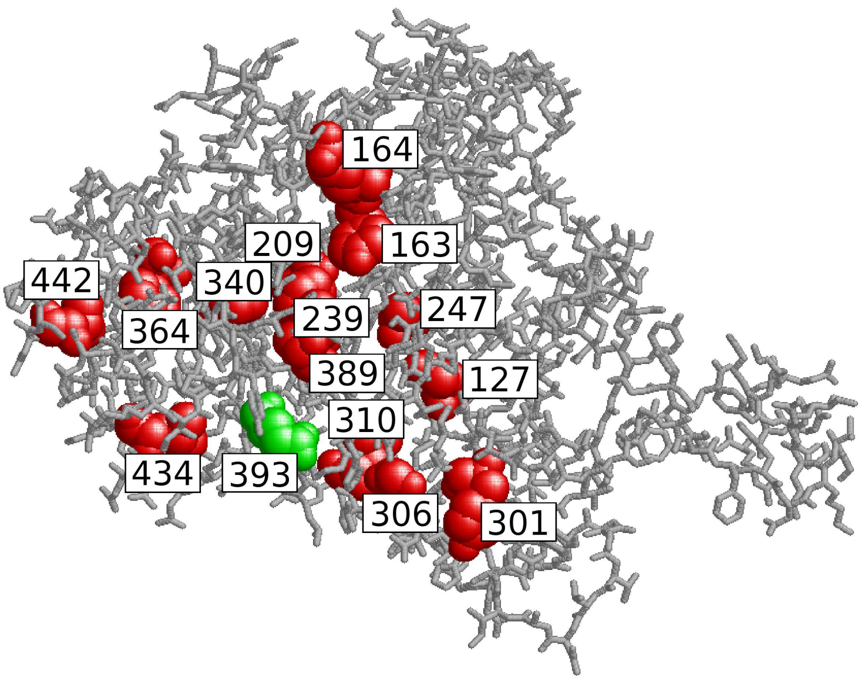


Figure 5.8: Protein CGNT chain A residues with high *conn*(r) scores *conn*(r) is calculated using pairs with significantly high transformed U values

As can be seen in Figure 5.8, the predicted residues are clustered in a specific area of the 3D structure of the protein and are not distributed randomly. Through use of the bistochastic matrix we detect similar residues as Merkl et al., but also find other significant residues. For example, in Figure 5.8 the residues 163, 239, 247 surround the cofactor pyridocal 5'-phosphate.

Another comparison was done with the work of Tinto et al. [TS08]. They analyzed Glucokinase for gene mutations, which can cause the maturity onset diabetes of the young disease. Tinto et al. identified 13 residues of the protein chain as mutation sites. Five of these were fully conserved throughout the corresponding MSA. Thus, they were not detectable by the method of Merkl et al. or by our approach, as these columns are filtered out as preprocessing step (Section 5.1). Using the scoring function and a calculated threshold for the *conn*(r) values the approach of Merkl et al. classifies one residue as significant. In our approach all eight residues are significant, but only four residues have a high enough *conn*(r) value to be identified as such.

The results show that we succeeded in making a robust and plausible mathematical model for the NMI approach. Especially the transformed beta approach, in contrast to the randomly fixed number of drawn pairs, is much more stable and based on well founded theories. The optional refinement with the bistochastic matrices can be used to adapt the classifier to various situations. It can be used not only to emulate the Merkl et al. model but also can be used to detect different significant residues which were not found by the former approach.
The different NMI variants can be used to express different views of correlated mutations as these are not fact, but more a matter of interpretation.

Chapter 6

Patch Classifier

Protein-protein-interaction is an important field of research in these years. As presented in Chapter 2.2.3 several theoretical methods for this problem exist. These use protein characteristics to make the distinction between a random surface reside and an interface residue. As shown by Brodag ([Bro08]), whose work was the primary motivation for this classifier, the distribution of residues in the interface differs from the distribution of surface residues. This information alone though does not qualify as a classifier since it is too unspecified. For a single residue on a protein chain a decision can be made based on the distribution if it is more likely to belong to the interface or the surface. This decision would not be very accurate due to the relatively small differences in the distribution. Table 6.1 shows the distribution values from our learning data set $\mathfrak{L}_{\mathfrak{hom}}$.

Amino Acids	A	C	D	E	F	G	H	I	K	L
Interface	0.055	0.011	0.056	0.068	0.042	0.063	0.03	0.048	0.066	0.078
Surface	0.065	0.007	0.085	0.102	0.018	0.081	0.024	0.025	0.1	0.047
Amino Acids	M	N	P	Q	R	S	T	V	W	Y
Interface	0.025	0.049	0.053	0.046	0.078	0.058	0.055	0.055	0.017	0.048
Surface	0.013	0.059	0.059	0.049	0.063	0.069	0.062	0.037	0.008	0.027

Table 6.1: The distributions of the different amino residues in $\mathfrak{L}_{\mathfrak{hom}}$

As can be seen, there are differences between the surface and interface amino residue usage. These differences are not significant enough to develop an accurate classifier based upon these distribution values.

While the distributions by themselves are not sufficient, we use them to devise a classifier that combines information about the residue distribution with information about the neighborhood of the residue. This is based on the theory that the neighborhood of interfaces plays an important role as well. It is a so-called "point classifier", since it is based upon a single residue (point) on a protein chain and not upon the interaction between two residues on two protein chains (edge). This combination led us to the assumption that it should be possible to distinguish between a random surface area of the protein and an area which consists mainly of interface amino acids. In these areas the amino acid distribution difference should be more distinctive and thus enough to accurately classify an area. These areas will be called *patches*.
To be able to discern these patches we need information about the protein structure. The main information we need is the protein chain itself, but we also need the tertiary structure information to be able to determine the spatial neighborhood of the residues on the chain.

6.1 Patch Definition

First of all, we have to define what a patch is on an amino acid chain, to be able to characterise and analyze it. The informal statement, that a patch is the neighborhood area of a single residue on an amino acid chain, is too imprecise to be used in an algorithmic approach. Our definition is based on the interpretation of a protein chain as a graph. Figure 6.1 shows chain A of the crystal structure of the neutral form of fructose-1,6-bisphosphate complexed with the product fructose 6-phosphate (NFFP), interpreted as graph.
We can start by defining a graph G_p of a protein chain P by interpreting all residues as nodes. Two nodes are connected by an edge if the two corresponding residues are neighbored, using our definition 4.2.
The graph G_p contains all protein residues, which includes the core residues. As noted in Section 2.2.2, interface residues are only part of the surface. Thus, we need to eliminate the core residues from the graph, to prevent negative impacts on the classification. Figure 6.2 shows the reduced graph of the protein NFFP. It also shows, that, with removing the core residues, the protein chain breaks down into several separate areas.
The reduced graph RG_P of a protein chain P consists of only surface residues as nodes. As above an edge is connecting two nodes, if the two residues are neighbored, according to our definition 4.2.

The following definition introduces a metric for distances on a the graph.

Definition 6.1
Let G be a connected graph, then we define $dist_G(v_1, v_2) \quad \big(v_1, v_2 \in V_G\big)$ as the number of edges on the shortest path between v_1 and v_2.

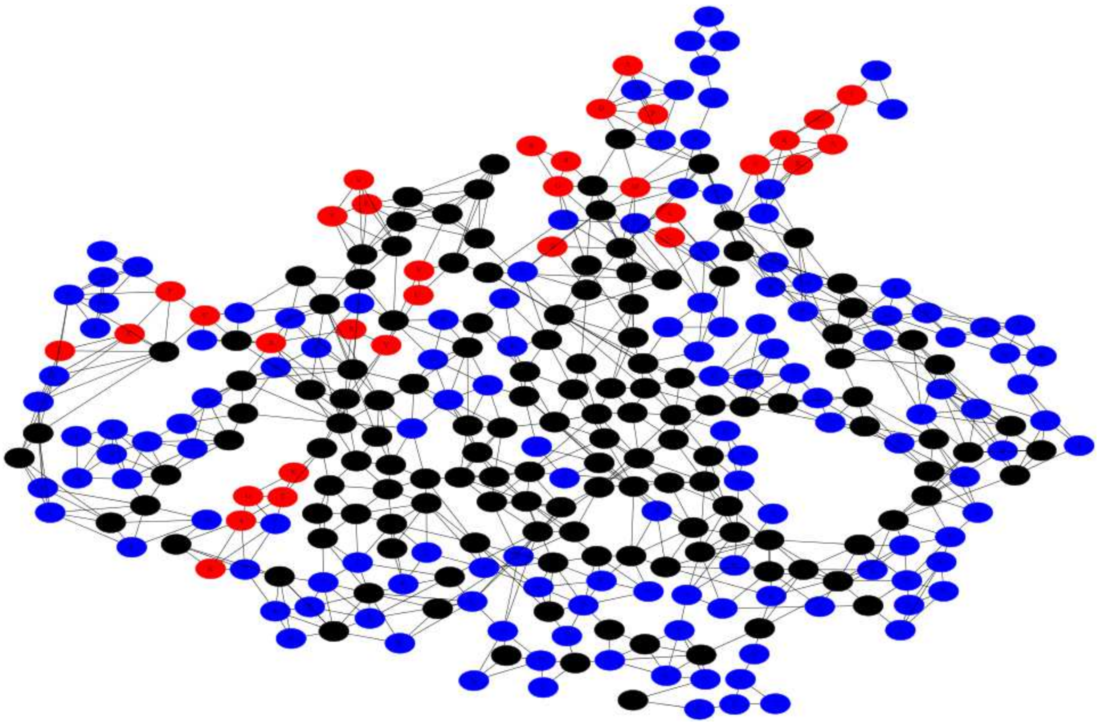


Figure 6.1: Chain A of the protein NFFP as graph; core is black, surface is blue, interface is red

Based on the reduced graph RG_P and the distance metric, we can formalize the definition of patches.
A patch Pat of depth d on a protein chain P around a seed residue sr consists of the residues which are within a distance of d of the node r on the reduced graph.
Because only the reduced graph without the core residues is used, the residues in a patch are only those which have an unbroken surface chain of edges to the seed residue.

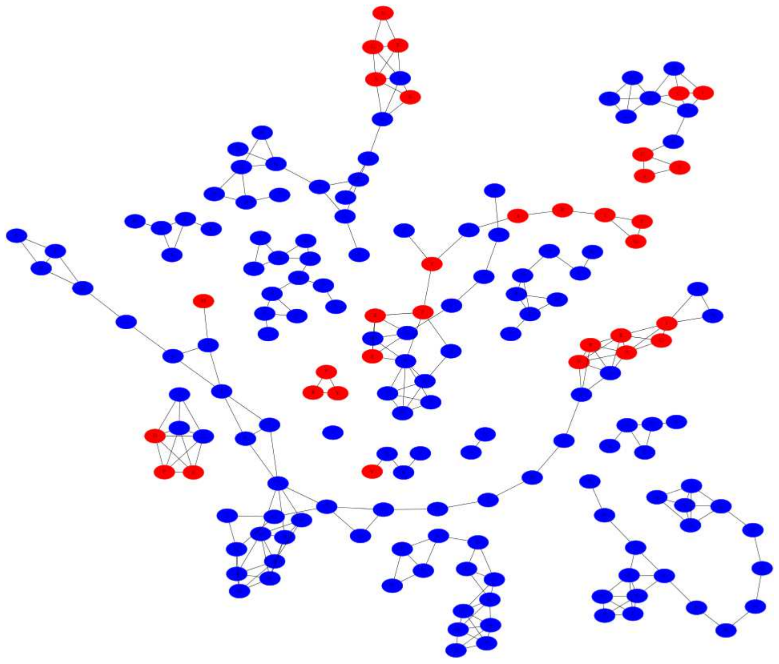


Figure 6.2: Chain A of the protein NFFP as reduced graph

6.2 Data Set

As data foundation we use the learning data set $\mathfrak{L}_{\mathfrak{hom}}$, as described in Chapter 4.3. However, for this particular classifier we need a slightly different version of our IFFs. This classifier is a point-classifier and is not based upon the interaction between two protein chains. Instead it is based upon the composition of only one chain. We need the residue, surface and neighbor information of the chain, which are calculated in the same way as in the original IFFs.
The interface residue information, however, derives not only from one interaction with one other chain but we identified all interactions to all other chains of the given PDB file. This leaves us with more interface residues than in the original IFF for each chain. This was done as the classifier, as such cannot distinguish between an interface residue which connects to protein chain A or which connects to protein chain B. Therefore, we need to mark all possible interface residues in the chain we are testing. To determine all interface residues, we calculate the interface residues of a chain not just for a single partner chain, but for all possible partner chains. This is called *multichain interfaces*. This is not uncommon for predictors, for example the Meta-PPISP prediction server [QZ07] also calculates multichain interfaces.
This process leads to 2330 files in the $\mathfrak{PL}_{\mathfrak{hom}}$ due to some files not being available due to MSA issues or other misgivings in the PDB.

6.3 Patch Composition

We have given a formal definition of a patch in 6.1. There are three possible residue compositions for a patch.
Due to the definition of the patch, core residues play no part in our patches. This leaves us with interface residues and random surface residues.
This leads us to three possible compositions:

- an Interface Patch (IP) consisting 100% of interface residues;
- a Surface Patch (SP) consisting 100% of surface residues which are not belonging to the interface;
- a Mixed Patch (MP) consisting of surface and interface residues.

The distribution of the patch types depends on the depth d of the patches. Table 6.2 shows how the distribution varies for depth 0-5 of $\mathfrak{PL}_{\mathfrak{hom}}$
A patch of depth zero is only the seed-residue itself, which explains the absence of mixed patches.

Depth	Interface Patches	pure Surface Patches	Mixed Patches
0	76684	274230	0
1	27588	222735	100591
2	14925	199004	136985
3	9691	181014	160209
4	7270	167035	176609
5	6021	155977	188916

Table 6.2: Amount of the different patch types for each depth for $\mathfrak{PL}_{\mathfrak{hom}}$.

The transition from depth 0 to depth 1 results in a severe loss of IPs (64%). The other transitions have a more steady decline in terms of IPs and SPs. The massive loss of IPs is explained by the interface residues which are not part of a larger interaction site, but singular extremities of the protein chain. For our model we only used the IPs and SPs as one hypothesis respectively null hypothesis. The reason for this is that these two models should have the most distinguished differences. Thus, if a classifier can be found on patches, it would work best on these two types of patches.

In Table 6.2, it is shown that for a depth of zero we have a ratio of approximately 1 IP to 2.7 SPs. Depth zero are only the seed residues, hence, this is the ratio we want to preserve for our testing to mirror these natural conditions. For the higher depths of the patches, we encounter a different ratio, as the amount of IPs is lower than the amount of SPs in comparison. Therefore, to preserve the original ratio of 2.7, we have to remove a number of SPs to have a sample which satisfies this ratio of 2.7.
Another reason to preserve this original ratio is that we want to avoid overfitting, when we use the PAC-learner to generate a hypothesis.

6.4 Procedure for Generating Patches

Generating patches for a given depth is a two-step procedure. First, we generate the IPs, because they only depend on the depth. Then, we generate a sample of SPs of the same depth, such that we have a ration of 2.7 IP to SP. This is done by randomly drawing SPs from all possible SPs, such that in the end we have a sample that upholds the ratio. Although this creates a varying sample, due to the randomness of the SPs, the results vary only slightly. This guarantees, that we do not base our theory on a single sample which might be completely artificial in its composition and so we, avoid overfitting.

We use algorithm 5 to generate our patches around a seed residue with a fixed depth. It produces an integer field of the length of the protein chain, where each residue that is part of the patch has the value one. At this point, we have two data sets, the interface patches and the corresponding random surface patches of the same depth. These are our positive and negative samples for the PAC learning algorithm, but they are not in a form which can be processed by the PAC algorithm.
During our experiments, we discovered an interesting effect. There exists a maximum depth after which the amount of interface patches does not decrease anymore. For the learning data set this boundary is depth 25, and for the test data set it is depth 22.

This algorithm requires a protein chain C of length n as additional input and uses the following methods:
$C[x]$ **is surface** is true if x is a surface residue for C.
$C[x]$ **is interface** is true if x is an interface residue for C.

Algorithm 5 Patch Generator

```
Set depth of patches $d$
Initialize integer array $P[n]$
Initialize integer array Patch[21]
for all residues $r$ of the protein chain $C$ do
  if C[$r$] is surface then
    $pos$ is position of $r$ in chain $C$
    $P[pos] \leftarrow 0$
    $i \leftarrow 0$
    repeat
      for $j < n$ do
        if $P[j] = i$ then
          Generate LN list of all neighbors from P[j]
          for all entries $m$ of LN do
            if $C[m]$ is surface then
              pos is position of $m$ in chain $C$
              $P[m] = i + 1$
            end if
          end for
        end if
        i++
      end for
    until $i = d$
  end if
end for
for $j < n$ do
  if $P[j] \geq 0$ then
    $P[j] = 1$
  end if
end for
```

6.4.1 Generating the PAC Samples from the Patches

For the calculation of a hypothesis from the PAC Learner (see Chapter 3.2.6) it is necessary to generate an input file containing positive and negative samples which follow a certain specification. In order to use the patches as input samples for our learning algorithm, we need to convert the patches into a PAC sample. We use five boolean values to denote the occurrence of each amino residue for a single patch. The first boolean value stands for 0 occurrences of that amino residue in this patch. The last boolean value stands for 4 or more residues of that particular type in this patch. Due to the size of the patches and the distribution of the amino residues, the maximal value of 4 is sufficient to create diversified samples for almost all patches. We used a number of 5 and 10 boolean values and gained the same hypothesis from the PAC algorithm each time. As the amount of boolean values has no impact on the results and the efficiency of the PAC algorithm is decreased by a large factor if the amount of boolean values is increased, we decided to use the smaller amount. Another negative factor is the increased dataspace needed to store the results. Algorithm 6 converts the output of algorithm 5 into a PAC sample. An example of an PAC sample for a patch can be found in Appendix B.

Algorithm 6 PAC Sample Generator

```
Require: Integer field OCC[21] denoting the occurrences of each amino acid in
  the patch and the type of patch in position 21 (IP,SP or MP)
  Integer field SAM[101]
  for i<20 do
    $j \leftarrow 0$
    repeat
      if OCC[i] = j then
        $SAM[j + 5 * i] \leftarrow 1$
      end if
    until j = 5
  end for
  if OCC[21] = 1 then
    $SAM[101] \leftarrow 1$
  else
    $SAM[101] \leftarrow 0$
  end if
  Write SAM as string
```

6.5 Assessment of the PAC Hypotheses

We generated patches for the depth 0-5 and 25 to evaluate the impact of the depth on the classifier. We also generated five sets of SPs for each depth with the ratio of 2.7 to avoid overfitting and to ensure that we avoid an singular constellation to base our hypothesis on.
Table 6.3 shows some results from different depths; a detailed list of results for each depth can be found in Appendix C.
The column “Depth” shows the depth of the respective patch sample.
The column “Total IPs” shows the total amount of interface patches.
The column “True IPs” shows the amount of interface patches correctly identified by the classifier, which is the sensitivity of the discriminator.
The column “Total SPs” shows for the total amount of surface patches.
The column “True SPs” stands for the amount of surface patches correctly identified by the classifier, which is the specificity of the discriminator.
The column “Precision” shows the ratio between correctly identified interface patches and the total amount of predicted interface patches.
The column “Accuracy” shows the overall accuracy of the discriminator.

Depth	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0	76684	5202 (6.8%)	274230	246309 (89.8%)	15.7%	71.6%
1	27588	7141 (25.9%)	83997	68191 (81.2%)	31.1%	67.5
2	14925	3425 (23.1%)	42013	37231 (88.7%)	41.7%	71.4%
3	9691	1936 (20.0%)	27236	25191 (92.2%)	48.6%	73.5%
4	7270	1973 (27.2%)	20089	18717 (93.2%)	59.0%	75.6%
5	6021	1242 (20.7%)	17238	16588 (96.3%)	65.6%	76.7%
25	3832	1694 (44.2%)	11255	9918 (88.1%)	55.9%	77.0%

Table 6.3: Results of using the PAC hypothesis on different patch depths for $\mathfrak{PL}_{\mathfrak{hom}}$

For depth zero, we gain a hypothesis with a good specificity of 89.8% but a low sensitivity of 6.8%. The difference between the surface residues and the interface residues is not enough to truly separate the two distributions. And due to the ratio not being equal between IPs and SPs, we have slight overfitting here. Thus, the hypothesis tends to dismiss samples as surface patches, as the total amount of possible false negatives is lower than that of possible false positives.

Compared to depth zero and depth two, we have a loss of accuracy at depth one. This cannot be fully explained, but we assume that we lose interface residues that are singular extremities and thus are easily classified. And these patches, that incorporate just the immediate neighborhood, are not distinctive enough to provide a good discrimination. These two reasons lead to the loss of specificity. The precision is steadily improving until depth five.
All hypotheses share the unequal ratio between IPs and SPs, which gives the specificity a higher impact on the overall accuracy than the sensitivity. As these are the natural conditions, we have to put up with this slight case of overfitting.
The PAC-learner hypotheses yield a decent discriminator to separate interface patches from random surface patches. With increasing depth the patches have better discrimination. The downside of a higher depth is the smaller amount of interface patches compared to the original amount (depth zero).

6.5.1 Assessment on Independent Data Set

All the data and hypotheses we have gathered have been based on our learning data set $\mathfrak{PL}_{\mathfrak{hom}}$. Hypotheses are not tailored specifically to the set $\mathfrak{PL}_{\mathfrak{hom}}$, but work on protein data in general. We confirm this by using the hypotheses on an independent test data set $\mathfrak{PT}_{\mathfrak{hom}}$. Same as for the learning data, we need different IFFs for this set as well, which leads to a total of 553 files. The distribution of the different patch types of this $\mathfrak{PT}_{\mathfrak{hom}}$ set is shown in Table 6.4.

Depth	Interface Patches	Pure Surface Patches	Mixed Patches
0	17465	68309	0
1	5936	56163	23675
2	3151	50545	32078
3	2048	46258	37468
4	1613	42879	41282
5	1367	40210	44197
25	952	29095	55727

Table 6.4: Amount of the different patch types for each depth for $\mathfrak{PT}_{\mathfrak{hom}}$

The results shown in Table 6.5 are similar to those gained on the learning set $\mathfrak{PL}_{\mathfrak{hom}}$ (Table 6.3). This shows that our calculated hypotheses are independent of the data set.

Depth	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0	17465	1243 (7.1%)	68309	61654 (90.0%)	15.7%	73.3%
1	5936	1514 (25.5%)	16399	13352 (81.4.%)	33.2%	66.6%
2	3151	789 (25.0%)	8483	7479 (88.2%)	44.0%	71.1%
3	2048	922 (45.0%)	6104	5049 (82.7%)	46.6%	73.2%
4	1613	549 (34.0%)	4669	4234 (90.1%)	55.8%	76.1%
5	1367	354 (25.9%)	4040	3808 (94.3%)	60.4%	77.0%
25	952	479 (50.3%)	2622	2260 (86.2%)	57.0%	75.8%

Table 6.5: Results of using the PAC hypotheses on different patch depths for $\mathfrak{PT}_{\mathfrak{hom}}$

6.6 Refinement Measures

To further improve the results and gain the most out of our approach, we looked for additional measurements to improve our results. These measurements should be based on typical characteristics of patches to complement the hypothesis generated by the PAC learning algorithm.
We explored several directions to complement the hypotheses of the PAC learning. One major direction is using additional data to gain more characteristics of patches, that were distinctive for interface patches and random surface patches. We devised two refinement measures which are explained in detail in this section.
Also under consideration were restrictive measures, which meant limiting the data to certain important amino acids or residue groups, where the residue distribution is more distinctive. We dismissed these restrictive measures, due to the fact that our data is already limited and this would mean an even smaller database to work with. There are some restrictive measures mentioned in the concluding Chapter 8 for possible future avenues.

6.6.1 Conservation Score

The first refinement measure is based upon using additional data with the already existing IFFs. The MSAs were a likely candidate, they provide insight about how residues are conserved throughout the protein family. This can be expanded from a single residue to encompass a patch and observe how the patch is conserved throughout the MSA.

The first way to utilize MSAs is to improve the distributions of the amino acids. This can be done by counting all amino acids in the columns of a patch in the MSA and then to use this distribution as learning sample for the PAC learner. This is easy to use, but has one major flaw: it is likely to yield similar results as the original protein patch distribution. There is no real new information gain in this method, as the expanded distribution is very similar to the original distribution.

To get some new information and thus improve the results of the discrimination, we take a closer look at how well conserved the patches are throughout the MSA. The general consensus is that a significant residue in a chain will be either conserved or it will have a correlated mutation throughout the MSA. Thus, we developed a score to rate how conserved a patch is and use this score to further discriminate between interface patches and random surface patches. In order to be able to define a conservation score for a patch, we first need to define what a conservation score is for a single column in a MSA. As we do not know if the chain we are evaluating is actually the main species in the MSA or just another mutation, we determine the dominant acid of a column instead of using the residue on the original chain.

Definition 6.2
The dominant acid d in a MSA column C is the amino acid $d \in \mathfrak{A}$ with
$d = max\{\#a \in C | a \in \mathfrak{A}\}$.

Therefore, with this definition we can define what a conservation score for one column in the MSA is.

Definition 6.3
The conservation score $CV(C)$ of a single column C for a given MSA is calculated by $d/n, n$ being the length of C without counting any gaps and d being the dominant amino acid of that column.
If the column consist of more than 20% gaps it is marked as not conserved (score 0).

Definition 6.4
The conservation score CV of a patch P for a given MSA is calculated as

$$\sum_{C \in P} CV(C)/n \quad with \quad n = \#Columns\ in\ P.$$

The algorithm 7 is used for calculating the conservation score for a patch.

Algorithm 7 Patch Conservation Score Calculator

```
Require: Integer field Patch[n], which denotes the residues that are part of the
  patch with 1
  $conserv \leftarrow 0$
  $size \leftarrow 0$
  for i<n do
    if $Patch[i] = 1$ then
      size++
      Get column $C_i$ of the MSA
      $depth \leftarrow depthofMSA$
      initialize integer field $temp - amino[20]$ with 0
      $gaps \leftarrow 0$
      for $j < depth$ do
        if $C_i[j]$ NOT GAP then
          $a$ is the integer code for the amino acid at $C_i[j]$
          $temp - amino[a] + +$
        else
          $gaps + +$
        end if
      end for
      if $gaps/depth < 0.2$ then
        $d \leftarrow -100$
        for $j < 20$ do
          if $temp - amino[j] > d$ then
            $d \leftarrow temp - amino[j]$
          end if
        end for
        $conserv \leftarrow conserv + d/(depth - gaps)$
      else
        $conserv \leftarrow conserv + 0$
      end if
    end if
  end for
  $cv \leftarrow conserv/size$
  return $cv$
```

Table 6.6 shows the average conservation score values for the different patch depths. It shows that this conservation score is rather invariant for the different depths, but the difference between surface and interface patches is relatively small. This score can be used as a singular discrimination device for patches, which is shown in detail in Appendix C.

Depth	CV of IPs	CV of SPs	CV of MP's
0	0.327	0.282	-
1	0.341	0.278	0.311
2	0.340	0.275	0.311
3	0.332	0.272	0.311
4	0.329	0.271	0.311
5	0.33	0.16	0.3

Table 6.6: Average conservation score for the different patch types and different depths $\mathfrak{PL}_{\mathfrak{hom}}$

If we use the conservation score as a classifier, we need to set a cutoff value; everything lower than this cutoff is considered to be a random surface patch. Table 6.7 shows some exemplary results for different patch depths, with a cutoff 0.8 (the legend for the table is the same as for table 6.3). More detailed results are found in Appendix C. This table confirms the observation made on Table 6.6, that the conservation score is very stable throughout all depths of patches.

Depth	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0	76884	12813 (16.7%)	274230	245459 (89.5%)	30.8%	73.6%
1	27588	3629 (13.2%)	75782	72268 (95.4%)	50.8%	73.4%
2	14925	1845 (12.4%)	42016	40464 (96.3%)	54.3%	74.3%
3	9691	1096 (11.3%)	27250	26394 (96.9%)	56.1%	74.4%
4	7270	822 (11.3%)	19905	19350 (97.2%)	59.7%	74.2%
5	6021	692 (11.5%)	16980	16499 (97.2%)	59.0%	74.7%
25	3832	539 (14.1%)	11252	10904 (96.9%)	60.8%	75.9%

Table 6.7: Results of using the conservation score on different patch depths for $\mathfrak{PL}_{\mathfrak{hom}}$

6.6.2 Connectivity Score

The second refinement measure is based upon a new characteristic gained from already known data. This characteristic is embedded in the structural information of our patches. We already regard patches as graphs, hence, we can make use of certain characteristics of graphs to gain additional information.
In this case, we can determine how well the graph of a patch is connected. Based on the current results of research on proteins, it is safe to assume that a patch consisting only of interface residues is structurally more coherent than a random surface patch. We define this score using conventions of graph theory.

Definition 6.5
The connectivity score CT of a given patch P is defined as d/m, where d is the number of edges in P and m is the maximal number of edges possible in P.
If the graph contains only one node and no edges the score is set to 1.

To calculate the connectivity of a patch, we count all connections that the members of the patch have with each other. This amount is then divided through the maximum number of edges, which is $(n \cdot (n-1))/2$, if the patch has a size of n. We use algorithm 8 to calculate the connectivity score for a patch.

Algorithm 8 Patch Connectivity Score Calculator

```
Require: Integer field Patch[n], which denotes the residues that are part of the
  patch with 1
  $connect \leftarrow 0$
  $size \leftarrow 0$
  for i<n do
    if $Patch[i] = 1$ then
      size++
      Generate LN list of all neighbors from P[j]
      for all entries $m$ of LN do
        if $P[m] = 1$ AND $m > i$ then
          connect ++
        end if
      end for
    end if
  end for
  $ct \leftarrow connect/(size * (size - 1)/2)$
  return $ct$
```

We determine the average connectivity score for the different patch types and for different patch depths, which is shown in Table 6.8.

Depth	CT of IPs	CT of SPs	CT of MP's
0	-	-	-
1	0.802	0.798	0.757
2	0.562	0.520	0.456
3	0.509	0.414	0.33
4	0.518	0.365	0.264
5	0.543	0.341	0.224
25	0.7	0.21	0.12

Table 6.8: Average connectivity score for the different patch types and different depths for $\mathfrak{PL}_{\mathfrak{hom}}$

As can be expected, depth one surface and interface patches connectivity scores are fairly close to each other, as these small patches have a high connectivity. As the depth increases, it is obvious that the interface patches are better connected than the random surface patches. This score can be also used as a singular discrimination device for patches, which is shown in detail in Appendix C. As with the conservation score we need a cutoff value to determine which samples are positive and which are negative. Exemplary results for cutoff 0.9 are shown in Table 6.9.

Depth	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
1	27588	10263 (37.2%)	75444	48850 (64.8%)	27.8%	57.4%
3	14925	1793 (12.0%)	41718	38854 (93.1%)	38.5%	71.8%
3	9691	1790 (18.5%)	27128	25216 (93.0%)	48.4%	73.3%
4	7270	1790 (24.6%)	20088	18481 (92.0%)	52.7%	74.1%
5	6021	1790 (29.7%)	17126	15694 (91.6%)	55.6%	75.5%
25	3832	1790 (46.7%)	11252	9938 (88.3%)	57.7%	77.8%

Table 6.9: Results of using the connectivity score on different patch depths for $\mathfrak{PL}_{\mathfrak{hom}}$

This score improves with increasing patch depth. The number of interface patches with a high connectivity score is stable after depth three. In comparison, the number of random surface patches with high connectivity score steadily declines.

6.7 Assessment with Refinement

As we have defined and computed these refinement measures, we have several options on how to employ them. After reviewing the single discrimination results from the hypotheses and the refinement measures, we decided to use them as three equal scores. As can be seen throughout the result tables, our specificity is very good (above 90%) for all three different scores (hypothesis, CV, CT), but the sensitivity is lacking, which is our main concern and the improvement we want to focus upon.

With this focus in mind, we use the notion of a *dominant score*, which means that if this score classifies a sample as positive it is rated as positive for the whole variant. This leads to nine possible variant combinations.
Variants 1 to 3 are that one of the three scores is dominant. Thus, these variants classify samples as positive that are either rated positive by the dominant score or that are rated positive by both other scores. Similar to the first three, we define the variants 4 to 6 by denoting two scores as dominant, effectively eliminating the last score. Here only the positive samples rated by any of the dominant scores are classified as positive by the variant.
Variant 7 is that all scores are dominant, this would likely be the variant with the highest sensitivity, as a sample rated as positive by any score is classified as positive for the variant.
For variant 8 we use a majority system, in which at least two of the scores have to rate a sample as positive in order for it to be classified positive by the variant.
Variant 9 also uses a majority system, but here all three scores have to rate a sample as positive in order for it to be classified positive by the variant.

Due to the large number of results only the best variants are represented in Table 6.10. The complete result tables for each depth can be found in Appendix C. The first row is always the results of using only the hypothesis and the next two are the best variants in overall accuracy.

Variant (depth)	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
Hyp(1)	27588	7141 (25.9%)	75948	61726 (81.3%)	33.4%	66.5%
CV dom(1)	27588	6132 (22.2%)	75948	65457 (86.2%)	36.9%	69.1%
All agree(1)	27588	484 (1.8%)	75948	75404 (99.3%)	47.1	73.3%
Hyp(2)	14925	3435 (23.0%)	41638	36933 (88.7%)	42.2%	71.4%
CV dom(2)	14925	2368 (15.9%)	41638	39069 (93.8%)	48.0	73.3%
All agree	14925	177 (1.2%)	41638	41449 (95.3%)	48.4	73.6%
Hyp(3)	9691	3900 (40.2%)	27228	22448 (82.4%)	44.9%	71.4%
CT dom(3)	9691	2371 (24.5%)	27228	24977 (91.7%)	51.3%	74.1%
Majority(3)	9691	2265 (23.4%)	27228	25249 (92.7%)	53.4%	74.5%
Hyp(4)	7270	2360 (32.5%)	20043	18118 (90.0%)	55.1%	75.0%
Hyp dom(4)	7270	2512 (34.6%)	20043	18010 (90.0%)	55.3%	75.1%
Majority(4)	7270	1896 (26.1%)	20043	18631 (93.0%)	57.3%	75.2%
Hyp(5)	6021	1443 (24.0%)	17084	16183 (94.7%)	61.6%	76.3%
Hyp dom(5)	6021	1668 (27.7%)	17084	16006 (93.7%)	60.7%	76.5%
Majority(5)	6021	1453 (24.1%)	17084	16171 (94.7%)	61.4 %	76.3%
Hyp(25)	3832	1909 (49.8%)	11292	9758 (86.4%)	55.4%	77.1%
Hyp dom(25)	3832	2245 (58.6%)	11292	9500 (84.1%)	55.6%	77.7%
Majority(25)	3832	2134 (55.7%)	11292	9700 (85.9%)	57.3%	78.2%

Table 6.10: Results of different score variants for different patch depths for $\mathfrak{PL}_{\mathfrak{hom}}$

The worst results are produced by variant 7, where all scores are dominant. We do achieve the highest sensitivity of all variants (above 50%), but it is also the one with the worst specificity and the overall worst accuracy. It also has for higher depths a lower precision than the other variants.

Variant 1-6 have varying results, depending on the variant and the depth of the sample. For example, the CT score is not that reliable for patches of low depth. The variants, where the CT score is dominant or the CT score and another score are dominant, do not work well on lower patch depths but have good results on the higher patch depths. The CV score works almost exactly opposite, while having good results on low depth patches, it grows worse on high depth patches.
The variant where all scores have to agree, if a patch is considered a positive sample or not has quite a good overall accuracy. It has a very low sensitivity (usually under 10%) which can be useful depending on the algorithm that uses this discriminator. It also has a very high precision.
The best overall variant is the two-out-of-three majority vote method. It surpasses using only the PAC hypothesis in accuracy on almost all depths. It is also very often the variant with the best accuracy or the runner up.
As can be seen in all depths of patches, the overall accuracy improves through the combination of all three scores.
And as can be seen in Appendix C, Section 4, we get similar results on the test data set $\mathfrak{PT}_{\mathfrak{hom}}$.

6.7.1 Upholding the Ratio

We always maintained the ratio of 2.7 surface patches to 1 interface patch throughout our learning and testing.
If we use the above learned hypotheses and refinement measures on the full data set without removing any SPs on a specified depth, we achieve an overall accuracy of over 90%. The deciding factor is the absolute number of negative samples; as they are overrepresented in such a test, the sensitivity is almost of no consequence anymore. Only the specificity matters, which leads to the overall good accuracy; however, the sensitivity and specificity percentages stay almost the same.

6.7.2 Assessment under Field Conditions

We have used an optimized setting for learning our hypotheses and our CV and CT scores by only using pure interface and pure surface patches. We can also assess the most negative setting to use the patch classifier. If the classifier is used on a protein where the interface is unknown, we have no way to discern when building the patches if a patch is pure or mixed. Additionally, there will be no way to uphold our ratio of 1 interface patch to 2.7 surface patches, but we simply have to build all patches and classify them; these are the so-called field conditions.
Due to the fact that we generated our hypotheses upon pure interface patches, we consider all mixed patches as negative samples, even if they have an interface residue as seed.

Results for the Learning Data Set

Table 6.11 emulates a classification under field conditions for each depth, therefore, all patches that are generated are classified. The proteins were taken from the learn data set $\mathfrak{PL}_{\mathfrak{hom}}$ and the classification done with the majority refinement variant and with a 0.9 cutoff for the CV and CT scores.

Depth	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
1	27588	4519 (16.4%)	323326	271572 (84.0%)	8.0%	78.7%
2	14925	1382 (9.3%)	335989	322899 (96.1%)	9.5%	92.4%
3	9691	1748 (18.0%)	341223	325264 (95.3%)	9.9%	93.2%
4	7270	1448 (19.9%)	343644	329831 (96.0%)	9.5%	94.4%
5	6021	1168 (19.4%)	344893	334064 (96.9%)	9.7%	95.5%
25	3832	1489 (38.6%)	347082	333436 (96.1%)	9.8%	95.4%

Table 6.11: Results of the majority score variant for different patch depths for $\mathfrak{PL}_{\mathfrak{hom}}$

Results for the Test Data Set

Table 6.12 shows the results for test data set $\mathfrak{PT}_{\mathfrak{hom}}$ while using field conditions. The majority refinement variant was used with a 0.9 cutoff for the CV and CT scores.

Depth	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
1	5925	982 (16.6%)	79609	66239 (83.2%)	6.8%	78.6%
2	3148	229 (7.3%)	82386	79533 (96.5%)	7.4%	93.3%
3	2047	442 (21.6%)	83487	79126 (94.8%)	9.2%	93.0%
4	1612	385 (23.9%)	83922	80185 (95.5%)	9.3%	94.2%
5	1366	313 (22.9%)	84168	81093 (96.3%)	9.2%	95.2%
25	951	550 (40.7%)	84583	80707 (95.4%)	12.4%	94.8%

Table 6.12: Results for the majority variant for different patch depths for $\mathfrak{PT}_{\mathfrak{hom}}$

The precision is low compared to the results on optimized settings. This was expected, especially as we view mixed patches with an interface seed as negative. If combined with a good preselection that eliminates a part of the negative samples, the classifier would be more in line with the optimized results.

6.8 Comparison to Meta-PPISP

The patch classifier was not created to be an prediction tool, but it can be used as such and thus can be compared to other predictors.
We chose the *Meta-PPISP* meta web server as comparison predictor [QZ07]. It combines three different predictors through linear regression. Thus, it covers a few different approaches and it also predicts multichain interfaces.
We picked two example proteins to evaluate the performance of the patch classifier compared to the Meta-PPISP.

The first experiment in which we compared the classifiers is the analysis of the chain A of the crystal structure of the neutral form of fructose-1,6-bisphosphate complexed with the product fructose 6-phosphate (NFFP). Viewed as multichain interface and under our definitions of interface, this chain has 39 interface residues and 139 non-interface surface residues.
For our patch classifier we used a depth of four and cutoffs of 0.9 for both refinement measures. This depth has 3 interface patches which are all identified as such, additionally we have 2 non-interface patches identified as positives. The rest is correctly identified as non-interface. META-PPISP identifies 9 interface positions correctly and has 13 false positives.

Figure 6.3 shows the predicted residues. The correct identified patch interfaces are colored blue and the false positives are colored red. The correct predictions made by META-PPISP are green and the incorrect predictions are colored yellow.

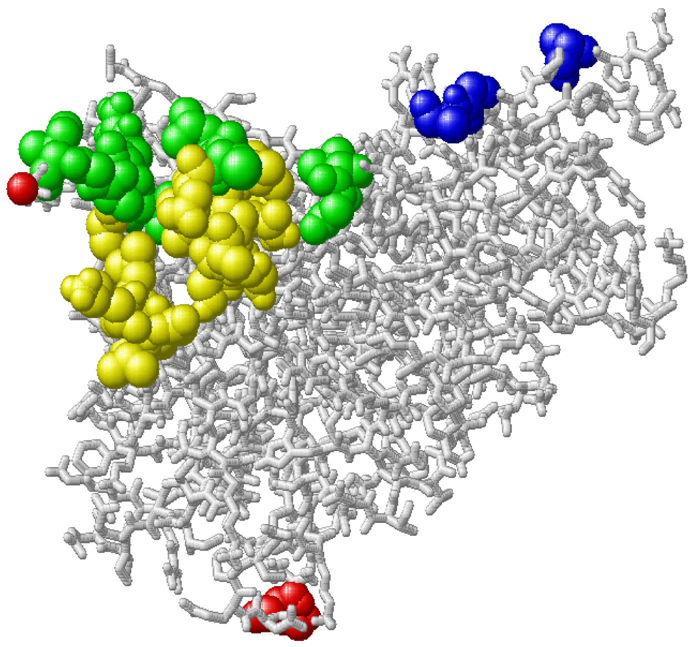


Figure 6.3: Predicted interface residues of chain A of the protein NFFP

It is discernable that META-PPISP identifies one large area as interface, although that area includes a lot non interface residues. Our patch classifier is more spread and makes predictions for smaller areas.

The second experiment for comparison analyses the chain B of the human beta-tryptase (HBT). This chain has 34 interface residues and 105 non-interface surface residues. We use the same conditions as above for our patch classifier. This results in having a total of 11 interface patches of which 10 are identified correctly. Additionally, 8 non-interface patches are incorrectly classified as interface patches. On this example, META-PPISP identifies 6 interface positions correctly and has 12 false positives.

We show the prediction of both classifiers for HBT in Figure 6.4. The same coloration is used as in Figure 6.3.

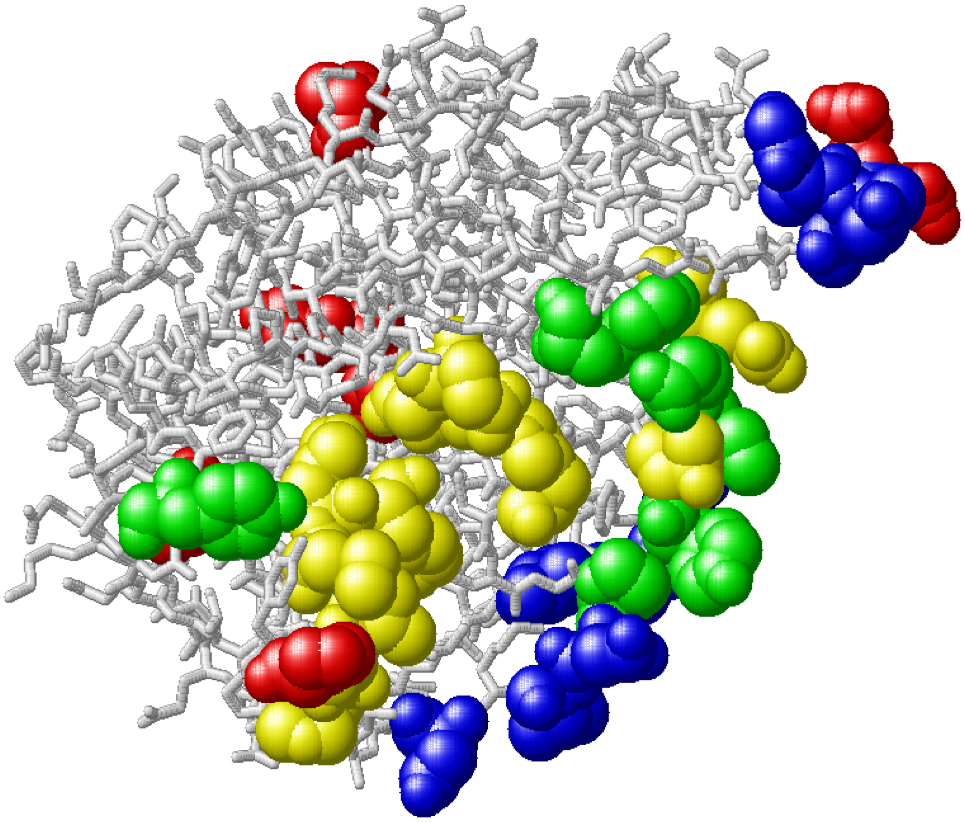


Figure 6.4: Predicted interface residues of chain B of the protein HBT

Again it is noticeable that META-PPISP makes its predictions in one large area and identifies the whole area has positive. Although we have no overlap here a few predicted residues are near to each other, indicating an area of interest, not necessarily an interface area. In comparison our patch classifier makes less correct predictions than the META-PPISP but also makes less mistakes. As we have seen in these examples, the patch classifier and META-PPISP have no overlap. This could indicate that a combined approach would yield even better results.

6.9 Conclusion

We have defined a classifier with a good specificity and low to average sensitivity, and a decent accuracy, with a change to a very good accuracy, if we omit the ratio between interface and surface patches.
The classifier gets better, the higher the depth of the patches is, which was expected. With the refinement measures and the different variants, the classifier can be adjusted to fit several needs and thus can be used in various settings.
Compared to other classifiers, our patch classifier gives similar results, as the sensitivity seems generally low for identifying interface residues [OR07b]. This can be explained by the fact that the method of determining interface residues by distance alone (Chapter 4.1.4) is prone to produce errors as only a small percentage of these interfaces are truly responsible for the binding of two proteins [BT98].
Our patch classifier has shown to yield comparable results to a current multi method predictor, which shows the effectiveness of our approach.
Possible future projects involving our patch classifier are given in Chapter 8.

Chapter 7

Comparison of SR Classifier and Patch Classifier

In this chapter we want to give a short comparison between the two classifiers. Although the significant residue (SR) classifier differs from the patch classifier, we try to compare the results, to show, how these classifiers complement and overlap each other.

The first example used for comparison is chain A of the crystal structure of the neutral form of fructose-1,6-bisphosphate complexed with the product fructose 6-phosphate (NFFP).
For our patch classifier we used a depth of four and cutoffs of 0.9 for both refinement measures. This depth has 3 interface patches which are all identified as such, additionally we have 2 non-interface patches identified as positive. The rest is correctly identified as non-interface. The SR classifier was used with refinement through the bistochastic matrix and all three NMI variants were calculated.

Figure 7.1 shows these results. For a better overview, we marked only the results of the patch classifier in combination with the SR classifier. The purple colored residues are two of the three correctly identified interface residues which are also significant for all three variants of the SR classifier. The last interface residue (blue) is not significant for any variant. The orange residue marks a false positive classified surface residue by the patch classifier. It is significant for all three variants of the SR classifier. The red residue is the second false positive by the patch classifier and it is not significant for the three SR variants.

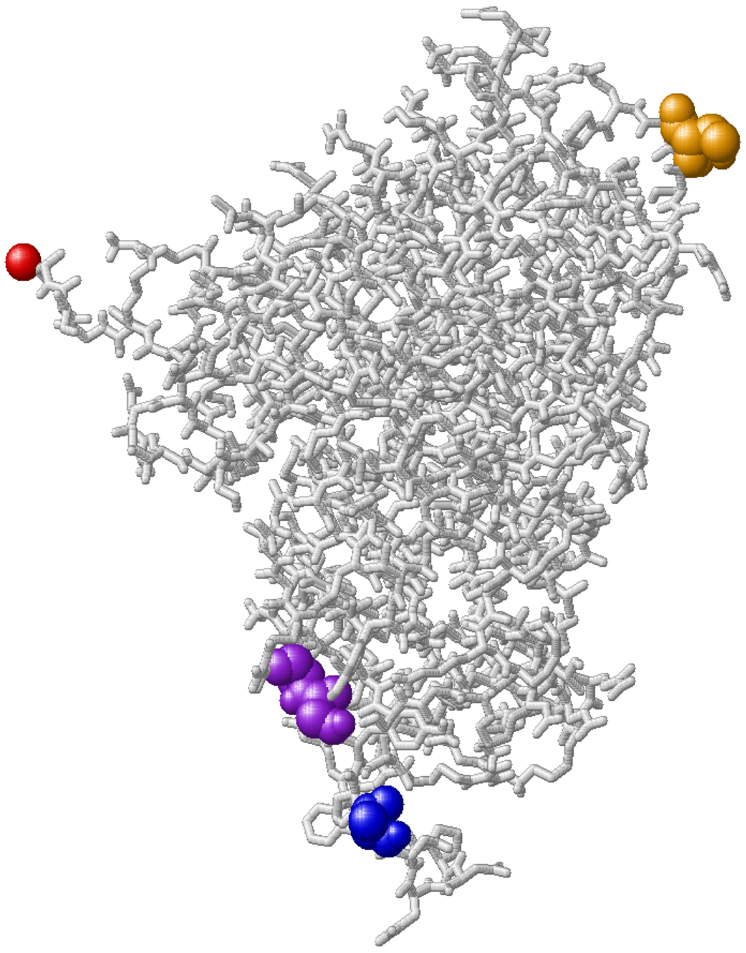


Figure 7.1: Overlap of the patch classifier and the SR classifier on chain A of the protein NFFP

The second protein chain we used to compare the two classifiers is chain B of the human beta-tryptase (HBT). This chain has 34 interface residues and 105 non-interface surface residues. We use the same conditions as above for our patch classifier. This results in having a total of 11 interface patches of which 10 are identified correctly. Additionally, 8 non-interface patches are incorrectly classified as interface patches. This comparison is visualized in Figure 7.2.

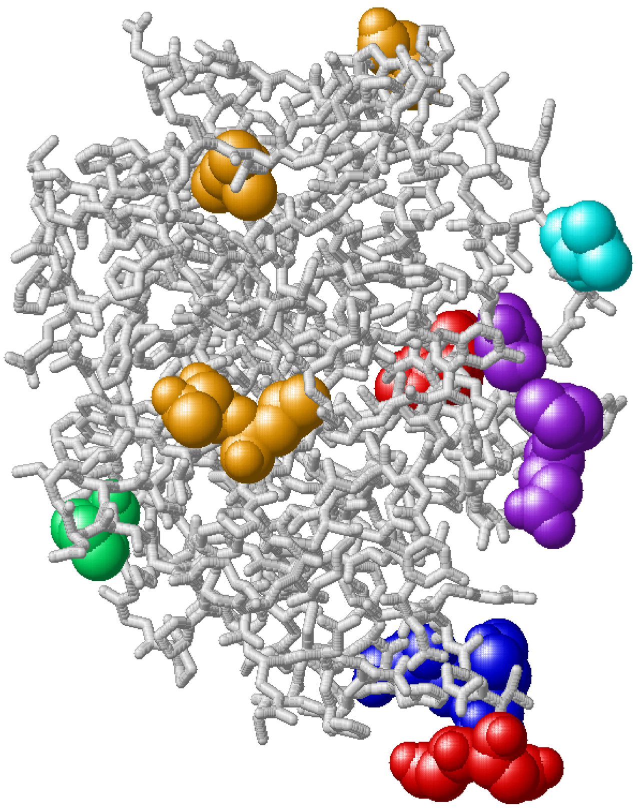


Figure 7.2: Overlap of the patch classifier and the SR classifier on chain B of the protein HBT

The three purple residues are interface residues correctly identified by the patch classifier and are also significant for all three SR variants. We have one interface residue that is only significant for the NMI_{min} variant and correctly classified by the patch, which is colored in cyan. The four orange residues are false positives of the patch classifier, but also significant for all NMI variants. As with the interface residues, we have one surface residue that is significant only for the NMI_{min} variant; it is colored in green. The blue (interface) and red (random surface) residues are the residues identified by the patch that are not significant for any NMI variant.
These examples show that the two classifiers interlock and can be used to further filter out the more important residues in a protein.

Chapter 8

Conclusion

We have shown in this work that the concepts of information theory and machine learning are well suited to promote the progress of creating new classifiers for evaluating protein residues. The combination of different concepts and methods seems to be better suited for developing a theoretical model for proteins than a single minded approach. With this concept we devised two new classifiers for detecting important residues on a protein chain.
The SR classifier is able to find important residues in proteins based on the respective MSA and is adaptable for different scenarios using different NMIs. It is also based on a plausible and stable mathematical model that simulates the relationship between correlation and coevolution.
The patch classifier has shown that the neighborhood of residues can be used to separate interface residues from random surface residues with a good accuracy. In comparison with a current multi method predictor, we achieved the same or better rate of accuracy on different examples.

8.1 Future Work

Both classifiers would benefit from an even larger database to learn from. Additionally, an extended analysis of different MSA heuristics could help to find even better suited MSAs for each classifier.
An interesting concept would be to combine the two classifiers. A logical choice would be to use the results of the SR classifier as input for the patch learning process. The patch classifier would then not be used for detecting interface residues but rather for detecting significant residues in a protein chain.

8.1.1 SR Classifier

The mathematical model for the SR classifier is stable and well defined but it is possible that some details can still be improved. For example, the null model for the not correlated residue pairs. We did not test all possible variants of creating a null model, so this avenue can be explored further. The calculation of the substitution matrix in Chapter 5.2 could be extended to encompass the use of phylogenetic trees. Although these trees are not 100% accurate, it would be an option to adjust the classifier to better fit a specified scenario, which is expressed through the phylogenetic trees. Another point of interest would be to determine which normalization variant is suited for which scenario, for example finding hotspots in a protein chain.

8.1.2 Patch Classifier

The patch classifier has some interesting options for future work possibilities. As mentioned in Chapter 6, we dismissed the option of restricting the patch classifier due to the database. With a larger or different database, these restrictions could be implemented, which would give the patch classifier a different focus. Instead of using the interface residues for the learning process, the input could be restricted to certified hotspots.
Another avenue would be to develop and use more refinement measures. For example, using the chemical properties of a patch constellation or the thermodynamical entropy of a patch. It is safe to assume that incorporating more protein characteristics would boost the accuracy of the classifier.
Another project, which is already being worked on, is incorporating the patch classifier into a conditional random field predictor. This predictor should be able to preselect the residues and give the patch classifier a better setting to work with.

Appendix A

An Exemplary IFF

An exemplary artificial interface information file; we use these files to store the relevant protein data for our projects (see Chapter 4).

```
>CHAIN1:A
IQAEEWYF
<SURFACE1:
SSSSSCCS
*NEIGHBORS1
1:2,3,4,
2:1,3,4,
3:1,2,4,
4:1,2,3,5,6,7,
5:4,6,7,
6:4,5,7,
7:4,5,6,8,
8:7,
>CHAIN2:B
IQAEEWYF
<SURFACE2:
SSSSSCCS
*NEIGHBORS2
1:2,3,4,
2:1,3,4,
3:1,2,4,
4:1,2,3,5,6,7,
5:4,6,7,
6:4,5,7,
7:4,5,6,8,
8:7,
#PAIRS:
5-GlU:8-PHE
1-ILE:7-TYR
```

Appendix B

Example of a Protein Patch PAC Sample

We use a surface patch P as example, which contains 8 residues and these are split in the following way:

- Cysteine: 3 residues,
- Glycine: 2 residues,
- Leucine: 1 residue,
- Tyrosine: 2 residues,
- all other amino acids have zero occurrences.

Through the algorithm described in Section 6.4.2, this patch configuration will be translated into an input sample for the PAC learner by converting each amino residue occurrence into 5 boolean values. As example, Cysteine, with its three occurrences in the patch, would be translated into: (0,0,0,1,0). The whole patch would be translated into the following string:

(1,0,0,0,0,0,0,0,1,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,
0,0,1,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,1,0,0,0,
1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,
1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,1,0,0,
0)

The first 100 numbers encode the amino residues occurrences in the patch and the last number signifies that it is a negative sample.

Appendix C

Results for Different Scores for Patches

The following sections shows results for the different scores and refinement measures developed for discriminating interface patches from surface patches (see Chapter 6).

As a reminder we give the legend for the result tables. In contrast to the table in Chapter 6, the first column stands for which sample is shown as each table has only samples of the same depth.
The column **"Total IPs"** stands for the total amount of interface patches.
The column **"True IPs"** stands for the amount of interface patches correctly identified by the individual score, which is the sensitivity of the discriminator.
The column **"Total SPs"** stands for the total amount of surface patches.
The column **"True SPs"** stands for the amount of surface patches correctly identified by the individual, which is the specificity of the discriminator.
The column **"Precision"** shows the ratio between correctly identified interface patches and the total amount of predicted interface patches.
The **"Accuracy"** column shows the overall accuracy of the discriminator.

C.1 PAC-Learner Hypotheses

These tables offer an overview of the results, which can be gained from using the hypotheses created by the PAC-learner on different samples of patches of varying depth. All these tests were done with the primary learning set $\mathfrak{L}_{\mathfrak{hom}}$.

C.1.1 Results for Depth 0

For depth 0 there is only one sample available, as this is already our complete learning data set, as a reminder, depth 0 are only the residues themselves.

Sample	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
1	76684	5202 (6.8%)	274230	246309 (89.8%)	15.7%	71.6%

C.1.2 Results for Depth 1

Sample	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
1	27588	9968 (36.2%)	84486	62422 (73.9%)	31.1%	64.6%
2	27588	9477 (34.4%)	84531	62862 (74.4%)	30.4%	64.5%
3	27588	9532 (34.6%)	84606	62936 (74.4%)	30.5%	64.5%
4	27588	7141 (25.9%)	83997	68191 (81.2%)	31.1%	67.5%
5	27588	9968 (36.2%)	84735	62656 (74.7%)	31.1%	64.7%

C.1.3 Results for Depth 2

Sample	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
1	14925	5378 (36.1%)	41882	33653 (80.4%)	39.5%	68.7%
2	14925	4635 (31.1%)	41745	35134 (84.2%)	41.2%	70.2%
3	14925	3425 (23.1%)	42013	37231 (88.7%)	41.7%	71.4%
4	14925	4641 (31.1%)	41945	35070 (84.7%)	40.3%	69.8%
5	14925	4069 (27.3%)	41997	36146 (86.1%)	41.0%	70.6%

C.1.4 Results for Depth 3

Sample	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
1	9691	2200 (22.8%)	27336	24974 (91.4%)	48.2%	73.4%
2	9691	1936 (20.0%)	27236	25191 (92.2%)	48.6%	73.5%
3	9691	2375 (24.6%)	27377	24803 (90.6%)	49.0%	73.3%
4	9691	3904 (40.3%)	27427	22618 (82.5%)	44.8%	71.5%
5	9691	3528 (36.5%)	27384	23356 (85.3%)	46.7%	72.5%

C.1.5 Results for Depth 4

Sample	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
1	7270	2621 (36.1%)	20108	17788 (88.5%)	53.0%	74.5%
2	7270	1937 (26.7%)	20181	18568 (92.1%)	54.6%	74.7%
3	7270	2314 (31.9%)	20135	18376 (91.3%)	56.8%	75.5%
4	7270	1671 (23.0%)	19942	18786 (94.3%)	59.1%	75.2%
5	7270	1973 (27.2%)	20089	18717 (93.2%)	59.0%	75.6%

C.1.6 Results for Depth 5

Sample	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
1	6021	1474 (24.5%)	17072	16150 (94.6%)	61.5%	76.3%
2	6021	1356 (22.6%)	16959	15972 (94.2%)	57.9%	75.4%
3	6021	1346 (22.4%)	17368	16566 (95.4%)	62.7%	76.5%
4	6021	1140 (18.0%)	17107	16465 (96.3%)	64.0%	76.1%
5	6021	1242 (20.7%)	17238	16588 (96.3%)	65.6%	76.7%

C.1.7 Results for Depth 25

Sample	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
1	3832	1694 (44.2%)	11255	9918 (88.1%)	55.9%	77.0%
2	3832	1999 (52.2%)	11252	9555 (84.9%)	54.1%	76.6%
3	3832	1694 (44.2%)	11043	9712 (88.1%)	56.0%	76.7%
4	3832	850 (22.2%)	11280	10776 (95.5%)	62.8%	76.9%
5	3832	535 (14.0%)	11341	11089 (97.8%)	68%	76.6%

C.1.8 Hypothesis on Test Data

These samples were taken from the test data set $\mathfrak{T}_{\mathfrak{hom}}$ and generated with the correspondent hypothesis for each depth. As we used these results to test the independence of our classifier from the learning data set, one sample per depth is sufficient.

Sample	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0	17407	1234 (7.1%)	69309	62587 (90.3%)	15.5%	73.3%
1	5925	1514 (25.5%)	16252	14149 (80.9%)	41.9%	66.1%
2	3148	789 (25.0%)	8733	7759 (88.8%)	44.8%	71.9%
3	2047	922 (45.0%)	5612	4586 (81.7%)	47.3%	71.9%
4	1612	549 (34.0%)	4716	4261 (90.4%)	54.7%	76.0%
5	1366	354 (25.9%)	4065	3872 (95.3%)	64.7%	77.8%
25	951	479 (50.3%)	2539	2182 (85.9%)	57.3%	76.2%

C.2 Refinement Measure Conservation Score

These tables show the discriminating properties of the refinement measure conservation score (Chapter 6.6), when only this score is used to differentiate between patches. For each depth of patches one sample was chosen and different cutoffs for the conservation score (CV) were used.
Column one stands here for the different cutoffs being used on the same sample.

C.2.1 Results for Depth 0

Cutoff	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0.5	76884	24948 (32.4%)	274230	204835 (74.7%)	26.4%	65.5%
0.6	76884	20158 (26.2%)	274230	222032 (81.0%)	27.9%	69.0%
0.7	76884	16298 (21.2%)	274230	234860 (85.6%)	29.3%	71.5%
0.8	76884	12813 (16.7%)	274230	245459 (89.5%)	30.8%	73.6%
0.9	76884	9003 (11.7%)	274230	255710 (93.2%)	32.7%	75.4%

C.2.2 Results for Depth 1

Cutoff	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0.5	27588	10532 (38.2%)	75782	56517 (74.6%)	34.9%	64.9%
0.6	27588	8260 (29.9%)	75782	63600 (83.9%)	40.4%	69.5%
0.7	27588	5874 (21.3%)	75782	68861 (90.9%)	45.9%	72.3%
0.8	27588	3629 (13.2%)	75782	72268 (95.4%)	50.8%	73.4%
0.9	27588	1787 (6.5%)	75782	74272 (98.0%)	54.2%	73.6%

C.2.3 Results for Depth 2

Cutoff	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0.5	14925	5827 (39.0%)	42016	31684 (75.4%)	36.1%	65.9%
0.6	14925	4631 (31.0%)	42016	35891 (85.4%)	43.1%	71.2%
0.7	14925	3107 (20.8%)	42016	38840 (92.4%)	49.5%	73.7%
0.8	14925	1845 (12.4%)	42016	40464 (96.3%)	54.3%	74.3%
0.9	14925	771 (5.2%)	42016	41419 (98.6%)	56.4%	74.1%

C.2.4 Results for Depth 3

Cutoff	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0.5	9691	3715 (38.3%)	27250	20802 (76.3%)	36.6%	66.4%
0.6	9691	2890 (29.8%)	27250	23639 (86.7%)	44.5%	71.8%
0.7	9691	1957 (20.2%)	27250	25415 (93.3%)	51.6%	74.1%
0.8	9691	1096 (11.3%)	27250	26394 (96.9%)	56.1%	74.4%
0.9	9691	448 (4.6%)	27250	26903 (98.7%)	56.4%	74.0%

C.2.5 Results for Depth 4

Cutoff	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0.5	7270	2725 (37.5%)	19905	15344 (77.1%)	37.4%	66.5%
0.6	7270	2107 (29.0%)	19905	17460 (87.7%)	46.3%	72.0%
0.7	7270	1440 (19.8%)	19905	18704 (94.0%)	54.5%	74.1%
0.8	7270	822 (11.3%)	19905	19350 (97.2%)	59.7%	74.2%
0.9	7270	359 (4.9%)	19905	19694 (98.9%)	63.0%	73.8%

C.2.6 Results for Depth 5

Cutoff	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0.5	6021	2224 (36.9%)	16980	13178 (77.6%)	36.9%	67.0%
0.6	6021	1734 (28.8%)	16980	14955 (88.1%)	46.1%	72.6%
0.7	6021	1196 (19.7%)	16980	15978 (94.1%)	54.4%	74.7%
0.8	6021	692 (11.5%)	16980	16499 (97.2%)	59.0%	74.7%
0.9	6021	312 (1.8%)	16980	16802 (99.0%)	63.7%	74.4%

C.2.7 Results for Depth 25

Cutoff	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0.5	3832	1363 (35.6%)	11252	8830 (78.5%)	36.0%	67.6%
0.6	3832	1073 (28.0%)	11252	9951 (88.4%)	45.2%	73.1%
0.7	3832	779 (20.3%)	11252	10575 (94.0%)	53.5%	75.3%
0.8	3832	539 (14.1%)	11252	10904 (96.9%)	60.8%	75.9%
0.9	3832	275 (7.2%)	11252	11093 (98.6%)	63.4%	75.4%

C.2.8 CV on Test Data

These samples were taken from the test data set $\mathfrak{T}_{\mathfrak{hom}}$ and generated with a CV cutoff value of 0.8.

Depth	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0	17407	3251 (18.5%)	68309	60535 (88.6%)	29.5%	74.4%
1	5925	868 (14.6%)	16399	15497 (94.5%)	49.0%	73.3%
2	3148	437 (13.9%)	8483	8131 (95.9.1%)	55.4%	73.6%
3	2047	266 (13.0%)	6104	5898 (96.6%)	56.4%	39.4%
4	1612	203 (12.6%)	4669	4505 (96.5%)	55.3%	39.4%
5	1366	171 (12.5%)	4040	3890 (96.3%)	53.3%	39.4%
25	951	122 (12.8%)	2622	2533 (96.6%)	57.8%	39.4%

C.3 Refinement Measure Connectivity Score

These tables show the discriminating properties of the refinement measure connectivity score (Chapter 6.6), when only the CT is used to decide which patches are interface patches and which are surface patches. For each depth of patches one sample is given and different cutoffs for the connectivity score (CT) are used.
Column one stands here for the different cutoffs being used on the same sample.

C.3.1 Results for Depth 0

As the patches consist of only one residue(the seed itself) the CT is set to 1 by definition and thus no discrimination can be done as all patches have the same CT.

C.3.2 Results for Depth 1

Cutoff	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0.5	27588	27556 (99.9%)	75444	305 (0.4%)	26.8%	27.0%
0.6	27588	26835 (97.3%)	75444	3660 (4.9%)	27.2%	29.6%
0.7	27588	15277 (55.4%)	75444	30710 (40.7%)	25.6%	44.6%
0.8	27588	13206 (47.9%)	75444	37651 (49.9%)	25.9%	49.4%
0.9	27588	10263 (37.2%)	75444	48850 (64.8%)	27.8%	57.4%

C.3.3 Results for Depth 2

Cutoff	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0.5	14925	8180 (54.8%)	41718	21815 (52.3%)	29.1%	53.0%
0.6	14925	5764 (38.6%)	41718	28973 (69.4%)	31.1%	61.3%
0.7	14925	2351 (15.8%)	41718	36608 (87.8%)	31.5%	68.8%
0.8	14925	2031 (13.6%)	41718	38098 (91.3%)	35.9%	70.8%
0.9	14925	1793 (12.0%)	41718	38854 (93.1%)	38.5%	71.8%

C.3.4 Results for Depth 3

Cutoff	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0.5	9691	4169 (43.0%)	27128	20454 (75.4%)	38.4%	66.9%
0.6	9691	2565 (26.5%)	27128	23474 (86.5%)	41.2%	70.7%
0.7	9691	1893 (19.5%)	27128	24852 (91.6%)	45.4%	72.6%
0.8	9691	1850 (19.1%)	27128	25032 (92.3%)	46.9%	73.0%
0.9	9691	1790 (18.5%)	27128	25216 (93.0%)	48.4%	73.3%

C.3.5 Results for Depth 4

Cutoff	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0.5	7270	2914 (40.1%)	20088	16571 (82.5%)	45.3%	71.2%
0.6	7270	2460 (33.8%)	20088	17315 (86.2%)	47.0%	72.3%
0.7	7270	1893 (26.0%)	20088	18170 (90.5%)	49.7%	73.3%
0.8	7270	1850 (25.4%)	20088	18238 (90.8%)	50.0%	73.4%
0.9	7270	1790 (24.6%)	20088	18481 (92.0%)	52.7%	74.1%

C.3.6 Results for Depth 5

Cutoff	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0.5	6021	2909 (48.3%)	17126	14018 (81.9%)	48.3%	73.1%
0.6	6021	2460 (40.9%)	17126	14648 (85.5%)	50.2%	73.9%
0.7	6021	1893 (31.4%)	17126	15443 (90.2%)	52.9%	74.9%
0.8	6021	1850 (30.7%)	17126	15562 (90.9%)	54.2%	75.2%
0.9	6021	1790 (29.7%)	17126	15694 (91.6%)	55.6%	75.5%

C.3.7 Results for Depth 25

Cutoff	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
0.5	3832	2909 (75.9%)	11252	8437 (75.0%)	50.8%	75.2%
0.6	3832	2460 (64.2%)	11252	9000 (80.0%)	52.2%	76.0%
0.7	3832	1893 (49.4%)	11252	9684 (86.1%)	54.7%	76.8%
0.8	3832	1850 (48.3%)	11252	9817 (87.2%)	56.3%	77.3%
0.9	3832	1790 (46.7%)	11252	9938 (88.3%)	57.7%	77.8%

C.3.8 CT on Test Data

These samples were taken from the test data set $\mathfrak{T}_{\mathfrak{hom}}$ and generated with a CT cutoff value of 0.9.

Depth	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
1	5925	2338 (39.4%)	16399	10570 (64.5%)	28.6%	57.8%
2	3148	845 (26.8%)	8483	7572 (89.3%)	48.1%	72.3%
3	2047	468 (22.9%)	6104	5650 (92.6%)	50.8%	75.0%
4	1612	468 (29.0%)	4669	4287 (91.8%)	55.1%	75.7%
5	1366	468 (34.2%)	4040	3663 (90.7%)	55.4%	76.4%
25	951	468 (49.2%)	2622	2305 (87.9%)	59.6%	77.6%

C.4 Hypotheses with refinement

This section shows the result of using different variants for combining the hypothesis with the two refinement measures (CV and CT) for the different patch depths. For these tables each row depicts another method of choosing the refinements, so the first column stands for the refinement variant.
Hyp: Samples are evaluated with the Hypothesis gained from the PAC-learner.
Hyp dom: Hypothesis is dominant, all samples evaluated by the hypothesis as positive are marked positive. If the hypothesis evaluates a sample as negative and both the CV_CT score evaluate it as positive it is marked positive.
CV dom: CV score is dominant. Similar to the Hypothesis is dominant variant above.
CT dom: CT score is dominant. Similar to the Hypothesis is dominant variant above.
Hyp_CV: Hypothesis and CV score are dominant, if any sample is evaluated as positive by any of these two it is marked as positive. The last score has no influence in this variant.
Hyp_CT: Hypothesis and CT score are dominant, if any sample is evaluated as positive by any of these two it is marked as positive. The last score has no influence in this variant.
CV_CT: CV_CT score are dominant, if any sample is evaluated as positive by any of these two it is marked as positive. The last score has no influence in this variant.
All dom: All scores are dominant, so if any score evaluates a sample as positive it is marked as positive.
Majority: Two of the three scores have to evaluate a sample as positive before it is marked as positive.
All agree: All scores have to evaluate a sample as positive for the sample to be marked as positive.

C.4.1 Results for Depth 0

The CV cutoff is 0.8, CT score is of no consequence for Depth 0.

Variant	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
Pure Hyp	76684	5202 (6.8%)	274230	246309 (89.8%)	15.7%	71.6%
CV dom	76884	12813 (16.7%)	274230	245459 (89.5%)	30.8%	73.6%
All dom	76884	17258 (22.4%)	274230	219694 (80.1%)	24.0%	67.5%
All agree	76684	757 (1.0%)	274230	272074 (99.2%)	26.0%	77.4%

C.4.2 Results for Depth 1

The CV cutoff is 0.8, CT cutoff is 0.9.

Variant	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
Hyp	27588	7141 (25.9%)	75948	61726 (81.3%)	33.4%	66.5%
Hyp dom	27588	7750 (28.1%)	75948	61043 (80.4%)	34.2%	66.4%
CV dom	27588	6132 (22.2%)	75948	65457 (86.2%)	36.8%	69.1%
CT dom	27588	10554 (38.3%)	75948	48617 (64.0%)	36.8%	57.2%
Hyp_CV	27588	9076 (32.9%)	75948	61471 (78.9%)	38.5%	66.7%
Hyp_CT	27588	13498 (48.9%)	75948	43112 (56.8%)	29.1%	54.7%
CV_CT	27588	11880 (43.1%)	75948	47544 (62.6%)	29.8%	57.4%
ALL dom	27588	14824 (53.7%)	75948	42021 (55.3%)	30.4%	54.9%
Majority	27588	4806 (17.4%)	75948	66548 (87.6%)	33.7%	68.9%
All agree	27588	484 (1.8%)	75948	75404 (99.3%)	47.1%	73.3%

C.4.3 Results for Depth 2

The CV cutoff is 0.8, CT cutoff is 0.9.

Variant	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
Hyp	14925	3435 (23.0%)	41638	36933 (88.7%)	42.2%	71.4%
Hyp dom	14925	3484 (23.3%)	41638	36880 (88.6%)	42.3%	71.4%
CV dom	14925	2368 (15.9%)	41638	39069 (93.8%)	48.0%	73.3%
CT dom	14925	1999 (13.4%)	41638	38737(93.0%)	40.8%	72.0%
Hyp_CV	14925	4380 (29.3%)	41638	36185 (86.1%)	44.5%	71.7%
Hyp_CT	14925	4011 (26.9%)	41638	35853 (56.8%)	40.9%	70.5%
CV_CT	14925	2895 (19.4%)	41638	38042 (91.4%)	44.6%	72.4%
ALL dom	14925	4907 (32.9%)	41638	35158 (84.4%)	43.1%	70.8%
Majority	14925	1472 (10.3%)	41638	39674 (95.5%)	42.8%	72.9%
All agree	14925	177 (1.2%)	41638	41449 (95.3%)	48.4%	73.6%

C.4.4 Results for Depth 3

The CV cutoff is 0.65, CT cutoff is 0.9.

Variant	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
Hyp	9691	3900 (40.2%)	27228	22448 (82.4%)	44.9%	71.4%
Hyp dom	9691	3943 (40.7%)	27228	22428 (82.4%)	45.1%	71.4%
CV dom	9691	3653 (37.7%)	27228	23348 (85.7%)	51.3%	73.1%
CT dom	9691	2371 (24.5%)	27228	24977 (91.7%)	51.3%	74.1%
Hyp_CV	9691	5331 (55.0%)	27228	20527 (75.4%)	44.3%	70.0%
Hyp_CT	9691	4049 (41.8%)	27228	22156 (81.4%)	44.4%	71.0%
CV_CT	9691	3759 (38.8%)	27228	23076 (84.8%)	47.5%	72.7%
ALL dom	9691	5437 (56.1%)	27228	20255 (74.4%)	43.8%	69.6%
Majority	9691	2265 (23.4%)	27228	25249 (92.7%)	53.4%	74.5%
All agree	9691	391 (4.0%)	27228	26906 (98.8%)	54.8%	73.9%

C.4.5 Results for Depth 4

The CV cutoff is 0.55, CT cutoff is 0.9.

Variant	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
Hyp	7270	2360 (32.5%)	20043	18118 (90.0%)	55.1%	75.0%
Hyp dom	7270	2512 (34.6%)	20043	18010 (90.0%)	55.3%	75.1%
CV dom	7270	3415 (47.0%)	20043	15738 (78.5%)	44.2%	70.1%
CT dom	7270	2149 (29.6%)	20043	18281 (91.2%)	54.9%	74.8%
Hyp_CV	7270	4031 (55.4%)	20043	15117 (75.4%)	45.0%	70.1%
Hyp_CT	7270	2765 (38.0%)	20043	17660 (88.1%)	53.7%	74.8%
CV_CT	7270	3668 (50.5%)	20043	15388 (76.8%)	44.1%	69.8%
ALL dom	7270	4284 (58.9%)	20043	14767 (73.7%)	44.8%	69.8%
Majority	7270	1896 (26.1%)	20043	18631 (93.0%)	57.3%	75.2%
All agree	7270	412 (5.7%)	20043	19753 (98.6%)	58.7%	73.8%

C.4.6 Results for Depth 5

The CV cutoff is 0.55, CT cutoff is 0.85.

Variant	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
Hyp	6021	1443 (24.0%)	17084	16183 (94.7%)	61.6%	76.3%
Hyp dom	6021	1668 (27.7%)	17084	16006 (93.7%)	60.7%	76.5%
CV dom	6021	2775 (46.1%)	17084	13678 (80.1%)	44.9%	71.2%
CT dom	6021	1906 (31.7%)	17084	15651 (91.6%)	57.1%	76.0%
Hyp_CV	6021	2990 (49.7%)	17084	13513 (79.1%)	45.6%	71.4%
Hyp_CT	6021	2121 (35.2%)	17084	15486 (90.6%)	57.0%	76.2%
CV_CT	6021	3228 (53.6%)	17084	13158 (77.2%)	45.1%	70.9%
ALL dom	6021	3443 (57.2%)	17084	12993 (76.1%)	45.7%	71.1%
Majority	6021	1453 (24.1%)	17084	16171 (94.7%)	61.4%	76.3%
All agree	6021	339 (5.6%)	17084	16952 (99.2%)	72.0%	74.8%

C.4.7 Results for Depth 25

The CV cutoff is 0.5, CT cutoff is 0.6.

Variant	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
Hyp	3832	1909 (49.8%)	11292	9758 (86.4%)	55.4%	77.1%
Hyp dom	3832	2245 (58.6%)	11292	9500 (84.1%)	55.6%	77.7%
CV dom	3832	2486 (64.9%)	11292	7989 (70.7%)	42.9%	69.3%
CT dom	3832	2556 (66.7%)	11292	8995 (79.7%)	52.7%	76.4%
Hyp_CV	3832	2597 (67.8%)	11292	7789 (69.0%)	42.6%	68.7%
Hyp_CT	3832	2667 (69.6%)	11292	8795 (77.9%)	51.6%	75.8%
CV_CT	3832	2908 (75.9%)	11292	7284 (64.5%)	42.0%	67.4%
ALL dom	3832	3019 (78.8%)	11292	7084 (62.7%)	41.8%	66.8%
Majority	3832	2134 (55.7%)	11292	9700 (85.9%)	57.3%	78.2%
All agree	3832	579 (15.0%)	11292	10961 (94.7%)	63.6%	76.3%

C.4.8 Majority Variant on Test Data

These samples were taken from the test data set $\mathfrak{T}_{\mathfrak{hom}}$ and generated with the majority refinement variant and the cutoffs used in the tables before.

Depth	Total IPs	True IPs (Sensitivity)	Total SPs	True SPs (Specificity)	Precision	Accuracy
1	5925	1143 (19.3%)	16275	14168 (87.1%)	35.2%	68.9%
2	3148	398 (12.6%)	8637	8466 (98.0%)	69.9%	75.2%
3	2047	554 (27.1%)	6057	5568 (91.9%)	53.1%	75.6%
4	1612	476 (29.5%)	4786	4454 (93.1%)	58.9%	77.0%
5	1366	464 (33.9%)	3902	3621 (92.8%)	62.3%	77.5%
25	951	550 (57.8%)	2628	2249 (85.6%)	53.8%	78.2%

Bibliography

[AC98] **Adams**, Jerry M. and Suzanne **Cory**: *The Bcl-2 Protein Family: Arbiters of Cell Survival*, *Science*, 281, 1998.

[AD74] **Ahrens**, J H and U **Dieter**: *Computer methods for sampling from gamma, beta, poisson and bionomial distributions*, *Computing*, 12(3):pages 223–246, 1974.
URL `http://www.springerlink.com/index/10.1007/BF02293108`

[AL90] **Altschul**, Stephen F. , Warren Gish , Webb Miller , Eugene W. Myers and David J. **Lipman**: *Basic local alignment search tool*, *Journal of Molecular Biology*, 1990.

[BA05] **Bordner**, A.J. and R. **Abagyan**: *Statistical analysis and prediction of protein-protein interfaces*, *Proteins*, 60, 2005.

[BB00] **Berman**, Helen M. , John Westbrook , Zukang Feng , Gary Gilliland , T. N. Bhat , Helge Weissig , Ilya N. Shindyalov and Philip E. **Bourne**: *The Protein Data Bank*, *Nucleic Acids research*, 28:pages 235–242, 2000.

[BJ06] **Burgoyne**, N.J. and R.M. **Jackson**: *Predicting protein interaction sites: binding hot-spots in protein-protein and protein-ligand interfaces*, *Bioinformatics*, 22, 2006.

[Bro08] **Brodag**, Thomas: *PAC-Lernen zur Insolvenzerkennung und Hotspot-Identifikation*, phdthesis, Universitaet Goettingen, 2008.

[BSB$^+$10] **Bremm**, Sebastian, Tobias **Schreck**, Patrick **Boba**, Stephanie **Held** and Kay **Hamacher**: *Computing and visually analyzing mutual information in molecular co-evolution*, *BMC Bioinformatics*, 11(1):page 330, 2010.
URL `http://www.biomedcentral.com/1471-2105/11/330`

[BT98] **Bogan**, Andrew A and Kurt S **Thorn**: *Anatomy of Hot Spots in Protein Interfaces*, *J.Mol.Bio*, 280, 1998.

[BW05] **Bradford**, J.R. and D.R. **Westhead**: *Improved prediction of protein-protein binding sites using a support vector machines approach*, *Bioinformatics*, 21, 2005.

[BW06] **Bradford**, James R. , Chris J. Needham , Andrew J. Bulpitt and David R. **Westhead**: *Insights into protein-protein interfaces using a Bayesian network prediction method*, *Journal of molecular biology*, 362, 2006.

[BW10] **Brodag**, Thomas , Steffen Herbold and Stephan **Waack**: *A Generalized Model of PAC Learning and its Applicability*, 2010, modified PAC algorithm.

[CB06] **Chung**, Jo-Lan , Wei Wang and Philip E. **Bourne**: *Exploiting sequence and structure homologs to identify protein-protein binding sites*, *Proteins*, 62, 2006.

[CT91] **Cover**, Thomas M. and Joy A. **Thomas**: *Elements of Information Theory*, WILEY, 1991.

[CZ05] **Chen**, H. and H.-X. **Zhou**: *Prediction of interface residues in protein-protein complexes by a consensus neural network: test against NMR data*, *Proteins*, 61, 2005.

[DH04] **Dehling**, Herold and Beate **Haupt**: *Einfuehrung in die Wahrscheinlichkeitstheorie und Statistik*, Springer, 2004.

[dVB06] **de Vries**, Sjoerd J. , Aalt D.J. van Dijk and Alexandre M.J.J. **Bonvin**: *WHISCY: what information does surface conservation yield? Application to data driven docking*, *Proteins*, 63, 2006.

[DW05] **Dunn**, L. C. Martin , G. B. Gloor , S. D. and L. M. **Wahl**: *Using information theory to search for co-evolving residues in proteins*, *Bioinformatics/computer Applications in The Biosciences*, 21:pages 4116–4124, 2005.

[EP07] **Eyal**, Eran , Milana Frenkel-Morgenstern , Vladimir Sobolev and Shmuel **Pietrokovski**: *A Pair-to-Pair Amino Acids Substitution Matrix and its Applications for Protein Structure Prediction*, *PROTEINS: Structure, Function, and Bioinformatics*, 67:pages 142–153, 2007.

[ES95] **Eisenhaber**, Frank , Philip Lijnzaad , Patrick Argos , Chris Sander and Michael **Scharf**: *The Double Cubic Lattice Method: Efficient Approaches to Numerical Integration of Surface Area and Volume and to Dot Surface Contouring of Molecular Assemblies*, *Journal of Computational Chemistry*, 16:pages 273–284, 1995.

[FC02] **Fariselli**, Piero , Florencio Pazos , Alfonso Valencia and Rita **Casadio**: *Prediction of protein-protein interaction sites in heterocomplexes with neural networks.*, *European Journal of Biochemistry*, 269, 2002.

[FM06] **Friedrich**, Torben , Birgit Pils , Thomas Dandekar , Jörg Schultz and Tobias **Müller**: *Modelling interaction sites in protein domains with interaction profile hidden Markov models*, *Bioinformatics*, 22, 2006.

[GC00] **Goh**, Chern-Sing , Andrew A. Bogan , Marcin Joachimiak , Dirk Walther , and Fred E. **Cohen**: *Co-evolution of proteins with their interaction partners*, *Journal of Molecular Biology*, 299, 2000.

[Hau88] **Haussler**, D: *Quantifying inductive bias: AI learning algorithms and Valiant's learning framework*, *Artificial Intelligence*, 36, 1988.

[HB06] **Hoskins**, Jemima , Simon Lovell and Tom L. **Blundell**: *An algorithm for predicting protein-protein interaction sites: abnormally exposed amino acid residues and secondary structure elements*, *Protein Science*, 15, 2006.

[HH92] **Henikoff**, Steven and Jorja G. **Henikoff**: *Amino acid substitution matrices from protein blocks*, *Proceedings of The National Academy of Sciences*, 89:pages 10915–10919, 1992.

[HHH94] **Henikoff**, Steven, Jorja G. **Henikoff** and Howard **Hughes**: *Position-based sequence weights*, *J. Mol. Biol*, 243:pages 574–578, 1994.

[Hil95] **Hillis**, David M.: *Approaches for Assessing Phylogenetic Accuracy*, *Systematic Biology*, 44, 1995.

[HK10] **Hildebrandt**, Andreas ,Anna Katharina Dehof , Alexander Rurainski , Andreas Bertsch , Marcel Schumann , Nora Toussaint , Andreas Moll , Daniel Stockel , Stefan Nickels , Sabine Mueller , Hans-Peter Lenhof and Oliver **Kohlbacher**: *BALL - Biochemical Algorithms Library 1.3*, *BMC Bioinformatics*, 11, 2010.

[HW91] **Haussler**, D , M. Kearns , N. Littlestone and M. **Warmuth**: *Equivalence of models for polynomial learnability*, *Inf Comput*, 95, 1991.

[HW11] **Herbold**, S. , Jens Grabowski and Stephan **Waack**: *Calculation and optimization of thresholds for sets of software metrics*, *Empirical Software Engineering*, 2011.

[KA07] **Kufareva**, Irina , Levon Budagyan , Eugene Raush , Maxim Totrov and Ruben **Abagyan**: *PIER: protein interface recognition for structural proteomics*, *Proteins*, 67, 2007.

[Kea98] **Kearns**, M: *Efficient noise-tolerant learning from statistical queries*, *Journal of the ACM*, 45, 1998.

[KL93] **Kearns**, M and M **Li**: *Learning in the presence of malicious errors*, *SIAM J Computing*, 22, 1993.

[KN04] **Keskin**, Ozlem , Chung Jung Tsai , Haim Wolfson and Ruth **Nussinov**: *A new, structurally nonredundant, diverse data set of protein-protein interfaces and its implications*, *Protein Science*, 13:pages 1043–1055, 2004.

[KS83] **Kabsch**, Wolfgang and Christian **Sander**: *Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features*, *Biopolymers*, 1983.

[KT04] **Koike**, A. and T. **Takagi**: *Prediction of protein-protein interaction sites using support vector machines*, *Protein Engineering Design and Selection*, 17, 2004.

[LBT05] **Landau**, Meytal , Itay Mayrose , Yossi Rosenberg , Fabian Glaser , Eric Martz , Tal Pupko and Nir **Ben-Tal**: *ConSurf2005: the projection of evolutionary conservation scores of residues on protein structures*, *Nucleic Acids Research*, 33, 2005.

[LC06] **Li**, Jing-Jing , De-Shuang Huang , Bing Wang and Pen **Chen**: *Identifying protein-protein interfacial residues in heterocomplexes using residue conservation scores*, *International journal of biological macromolecules*, 38, 2006.

[LL07] **Li**, Ming-Hui , Lei Lin , Xiao-Long Wang and Tao **Liu**: *Protein-protein interaction site prediction based on conditional random fields.*, *Bioinformatics*, 23, 2007.

[LR71] **Lee**, B. and F.M. **Richards**: *The Interpretation of Protein Structures: estimation of Static Accessibility*, *J.Mol.Bio*, 55:pages 379–400, 1971.

[LZ06] **Liang**, Shide Liang , Chi Zhang , Song Liu and Yaoqi **Zhou**: *Protein binding site prediction using an empirical scoring function*, *Nucleic Acids Research*, 34, 2006.

[MJ06] **Murakami**, Y. and S. **Jones**: *SHARP2: protein-protein interaction predictions using patch analysis*, *Bioinformatics*, 22, 2006.

[MT99] **Mammen**, E. and A. **Tsybakov**: *Smooth discrimination analysis.*, *The Annals of Statistics*, 27, 1999.

[MW01] **Morrison**, Kim L. and Gregory A. **Weiss**: *Combinatorial alanine-scanning*, *Current Opinion in Chemical Biology*, 5:pages 302–307, 2001.

[MZ08] **Merkl**, Rainer and Martin **Zwick**: *H2r: Identification of evolutionary important residues by means of an entropy based analysis of multiple sequence alignments*, *BMC Bioinformatics*, 9, 2008.

[NS04] **Neuvirth**, Hani Neuvirth , Ran Raz and Gideon **Schreiber**: *ProMate: a structure based prediction program to identify the location of protein-protein binding sites*, *Journal of molecular biology*, 338, 2004.

[NT97] **Nussinov**, Ruth , D Xu and C J **Tsai**: *Hydrogen bonds and salt bridges across protein-protein interfaces*, *Protein Engineering*, 1997.

[NW70] **Needleman**, S.B. and C.D. **Wunsch**: *A general method applicable to the search for similarities in the amino acid sequence of two proteins*, *J.Mol.Bio*, 48, 1970.

[OR03] **Ofran**, Yanay and Burkhard **Rost**: *Predicted protein-protein interaction sites from local sequence information*, *FEBS*, 2003.

[OR07a] **Ofran**, Yanay and Burkhard **Rost**: *ISIS: interaction sites identified from sequence*, *Bioinformatics*, 23, 2007.

[OR07b] **Ofran**, Yanay and Burkhard **Rost**: *Protein-Protein Interaction hotspots Carved into Sequences*, *PLos Computational Biology*, 3, 2007.

[PH95] **Pietrokovski**, Shmuel , Jorja G. Henikoff and Steven **Henikoff**: *The Blocks Database?A System for Protein Classification*, *Nucleic Acids Research*, 24, 1995.

[PM07] **Porollo**, A. and J. **Meller**: *Prediction-based fingeprints of protein-protein interactions*, *Proteins*, 66, 2007.

[QZ07] **Qin**, S.B. and H.-X. **Zhou**: *meta-PPISP: a meta web server for protein-protein interaction site prediction*, *Bioinformatics*, 23, 2007.

[RL05] **Res**, I , I. Mihalek and O. **Lichtarge**: *An evolution based classifier for prediction of protein interfaces without using protein structures*, *Bioinformatics*, 21, 2005.

[SD04] **Sen**, Taner Z , Andrzej Kloczkowski , Robert L Jernigan , Changhui Yan , Vasant Honavar , Kai-Ming Ho , Cai-Zhuang Wang , Yungok Ihm , Haibo Cao , Xun Gu and Drena **Dobbs**: *Predicting binding sites of hydrolase-inhibitor complexes by combining several methods*, *Bioinformatics*, 5, 2004.

[Sin64] **Sinkhorn**, Richard: *A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices*, *The Annals of Mathematical Statistics*, 35(2):pages 876–879, 1964.
URL `http://www.jstor.org/stable/2238545`

[SR73] **Shrake**, A. and J.A. **Rupley**: *Enviroment and Exposure to Solvent of Protein Atoms Lysozyme and Insulin*, *J.Mol.Bio*, 79:pages 351–371, 1973.

[ST03] **Storey**, John D. and Robert **Tibshirani**: *Statistical significance for genomewide studies*, *Proceedings of The National Academy of Sciences*, 100:pages 9440–9445, 2003.

[SW81] **Smith**, T.F. and M.S. **Waterman**: *Identification of common molecular subsequences*, *J.Mol.Bio*, 147, 1981.

[TS08] **Tinto**, Nadia , Adriana Zagari , Marina Capuano , Alfonso De Simone, Valentina Capobianco , Gerardo Daniele , Michela Giugliano , Raffaella Spadaro , Adriana Franzese , and Lucia **Sacchetti**: *Glucokinase Gene Mutations: Structural and Genotype-Phenotype Analyses in MODY Children from South Italy*, *Plos One*, 2008.

[Tsy04] **Tsybakov**, A.: *Optimal aggregation of classifiers in statistical learning*, *The Annals of Statistics*, 32, 2004.

[TW06] **Tillier**, Elisabeth RM , Paulo AS Nuin and Zhouzhi **Wang**: *The accuracy of several multiple sequence alignment programs for proteins*, *BMC Bioinformatics*, 47, 2006.

[Val84] **Valiant**, L.: *A theory of learnability*, *Communications of the ACM*, 27, 1984.

[Val85] **Valiant**, L.: *Learning disjunctions of conjunctions*, *Proceedings of the 9th International Joint Conference of Artificial Intelligence*, 1985.

[Wal04] **Wallach**, Hannah M.: *Conditional Random Fields: An Introduction.*, techrep, Department of Computer and Information Science, University of Pennsylvania, 2004.

[WH06] **Wang**, Bing , Hau San Wong and De-Shuang **Huang**: *Inferring protein-protein interaction sites using residue conservation and evolutionary information*, *Protein and Peptide Letters*, 13, 2006.

[WL06] **Wang**, Bing Wang , Peng Chen , De-Shuang Huang , Jing-jing Li Tat-Ming Lokc and Michael R. **Lyud**: *Predicting protein interaction sites from residue spatial sequence profile and evolution rate*, *FEBS Lett*, 580, 2006.

[WM06] **Waack**, Stephan , Oliver Keller , Roman Asper , Thomas Brodag , Carsten Damm , Katharina Surovcik , Peter Meinicke , Wolfgang Florian Fricke and Rainer **Merkl**: *Score-based prediction of genomic islands in prokaryotic genomes using hidden Markov models*, *BMC Bioinformatics*, 7, 2006.

[WM09] **Waack**, Stephan and Rainer **Merkl**: *Bioinformatik Interaktiv - Algorithmen und Praxis*, Weinheim, Wiley-VCH, 2009.

[Yac07] **Yaciuk**, Peter: *Co-Immunoprecipitation of Protein Complexes*, *Methods in Molecular Medicine*, 131, 2007.

[Zel11] **Zellner**, Hermann: *PresCont: Vorhersage von Protein-Protein Interaktionsflächen unter Verwendung struktureller und evolutionärer Eigenschaften*, phdthesis, Fakultät für Biologie und Vorklinische Medizin der Universität Regensburg, Regensburg, Germany, 2011.

[ZQ07] **Zhou**, H.-X. and S. **Qin**: *Interaction-site prediction for protein complexes: a critical assessment*, *Bioinformatics*, 23, 2007.

[ZS01] **Zhou**, H.-X. and Y **.Shan**: *Prediction of protein interaction sites from sequence profile and residue neighbor list*, *Proteins*, 44, 2001.

[ZST$^+$11] **Zellner**, Hermann, Martin **Staudigel**, Martin **Trenner**, Meik **Bittkowski**, Vincent **Wolowski**, Martin **Icking** and Rainer **Merkl**: *PresCont: Predicting Protein-Protein Interfaces Utilizing Four Residue Properties*, *Proteins: Structure, Function and Bioinforamtics*, 2011.

Appendix D

Acknowledgements

I want to thank all who helped me finish this thesis. Foremost my Professor Stephan Waack who supported and motivated me through the years. My thanks also goes to Carsten Damm for being my second reviewer and for the helpful questions over the years. To Steffen Herbold for sorting through the initial drafts and pointing out stylistic and other errors. To Markus Gieshold for the insightful discussions and the help on the implementations. To my sister Maya for helping me through the jungle of orthography. For making my thesis look more polished Vanessa End. And for some hints about how chemists view it all Malte Standera. I thank Stefanie Krinninger for noticing all the little things. For enduring my incessant questions about the formal procedure with the application my thanks goes to Annette Kadziora and Heike Jachinke. To the rest of my family for their support in all these years. And finally to all the friends who have kept me grounded.