AN ABSTRACT OF THE DISSERTATION OF

Ben Tribelhorn for the degree of Doctor of Philosophy in Computer Science
presented on August 14, 2013.

Title: Computational Methods for Protein-Protein Interface Prediction

Abstract approved: ______________________________

Michael J. Bailey

Protein-protein interactions underlie all biological processes and are a field of study that has wide implications throughout many other fields including medicine, genetics, biology, and ecology. Proteins are the building blocks and primary actors of life. They work together to accomplish virtually every task within a cell, including, metabolism, signal propagation, immune responses, and cell signaling. This problem is a logical successor to the Human Genome Project: now that we know so much about the DNA of living organisms, how do we advance our knowledge? The Human Genome and other DNA sequencing efforts have provided complete genetic sequences for more than 180 living organisms. However, these efforts fall short of describing or predicting life processes because the sequence of a protein is not enough to elucidate its function. Knowing this, the National Institute of Health started the Protein Structure Initiative, which seeks to increase knowledge of protein structure and has led to an increase in the number of known proteins struc-

tures. Unfortunately, even these efforts fall short as there are over 80,000 known protein structures but the function of many is completely unknown. The fledgling field of interface prediction seeks to use this wealth of structural information to be able to describe protein function and drastically increase our understanding of life processes.

Presented herein is a novel methodology for solving the protein-protein interface prediction problem leveraging a variety of Computer Science techniques. Specifically detailed is a process for decomposing this 3-dimensional problem into a feature extraction and classification problem using algorithms from computer vision and machine learning.

Computational Methods for Protein-Protein Interface Prediction

by

Ben Tribelhorn

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented August 14, 2013
Commencement June 2014

Doctor of Philosophy dissertation of Ben Tribelhorn presented on August 14, 2013.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

Ben Tribelhorn, Author

ACKNOWLEDGEMENTS

I would like to thank my Advisor, Dr. Mike Bailey, for his support and guidance without whom this work would not have been possible.

I would like to acknowledge Dr. Victor Hsu for his patience and support in this endeavor, especially in providing data for this work. I would also like to thank Camden Driggers for his explanations of Biochemistry.

To Aaron Moore, my business partner and friend, thank you for expanding my horizons and always demanding perfection.

Finally, to my friends and family, thank you for your patience and support.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

Chapter 1: Introduction

1.1 Overview

This work focuses on predicting the sites of protein-protein interactions. This problem is a logical successor to the Human Genome Project: now that we know so much about the DNA of living organisms, how do we advance our knowledge? The Human Genome and other DNA sequencing efforts have provided complete genetic sequences for more than 180 living organisms. However, these efforts fall short of describing or predicting life processes because the sequence of a protein is not enough to elucidate its function. Knowing this, the National Institute of Health started the Protein Structure Initiative, which seeks to increase knowledge of protein structure and has led to an increase in the number of know proteins structures. Unfortunately, even these efforts fall short as there are over 80,000 known protein structures but the function of many is completely unknown. The fledgling field of interface prediction seeks to use this wealth of structural information to be able to describe protein function and drastically increase our understanding of life processes.

Proteins are long chains composed of the 20 basic building blocks of life called **amino acids**. The longest known protein, titin, contains 34,350[1] amino acids. As amino acids can be connected in any arbitrary sequence, the space of possible

[1]Titin has many forms in the range 27,000-35,000 amino acids.

protein sequences is exponentially large: $> 20^{35,000}$. Within this space proteins have evolved to perform every life process and within this space are design options that would serve as cures for many diseases, but to locate and identify these is a huge computational challenge. Furthermore, limiting the search to solved protein structures presents a challenge as the number of structurally known protein interactions is just over 200, although data from other techniques (i.e. Y2H[2], TR-FRET[3], SPR[4], etc.) has demonstrated that virtually all proteins interact with other proteins or molecules either directly as their function or to modulate a function. Since the number of structures deposited into the Protein Data Bank[5] has doubled since 2006, this explosive growth of protein structure information motivates this field of study.

The computational challenge for determining if or how a pair of proteins interact is to identify the possible **interfaces** on each of the proteins and then determine the possible combined structure. Given a 3D structure of a protein, the goal is to classify correctly which surface regions are contained in a given protein-protein interface. Figure 1.1 illustrates various interface sites. This research presents a novel strategy for interface prediction based on surface segmentation and ensemble learning.

[2]Yeast two hybridization.
[3]Time Resolved Florescence Resonance Energy Transfer
[4]Surface Plasmon Resonance
[5]RCSB Protein Data Bank: http://www.pdb.org

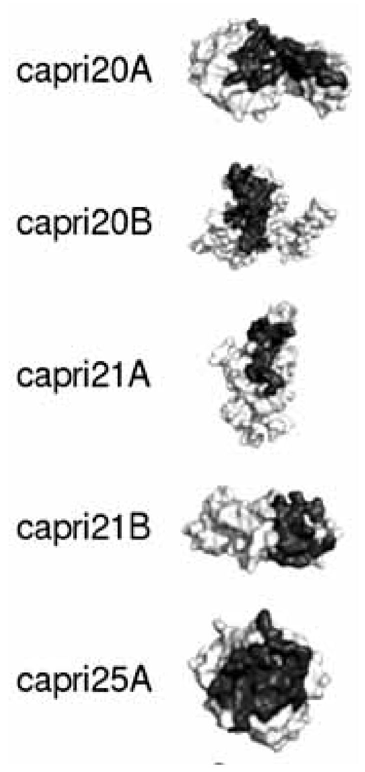


Figure 1.1: Interfaces (darkened) of CAPRI[1] targets, excerpted from [2].

Using techniques from computer vision, this problem is moved into a more traditional space where these methods have proved effective. Previous work on this problem focuses on amino acids as the principle actors and units for determination (either through sequence analysis or patches). In other words, the interface is assumed to be a collection of amino acids as opposed to a collection of independent atoms. Classifying interfaces based on the boundaries of amino acids is limiting and introduces an *a priori* bias to the learning problem that is simply illustrated when looking at solved examples: not all the atoms in the set of amino acids that "form" the interface are actually within the possible interaction zone (6 Å). By introducing a stochastic methodology to data segmentation prior to applying feature extraction this bias can be reduced or eliminated. This is the key to our research which advances the success in computational solutions in this field. This approach is unique in that features are not computed for arbitrary regions, but experimentally derived per amino acid. The advantage of this novel methodology is that it allows for the incorporation of higher level features that are absent in traditional approaches.

Solving the protein interaction problem today would broadly impact many fields, including medicine, where this ability would speed drug development.[6] Reducing the opportunity cost of drug development would be a tremendous boon. The total value of U.S. consumption of pharmaceutical drugs in 2009 was $300 billion.[7] Obviously, improving the process of pharmaceutical development would have real impact worldwide. Another important impact would be the ability to find the protein partners of a given protein and elucidate signaling pathways, which would greatly increase our knowledge of biology.

The actions of proteins from folding to docking are NP-Hard problems[8, 9, 10], even though they are a huge part of virtually every biological process. There has been limited success in the field to date, but since these processes complete within a fraction of a second at every moment, there *must* be a tractable method to predict (compute) these biological actions. Given the difficulty of these problems on a theoretical basis, it motivates new techniques for predicting what nature accomplishes with obvious ease and success.

1.2 Proteins

In order to define the problems addressed in this thesis, it is necessary to discuss proteins generally before going into detail on the problem statement. For reference, definitions of bolded typeface terms are provided in Section 1.4.

[6]The process of drug development would be sped up by finding modulations of a key interaction or an enzyme activity by identifying off targets and describing the mechanism of action before drugs have to be tested in the lab and clinically.

[7]http://www.bls.gov/ppi/pharmpricescomparison.pdf

Proteins are genetically coded as a linear chain of amino acids. The human body uses 20 amino acids as the building blocks for all the proteins it produces and uses.[8] These long chains of amino acids fold spontaneously in solution or with the help of other proteins into complex 3D structures that dictates the protein function. These structures (the spatial coordinates of the atoms of a protein) can be determined using NMR[11] or X-ray Crystallography[12]. Solved structures are usually deposited in the Protein Data Bank (PDB).

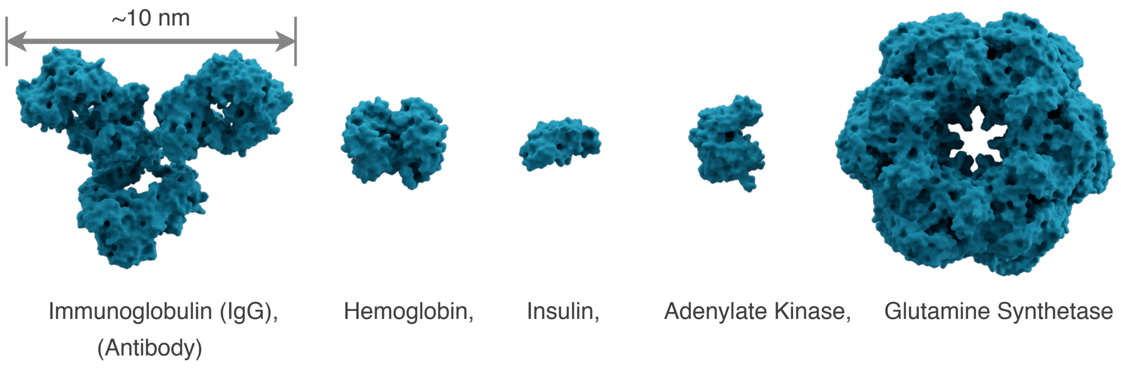


Figure 1.2: Various proteins.[9]

Proteins range in size and function, however as they are genetically coded it should be assumed that virtually every protein has or had a biological function. (Figure 1.2 shows a few common proteins[9].) To determine the function of a protein, we need to know what other proteins it interacts with and where on the surface it does so. Many proteins will interact both with themselves (forming dimers/multimers) and with others. This interaction is called **protein docking**

[8]There are actually two additional natural amino acids that are used very rarely in the human body.

[9]Edited http://en.wikipedia.org/wiki/File:Protein_composite.png
Also, titin has a diameter of about 1.5 μm

and it is incredibly challenging to predict, partially due to the fact that energy models provide an incomplete picture of the forces at work and because proteins can undergo **conformational change**, a possibly significant structure change.

As protein-protein interactions are the key to protein functions within biological processes, there is large interest in prediction. Current experimental methods of determining which specific proteins interact include *in situ* hybridization and mutation studies[10], both of which are incredibly labor intensive (months of time spent). Given the time is takes in lab, fast, highly accurate prediction methods would facilitate discovery in a variety of biological fields. Additionally, only a small set of complexes have been crystallized which is a requirement for determining the structure by x-ray crystallography. In addition, large proteins or complexes are not typically suited for structural determination by NMR. In order to accomplish this prediction goal computationally, one must address a two tier problem: 1) Do a particular pair of proteins interact, and if so 2) what structure do they form? There are many techniques that have been developed to address this problem holistically (described in the next Chapter), however given their low success, this research focuses on a narrower problem: interface prediction.

1.3 Interface Prediction

An **interface** is the surface region (collection of atoms) that interacts with another protein. Interaction is defined as a pair of atoms on opposite proteins within 6 Å of

[10] Also SPR, TR-FRET, etc.

each other (nucleus to nucleus)[11]. For reference 1 Å $= 10^{-10}$ meters which is on the scale of the length of a Carbon-Hydrogen bond, and 1.4 angstroms is approximately the radius of a water molecule[13]. Unfortunately, there are many challenges and limitations in protein-protein interface prediction. Current state-of-the-art methods are able to produce reasonably correct docking models for rigid-body interactions[2]. However, in common cases that include conformational changes and/or weak/transient interactions[12], the current approaches fail spectacularly[3]. The simple reason for this is that there is no single property sufficient for unambiguous identification of the interface. In fact, there is considerable consternation in the field as to which properties are actually useful for the identification of the docking interface. One reason for this lack of specificity is that the set of known interfaces is in the low hundreds as the experimental determination of complexes requires a lengthy and often unsuccessful process using X-ray Crystallography or NMR spectroscopy to map the interacting molecules both separately (free) and the docked complex (bound). Since computational prediction is incredibly challenging, often researchers will relax the problem and use energy-based simulations to produce a few hundred possible complexes and then attempt to rank them using specific domain knowledge.

[11]Some use a definition of 5 Å, others define interfaces as the loss of surface accessibility: *i.e.* what atoms get "buried" from docking. This document uses the definition above for all work with the rationale that interacting atoms are the most relevant to predicting complexes. 6 Å is the absolute maximum range for virtually all atomic interactions.

[12]Not all protein interactions form a stable or "permanent" complex. For example, transient interactions would include enzyme-based processes.

1.3.1 CAPRI

There is a specific forum for evaluating protein complex predictions called the Critical Assessment of PRedicted Interactions (CAPRI)[1]. Taken from their website: "*CAPRI is a blind prediction experiment managed by capri management . Its targets are unpublished crystal or NMR structures of complexes, communicated on a confidential basis by their authors to the CAPRI management. Participant predictor groups are given the atomic coordinates of two proteins that make biologically relevant interactions. They model the target complex with the help of the coordinates and other publicly available data (sequence, mutations etc.), and submit sets of ten models for assessment on the CAPRI Web site. After the prediction round is completed, the CAPRI assessors compare the submissions to the experimental structure, [and] evaluate the models on criteria that depend on the geometry and biological relevance of the predicted interactions.*" The scoring and ranking metrics specified by the CAPRI event are one of the standards for reporting prediction results. As the CAPRI group is now encouraging predictions of all types of interactions, in this work the term "protein-protein interaction" is used loosely. The goal in this work is to address any and all molecular docking, even if an actor is not a protein per se.

1.4 Biological Definitions

Amino acid: Building blocks of life consist of a side-chain which is unique for each of the 20 amino acids, composed of an alpha amine and a carboxyl group, which can react together to form polymers (proteins).

Residue: Synonym for amino acid. Used in this document to indicate the relevant portion of the exposed surface of an amino acid.

Protein: Covalently bonded chain of amino acids.

Docked Complex: A structure formed by non-covalent protein-protein interactions. These can include more than two constituent molecules.

Conformational Change: Structural change in a protein that occurs in docking.

Conservation: The replacement of an amino acid by a functionally similar analogue. The term implies a DNA mutation that "conserves" function.

Interface: Set of atoms at a docking site. Definitions can vary, i.e. buried atoms or the distance between molecules. For more, refer to Section 1.3.

Chapter 2: Background and Prior Work

2.1 The Problem

The problem of predicting protein-protein interactions focuses on being able to determine the bound complex formed given two structures of free molecules (a representation of the each molecule in an unbound state). Each protein structure contains two tiers of information: atomic level and residue level. Most researchers use the amino acid sequence as the basis for extracting features. Each amino acid has its own set of properties including solvation, interface propensity, and hydrophobicity. Amino acids can also be evolutionarily conserved which is another feature.[14, 15, 16] Finally, the most nebulous is the coordinate structure of the protein (each atom location). The 3D cartesian positions of each atom allows for the computation of inherent features including (but not limited to): surface accessibility, tertiary structure (neighbors), secondary structure, and the shape of the surface. There is no single feature that allows for the prediction of an interface, so we must combine features and consider any additional information that can be extracted.

Complexes are typically divided into different categories to further simplify the problem. The two types of interactions are permanent (obligate) and transient (non-obligate). Transient interactions typically form heterodimers (complexes with two components that are not the same molecule). Whereas permanent complexes can form heterodimers, homodimers (the same protein forming a complex with itself), or multimers[1]. Permanent complexes are generally easier to predict for a number of reasons. The most obvious is that to gain example data, we have to crystallize the complex which biases the dataset for prediction as it is much easier to get permanent complexes to crystallize. Secondly, the interfaces for these types of complexes tend to be larger, flatter, and better conserved than transient interfaces.[17, 18, 19] The next level of distinction is in the types of actors: 1) Enzyme/Inhibitor or Enzyme/Substrate , 2) Antibody/Antigen, 3) Others, 4) Antigen/Bound Antibody[2]. The two antibody subtypes are the most difficult and as recently as 2008 it has been suggested that attempting to evaluate prediction performance on those types of complexes detracts from the progress that can be made on the "easier" categories.[2, 18, 20] The classification of difficulty tends to reflect the degree of structural change that the components undergo when forming a specific complex. Remember that protein folding, the formation of the structure from a strand of amino acids, is a challenging problem in its own right, so it is not surprising that predicting conformational change is similarly difficult.

[1]A multimer can include both multiples of the same molecule as well as a variety of different molecules.

[2]http://zlab.umassmed.edu/benchmark/

2.2 Integrating Features & Prediction

Although success is difficult to quantify (see Section 3.2.1 for evaluation criteria), there is a forum, CAPRI, for evaluating efforts to solve the docking problem. As of 2009, the most successful methods for predicting protein docking combine algorithms into a multiple stage search problem.[3] Specifically this is done as follows:

1. Global rigid body search - Generate 1000s of possible permutations
2. Identify sub-regions of interest - Best geometric matches, other criteria
3. Locally refine docked complex - Monte Carlo methods
4. Rank resultant model complexes - Predict top 10

Method	Search	Protein flexibilty	Examples
Global methods based on FFTs or geometric hashing	Global	Minimal, smooth potential	ZDOCK[21] PIPER[22] PatchDock[23] MolFit[24]
Medium-range methods: i.e. Monte Carlo minimization	Limited region, stochastic	Moderate, side chains, some loops	RosettaDock[25] ICM-DISCO[26]
Restraint-based docking	*a priori* specification of interface residues	Can be substantial	HADDOCK[27, 28]

Figure 2.1: Classification of docking methods[3]

The methods for computing these steps fall into three primary categories: geometric matching, Monte Carlo minimization, and restraint-driven local search (See the table in Fig. 2.1). Often, machine learning algorithms are used to optimize an objective function from the input parameters, although there are some approaches that use specifically designed objective functions. However, the latest methods, including this work, are now focusing on adding a prior step which seeds regions of interest. Using the features discussed previously, an attempt is made to determine the surface regions that are possible interfaces.

Integrating the various properties of proteins is typically done in one of two paradigms. The first method is to consider residues as the components of interest and to determine the set of residues that form the protein interface. The second way is to generate surface patches that are constructed of sets of residues, and then attempt to determine the patch or patches that form the interface.[29, 30, 31, 32, 33, 34, 35] Also, within many efforts is an analysis at the sequence level which is where conservation has been shown to be a distinctive feature. The downside to these methods is that the sequence does not directly include structure information. Regardless of the method, some form of energy minimization has to be simulated to find a complex.

Since there is a lack of features that allow for an unambiguous identification of protein interfaces, many solve this problem by using machine learning techniques to work with weak features. Also, with such a wide set of potential types of interactions, methods that attempt to solve them all concurrently tend to run into tradeoffs between over- or under-prediction. Previous works have predominantly applied Artificial Neural Networks (ANN) [19, 32, 35, 36, 37, 38], Support Vector Machines (SVM) [29, 30, 34, 39, 40, 41, 42, 43, 44], Bayesian Networks (BN) [4, 31, 33, 45], and Conditional Random Fields (CRF) [46] to this problem. SVMs and BNs have seen the more success overall than ANNs.

2.3 Machine Learning Methods

The idea of an artificial neuron originally proposed by McColloch and Pitts (1943) has evolved into the Artificial Neural Network. It has been used in the protein interface prediction problem by a number of groups, however ANNs suffer in the context of learning because they do not have any convergence guarantee and often fall into a local minimum. They can also be very difficult to properly parameterize (an NP-hard problem). Often, a better solution is to use a Support Vector Machine (SVM).

Support Vector Machines attempt to find a hyperplane that segments the data into two classes. SVNs maximize the margin (distance from nearest data points to the dividing hyperplane), but in real world cases they can be sensitive to outliers and noise. To address this, SVMs have parameterized error functions that act as penalties to the maximization. (Although, this can lead to other issues). They owe their popularity to their tendency to produce results typically superior to most other methods in a variety of problem spaces. However, SVMs are easily biased when training data is not balanced, and for the protein interface problem this is a definite issue as interfaces are generally far smaller than half of the exposed surface area. SVMs have been used extensively in this problem: Wang et al.[39] make a nice comparison of different SVM parameterizations.

Graphical models (BNs & CRFs) seem a natural fit for encoding *a priori* information into a framework for inference. These models have been used primarily as patch predictors, the goal being to identify regions of residues that are likely to be part of an interface. There has been work that shows considering only local information from a single complex (intramolecular propensities) can work as a good predictor to narrow the search space beyond geometric matching.[47] Bayesian networks are often the graphical model of choice because it is straightforward to use domain knowledge to define the variables and the structure for inference, as opposed to Markov networks which are undirected. The difficulty with graphical models is that inference on large networks is very slow because it can be reduced to a maximum clique tree problem (NP-hard), so use of this method has to limit graph sizes.

Assi et al. use Bayesian Networks to identify “hot spots” or critical components of interfaces[4]. Figure 2.2 compares experimentally determined interfaces and critical residues with their predicted results for two proteins. Bradford et al.[45] also use a patch analysis technique, in their case they use simpler topologies with a few more variables. They tried to accomplish two things, one to predict if the patch is part of the interface, and secondly to predict if the type of binding is obligate or non-obligate (permanent vs. transient). They report better results than previous work using SVMs.

Unfortunately, using BNs requires human design to form a proper network and

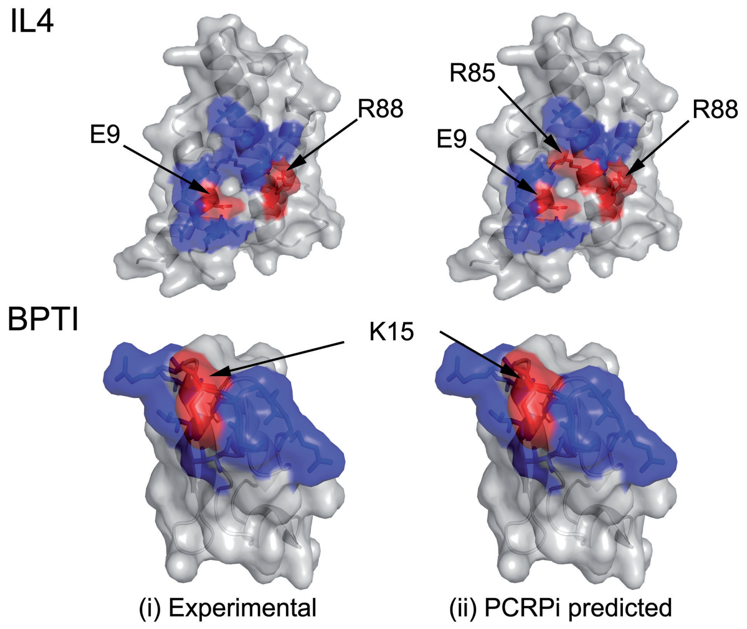


Figure 2.2: Blue depicts the interface and red highlights the critical residues.[4]

tends only to identify portions of an interface rather than the whole interface. As there are no successful methods that determine the entire interface from these key residues, it is clear that additional work to isolate the interface is needed.

2.4 State of the Art

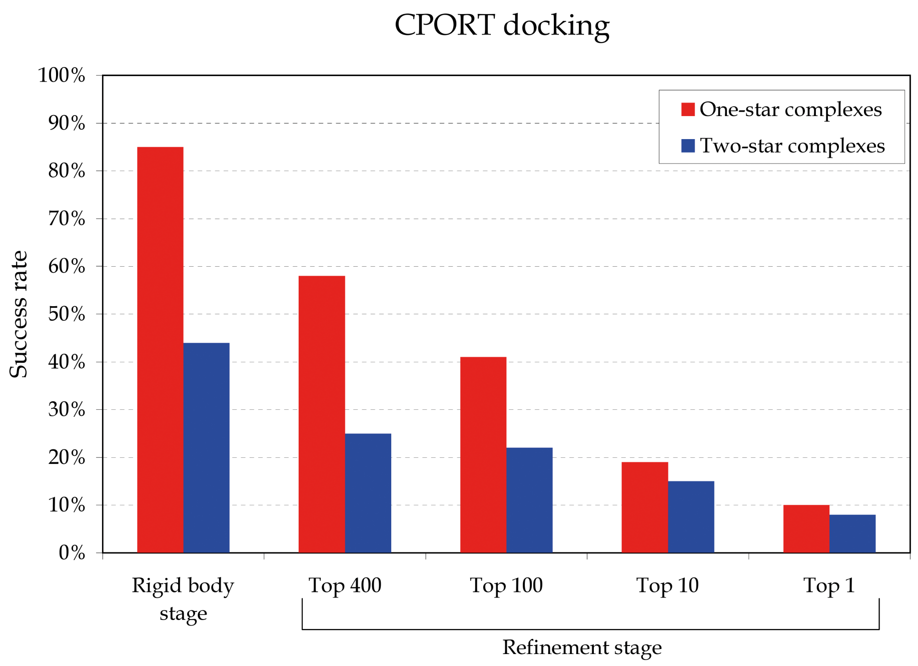
Based on the class of interaction, the best prediction method will vary. It is clear from the CAPRI event, there is no obvious front-runner in solving this problem. For a sense of the "State of the Art:" Using consensus information with HADDOCK[3][27], Figure 2.3 shows results from 2011[6] that illustrate the rate of success

[3]HADDOCK is a molecular simulation package that takes a prediction and uses it as restraints for defining the system being simulated.

addressing this problem. One, two, three star complexes is a rating for the quality of the solution where more stars is better. *Ab initio* docking shows HADDOCK without consensus information (aka ‘vanilla’ HADDOCK).

Refer to Section 2.2, and you will recall that most methods addressing the protein docking problem start with a geometric (rigid-body) search and then refine using stochastic methods to allow the proteins to change shape. The graphs shown in Figure 2.3 show, on the y-axis, the maximum possible success rate at each stage of this process. In other words, if an oracle looked at the set of possibilities and pointed out the correct prediction, how many correct predictions remain under consideration. Depicted clearly here: more difficult complexes (two and three star) are so difficult that even the rigid body stage misses most of the correct solutions. (This is usually because the harder complexes involve more conformational change). Notice in the top graph, the success rate for the Top 1 ranked prediction is under 10%. As the top prediction is *the* prediction, the system success rate is indicated by the final column. This illustrates the extreme difficulty of the problem and delineates the bar for success. As these results integrate predictions from many different researchers, it is one of the top predictors in this realm.

We illustrate the HADDOCK system results here because our system, to be detailed in Chapter 5 produces an output (interface prediction) that would be given to HADDOCK or a similar system. The goal of this research is to increase the performance of docking predictors by doing a better job of localizing the protein interface than current systems.



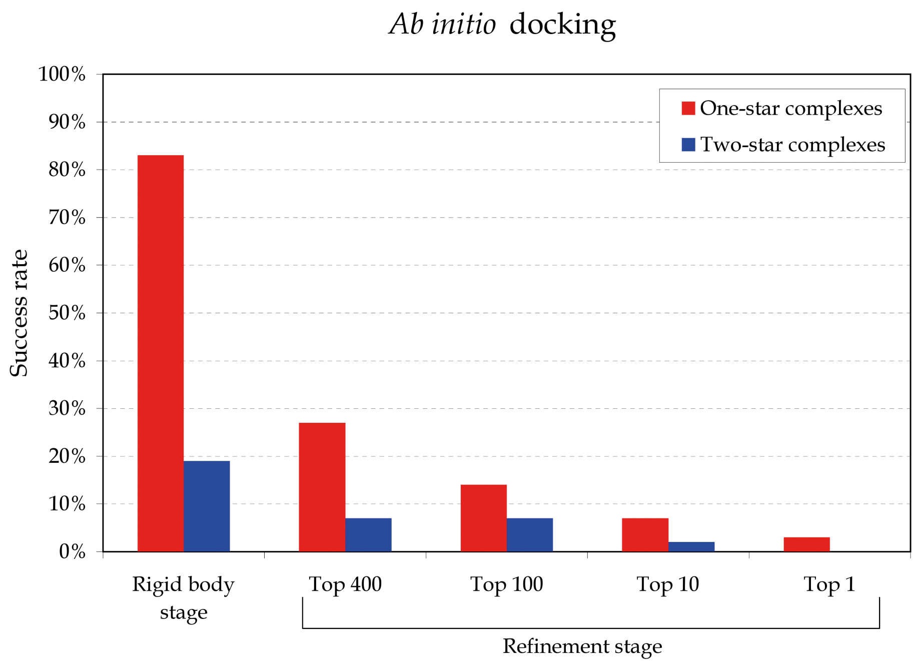


Figure 2.3: Docking results for CPORT-driven docking using HADDOCK (top), compared to HADDOCK *ab initio* docking (bottom). The figure shows the percentage of cases for which at least one structure of that quality was generated during the rigid body stage (10,000 structures), and the top 400 (all refined structures), 100, 10 and 1 of the refinement stage. One-star and two-star criteria correspond to the CAPRI definitions.[5] For the rigid body stage, the fnat criterion is not taken into account. doi:10.1371/journal.pone.0017695.g001 Image excerpted from [6].

2.5 Opportunity

Prior work formulates this problem as a single question of "What is the docking interface for a complex?" This paper argues that the problem is actually a two-tier question. First, given an arbitrary protein, what are the interfaces? Virtually every protein has a function; many of them dock with multiple other proteins (including themselves as in homodimers). This question has been addressed by some communities.[4, 30] The second tier is to determine if a complex forms given two proteins and their sets of interfaces. The motivation to separate these questions is partially due to the methods that can be applied, and in the difficulty in creating a completely new docking software. The work presented in the following chapters is appropriate for integrating with existing software, especially ones that expect input seeds like HADDOCK.

Chapter 3: The Problem, Data, & Methods

3.1 Problem

Predicting protein interfaces is the problem that this work focuses on as it is a required first step towards solving the docking problem. Given a protein, we know *a priori* that it has a function, but what is unknown is how it achieves that function. Since many complexes are collections of docked proteins we can expect that a protein has at least one but in many cases more than one interface. This leads to a difficulty when trying to apply machine learning methods because we cannot be fully confident in providing negative examples of interfaces. This is due to the fact that even if we know that a pair of proteins dock to form a complex, their binding interface many not be the only one. So when using the available data on protein interfaces, we need to employ methods that are robust to noisy data to compensate for an implicit bias in the training data. The methods that follow were investigated with this consideration in mind.

3.2 Methods

Presented in the following chapters are two methods for predicting protein interfaces. They are very different approaches, while the core of this work explores the latter it is important to view the investigations chronologically to see the evolution of this work.

The first approach discussed is a variance analysis (Chapter 4). With inspiration from the statistical analysis in Bordner et al.[30], this presents a technique we developed to incorporate the frequency of atomic occurrence. This work focused on the simple case of attempting to predict all interfaces without using negative information. This, unfortunately, fails at even the simpler test of recognizing a set of only interfaces.

The core of this work focuses on a second method that, in essence, transfers the interface prediction problem into a standard computer vision/machine learning paradigm. We use ideas from image analysis to create a pipeline for predicting interfaces: segmentation, feature extraction, classification, and agglomeration. This transference allows for the use of many traditional methods with novel adaptations. The implementation presented uses a multiple scale decomposition to extract more information than traditional residue- and patch-based methods. This system for identifying interfaces was used to generate entries to the 28th Round of the CAPRI event. For an objective discernment of the variations between parameter selections, the next section discusses the metrics used in this document. These are not the typical “success” metrics as those can be more forgiving. See the results discussion for more on the overall success of this work.

3.2.1 Error Metrics

In later sections, success is reported in terms of **Precision, Recall, $\mathbf{F_1}$** (also **F-score**) as opposed to a simpler success rate or accuracy. The reason for this is that the problem is biased towards predicting the negative as interfaces are a small portion of the overall protein. Reporting accuracy would be misleading in that under-prediction is favored (always predicting no interface means an accuracy $\approx 0.7 - 0.9$ or 70%-90% success[1]). So, we want to focus on terms that show how well we are actually solving the problem at hand. To do this we count atom-by-atom True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). Using these counts we can compute the metrics listed in the Equations 3.1, 3.2, 3.3. Each of these terms gives a value zero to one inclusive that can be roughly viewed as a percentage. For a sense of scale, most top predictors result in under 10% success depending on how it is reported. In this document it is the F_1 value most closely resembles "success"-rate. *In all graphs in this paper, a larger value (for each of these metrics) indicates a better success rate.*

$$Precision = P = \frac{TP}{TP + FP} \tag{3.1}$$

$$Recall = R = \frac{TP}{TP + FN} \tag{3.2}$$

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{3.3}$$

[1]Recall that the size of an interface typically ranges from 10-30% of the surface of a protein. So it is easy to achieve a high accuracy when failing to predict.

A short rationale: Precision is important because in this problem being able to localize and distinguish a portion of an interface addresses the hardest part of the problem and a high precision results from successfully identifying interface subsets without predicting a large portion of the surface. However, this becomes undefined when under-predicting (we typically define $P = 0$ if $TP = 0$), so Recall is also necessary to address how much of the interface is being missed. F_1 is a useful way to combine these two terms because it is the harmonic mean and it reports an overall success metric that is not biased by true negatives. F_1 is often used in the field of information retrieval for measuring search, document classification, and query classification performance.[48] Also note that as precision and recall are inversely related, attaining a high F_1 requires success at both measures.

3.3 Data

Each protein structure contains two tiers of information: atomic and amino acid parameters. Each atom is associated with a 3D position, exposed surface area, radius, polarity rank, and bond type while each amino acid is associated with a charge, polarity rank, secondary and tertiary structure, exposed surface area, and possibly equivalency information (common mutation that still contributes to docking known as evolutionary conservation). Unfortunately, the difficulty with this problem formulation is that it is not clear how best to combine parameters or even to calculate them.

Surface accessibility, for example, can be calculated in multiple ways (potentially skewing prediction). Most often it is defined using a cutoff for absolute area exposed or in relative terms, as a percentage of maximal accessibility. Obviously the goal from the perspective of computer science would be to learn from the data, but it has been shown that the type of complex being predicted has a strong impact on the utility of the indicators.[2] For this reason permanent complexes and transient complexes are often predicted separately, partially due to the difference in the "known" interactions. Transient interactions cannot be mapped as simply as permanent structures that lend themselves to crystallization-based analysis. This means that within the research and results presented in this document, there is a bias towards permanent complexes as that is the majority of the available examples.

We use the Lee-Richards molecular surface definition which is computed using the NACCESS software program.[49, 50] This program outputs the .asa and .rsa files used in this work. The .nbc files (non-bonded contacts) are generated using the CNSsolve software program.[51] The non-bonded contacts form the set of atoms that this work refers to as the **protein interface** for a given complex. Obviously computing these differently would change the results considerably.

The sets of complexes and some computed features used in this research were provided by Dr. Victor Hsu[2]. The selection of complexes is based on a standard benchmark set for this problem. Chapter 4 uses Benchmark 3 data[52] and Chapter

[2]Department of Biochemistry and Biophysics, Oregon State University

5 uses a subset of Benchmark 4 data[53].[3] These benchmark sets also rank the difficulty of each complex as easy, medium, and hard and provide the type of interaction. Difficulty is rated by the iRMSD[54], which is the deviation of the structure between the bound and unbound forms (before and after docking). We do not furnish these categories of difficulty to the learning algorithms used in this work.

As protein docking requires multiple actors to create a complex, images will specify the named complex that a specific protein is part of *i.e.* 1NCA and then will reference which component with the number 1 or 2. So 1NCA_1 refers to the first half of the 1NCA complex, and 1NCA_2 refers to the second half. The distinction between first or second is arbitrary, but the structures can be visually identified by comparing them to the components listed in the benchmark set. Typically, although not always, each component of a complex will have its own name. These components can be docked complexes themselves, however in these cases they are treated as a single molecule.

[3]http://zlab.umassmed.edu/benchmark/

Chapter 4: Variance Analysis

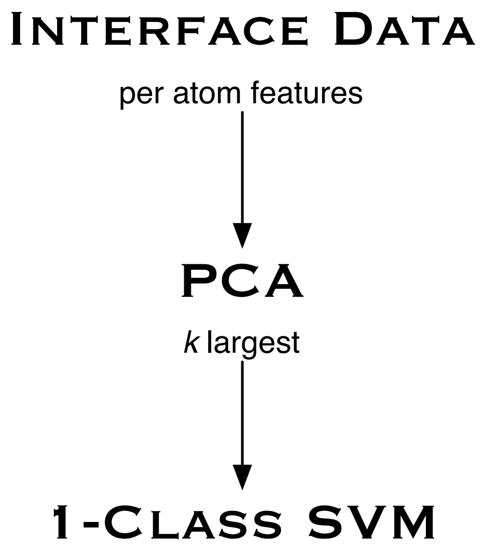


Figure 4.1: The variance analysis pipeline for classifying protein interfaces.

This chapter details a pipeline that uses variance analysis over a set of available features to attempt to predict the protein interface. Figure 4.1 shows the steps in this process, starting with atomic data. Each atom has a number of attributes, many of them are categorical and converted to the binary 0 or 1. Each atom is also associated with the amino acid that it is a component of, and has specific values shared by all atoms in a given type of amino acid. As the number of atoms in a patch can be variable, we need to convert the features to a fixed length attribute vector for classification. To achieve this goal Principle Component Analysis (PCA)[55] is employed to capture the variance of the features over different patches. The assumption is that the distribution of atoms would be indicative of a protein interface. Finally, a classification method is needed.

The typical uses of machine learning have focused on the basic version of each algorithm. The inspiration for the approach detailed in this chapter is an adaptation of Support Vector Machines (SVM)[56]. Using the knowledge that all protein interfaces are biologically favored and an effective active process, we extend this to the concept of a one class SVM. A single class SVM is different from a normal SVM in that instead of trying to distinguish two classes by maximizing the margin between them, it attempts to create a hypersphere around the data that correctly identifies the single class of interest (protein interfaces). This is particularly relevant to to this problem because we cannot merely assume that given a complex the entire remainder of the surface save for the interface is not a protein interface with another molecule. In fact, we know that proteins tend to bind with more than one other protein often, proteins bind with themselves.

Recall that Support Vector Machines are a construct for finding a linear separation through optimization of non-linearly separable original data through the use of kernel functions. In a one class SVM, the hypersphere (instead of hyperplane) that is created attempts to find a distribution that assumes the need to eliminate outliers and noise. So we can expect a one class SVM to achieve less than 100% success on a training dataset depending on the input data and the choice of kernel function.

4.1 Feature & Parameter Selection

This approach does a patch-style[1] prediction of interface/non-interface. As noted previously, the patches used are the sets of atoms that are already known to be interfaces. For the dataset, we used a set of 124 experimentally determined structures. I extracted the non-bonded contacts (atoms closer than 6 Å between the two proteins) for each protein.

The best set of atomic features consisted of $f = 20$ as determined by trial and error. We tried many variations of features (including QUANTA solvent accessibilities), but because so many variables are categorical with a large number of categories (20-40) the sparsity was detrimental to performance. In some cases this led to an error rate considerably worse than random guessing. The set of features is listed in Figure 4.2. Features 7-13 reduce amino acids from 20 to 7 categories. The reason to include so many features as categories instead of as real valued features is because SVMs tend to work better with all features normalized in the range $[0, 1]$.

As the number of interacting atoms can vary wildly based on the function and the involved proteins, the data needs to be normalized into a constant size feature vector. Often, other methods will use a frequency analysis to describe patches with a constant length feature vector. In this work, we used PCA to estimate principal components of length f. The method for computing PCA is as follows: given a matrix D_n with n data points, you have to center the data to remove the mean

[1]Patch-based prediction relies on segmenting the surface of a protein into surface regions. These are often simply computed using BFS on residues given a central residue.

	Feature	**Notes**
1	Residue Polarity Rank	Trinquier et al.[57]
2	Residue Charge	-1, 0, or 1
3	Secondary Structure	alpha helix, 3-10 helix, beta strand beta turn, beta turn no H-bond, 3-1 (extended left-hand) helix, other
4	Solvent Access	[49, 50]
5	Surface Area	[49, 50]
6	Van der Waals Radius	
7	Small and nonpolar	GLY, ALA, PRO
8	Small and polar	SER, CYS, THR
9	Large and nonpoalr	VAL, LEU, ILE, MET
10	Large and polar	ASN, GLN
11	Basic (pH)	HIS, LYS, ARG
12	Acidic (pH)	ASP, GLU
13	Aromatic	PHE, TRY, TRP
14	Hydrogen	
15	Sulfer	
16	Oxygen	
17	Nitrogen	
18	Carbon	
19	0	
20	0	

Figure 4.2: Feature vector that is constructed for each atom prior to PCA. Only features 4 and 5 are truly real valued, the rest are binary or effectively categorical (i.e. there are only so many types of atoms, so the Van der Waals radius is one of five values).

(Eqn. 4.2), then do Singular Value Decomposition (Eqn. 4.3), and then extract the vectors in V based on the largest eigenvectors (diagonal in Σ).

$$C_n = I_n - \frac{1}{n} \tag{4.1}$$

$$M = C_n D_n \tag{4.2}$$

$$M = U\Sigma V \tag{4.3}$$

We explored multiple open source solvers for SVD including OpenCV [58] and SLEPc/PETSc [59, 60]. We found that OpenCV when compiled performs drastically differently on different systems and on different operating systems on the same physical hardware.[2] In most builds, OpenCV would hang during the computation and never halt. We use the SLEPc extension to PETSc because it runs more consistently and produces smaller variations to minor changes to initial conditions. Retrospectively, the high numerical instability of this process recommends against this process.

By choosing the $k = \frac{3}{4} \cdot f$ top components from PCA, there is a constant feature vector of length $kf = 300$. This was determined by trial and error. The reason k is a fraction of f is that the smallest components from PCA represent noise in the data and end up being tremendously misleading to classification. Two input features are always set to zero, and they are intuitively similar in effect to the bias input to neural networks. The effect is that they allow for PCA to throw

[2]This is caused by the difference in compilers and how they optimize. Clearly the validation tests in OpenCV do not stringently check the equation solver.

out irrelevant or noisy data when we choose only the top k components. Without these two extra features, the prediction is much worse.

Finally, these feature vectors are given to a one-class SVM. The added benefit is that there is no natural way to include negative information without just generating surface regions randomly. It doesn't make sense to include the entire rest of the molecule as a single "non-interface" because it would go against the implicit knowledge of relative planarity and interface size as well. Additionally if you were to include every possible "non-interface" within a size range, the bias towards the negative class would be extreme (far beyond 100:1). This bias in training data has been an issue in other SVM approaches where the whole molecule is fed in as two regions (interface, non-interface).[46]

We used the OpenCV SVM implementation[3] which has two parameters γ and ν as well as the choice of kernel function. We can choose between kernels: linear, polynomial, sigmoid, and radial basis function (RBF). A discussion of kernel functions is beyond the scope of this work, so suffice to say that we chose to use an RBF kernel as they tend to perform the best in many machine learning contexts. The parameter ν embodies the idea of an activation function and reducing it from the default $\nu = 0.5$ value makes it more predictive, we settled on a value of $\nu = 0.25$ using grid search. And, γ is the parameter for the radial basis function kernel (RBF):

$$K(\chi_i, \chi_j) = e^{-\gamma||\chi_i - \chi_j||^2} \tag{4.4}$$

[3]The OpenCV implementation is based on *libsvm*[61], a popular program.

We found that the optimal range for this parameter was approximately $0.18 < \gamma < 0.3$. Ultimately, to achieve the best performance, grid search is used to settle on the best parameter choices.

As 116 complexes were used in this experiment, we have 232 interfaces to classify. Over an 8-fold cross validation we achieve an overall training error rate of 36% and a testing error rate of 94% (using $\gamma = 0.2$). We would expect to achieve closer to 100% success on training data since we are not providing any negative information. Additionally, since the testing error rate is so high (only succeeded in one fold), we have found that this method lacks enough input information to succeed at this problem. See Figure 4.3 for the results.

	Training	**Testing**
Cross Validated Error	36%	94.4%
Num. Successes	1039/1624	13/232

Figure 4.3: Eight-fold cross validated results of labeling known protein interfaces.

This methodology for predicting interfaces is unfortunately a very poor performer. Using just positive examples, we need to achieve better performance in the cross validated test sets to consider extending this method. Ultimately, this investigation shows that a variance analysis is not enough to predict interfaces. However, this lack of success led to rendering interfaces atom-by-atom (as opposed to highlighting amino acids) which illustrated the key insight that the next chapter is predicated upon (depicted in Fig. 5.3).

Chapter 5: Multiple Scale Analysis

This chapter presents a novel process for predicting protein interfaces, illustrated in Figure 5.1. This approach transfers the problem into a computer vision style problem, by creating a segmentation of the surface of an input protein. Then it computes features for regions of the surface, which can then be classified by machine learning techniques. Labeled regions are then agglomerated to output a predicted interface.

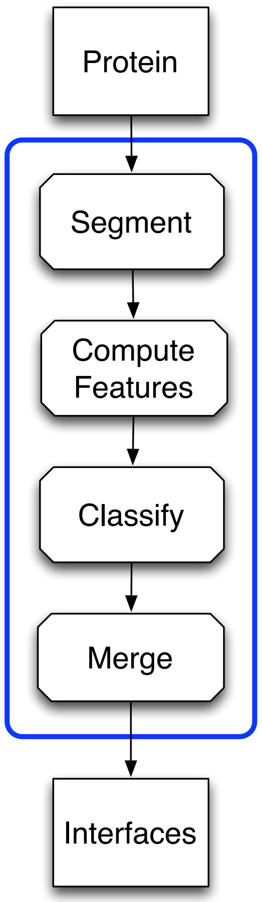


Figure 5.1: The multiple scale analysis pipeline for predicting protein interfaces independent of complex information.

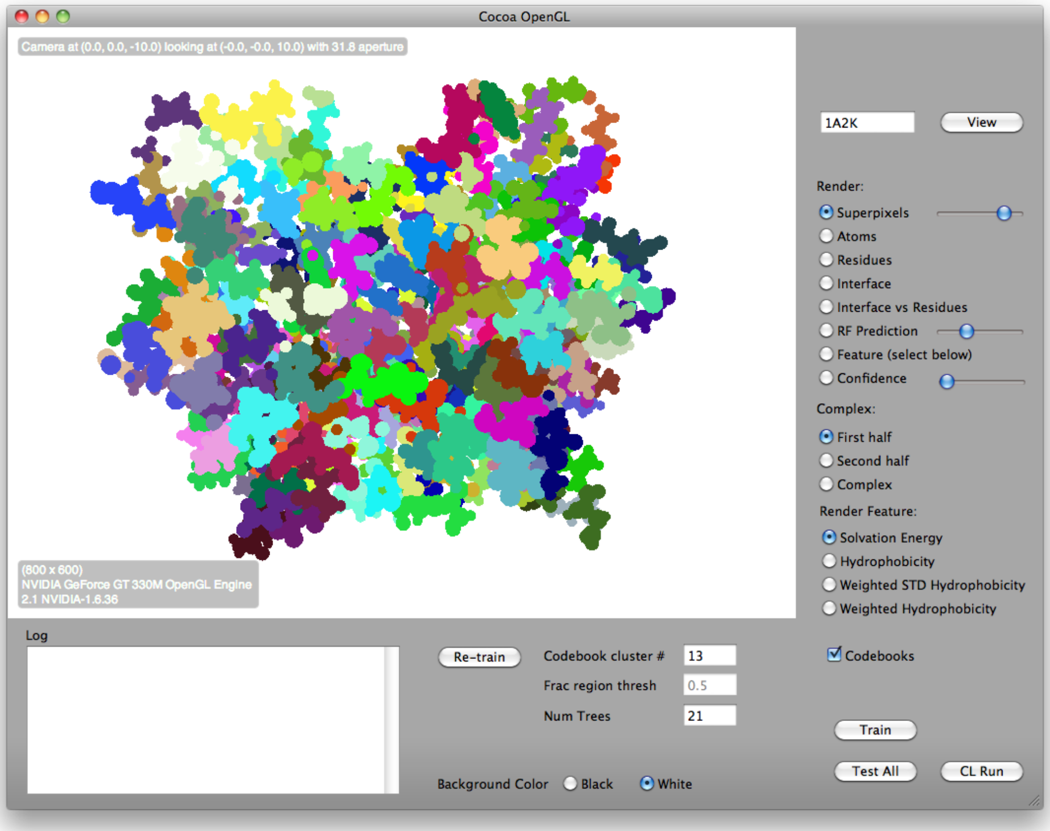
The process of segmentation, feature extraction, classification, and merger is very successful in computer vision and is a well established research area. By using a computer vision approach, a variety of studied algorithms can be leveraged towards the protein interface prediction problem. The intuitive parallel to this problem: an image analysis problem tries to identify a specific object within an image; we are attempting to identify an interface on the surface of a protein. More specifically, instead of identifying a collection of pixels, in this domain the goal is to label the correct set of atoms.

This Chapter uses a subset of Benchmark 4 data; see the listing in Appendix A for the distinction of which complexes are considered "Easy."

5.1 Computational Framework

The exploration of a multiple scale analysis technique followed the investigations of the previous chapter. As it was not completely clear, visually, why the previous method failed, using a custom graphical interface (Figure 5.2) as a wrapper for the analysis framework led to the key observation which is discussed in Section 5.1.1. This program is the source of all of the unattributed images in this document that illustrate proteins.

In order to construct a custom analysis pipeline that is modular, this program imports various data (.pdb, .asa, .nbc) and loads it into a custom format. (See Appendix B). This program then uses a combination of original code and the open source library OpenBabel[62] to compute various features for surface regions on the molecules. We modified OpenBabel to return atom specific descriptors as it is typically used to summarize a whole molecule.



Camera at (0.0, 0.0, -10.0) looking at (-0.0, -0.0, 10.0) with 31.8 aperture

(800 x 600)
NVIDIA GeForce GT 330M OpenGL Engine
2.1 NVIDIA-1.6.36

1A2K

Render:
[✓ Superpixels]
Atoms
Residues
Interface
Interface vs Residues
RF Prediction
Feature (select below)
Confidence

Complex:
[✓ First half]
Second half
Complex

Render Feature:
[✓ Solvation Energy]
Hydrophobicity
Weighted STD Hydrophobicity
Weighted Hydrophobicity

Codebook cluster #	13
Frac region thresh	0.5
Num Trees	21

[✓ Codebooks]

Background Color Black [✓ White]

Figure 5.2: Computational Framework Interface

5.1.1 Protein Surface Segmentation

Prior work in this field has focused on various specific subsets of interactions: homodimers/multimers, heterodimers, antigen/antibody, enzyme-inhibitor, etc. A great majority of previous work uses residue-level analysis as the basis of their methods: frequencies, patches, conservation, polarity, etc., at which point a learning algorithm is applied to the data. Atom-level information is sometimes included, but always with respect to the residue grouping.

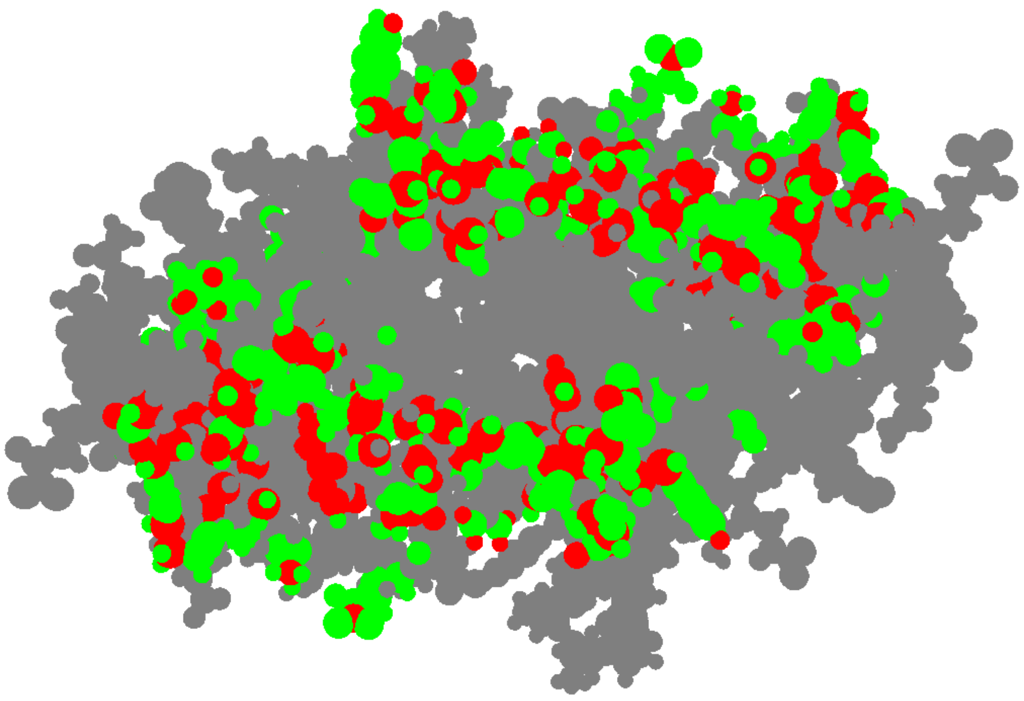


Figure 5.3: Red indicates the interface, green indicates any surface accessible atom part of a residue that contains at least one interface atom. (1AK2_1)

Separating interactions based on the boundaries of amino-acids is limiting and introduces an *a priori* bias to the learning problem. Figure 5.3 illustrates this problem. When classifying by residue, every green colored atom would be an over-segmentation of the interface (shown in red). By introducing a stochastic methodology to data segmentation prior to applying feature extraction this bias can be reduced or eliminated.

The proposed solution detailed in the next section is illustrated in Figure 5.4. It shows how a different segmentation can offer different boundaries which helps to alleviate the problems inherent in predicting based on residues. The region size shown is chosen to match the approximate size of amino acids.

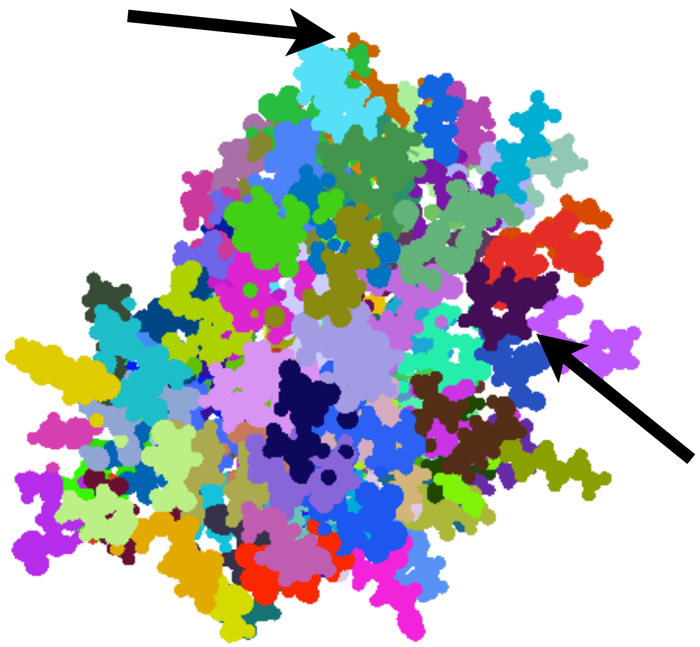
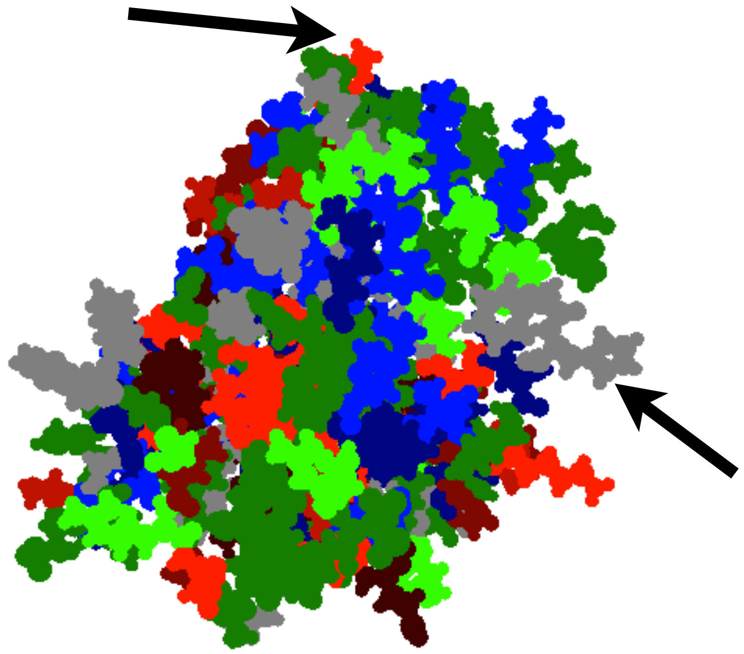



Figure 5.4: **Top:** Computed surface regions. Each color indicates a different region. **Bottom:** Amino acids. Each color indicates a type of amino acid. **Arrows:** The arrows point out two locations where the computed regions differ from traditional amino acid boundaries. The top arrow points out that two different regions (green and brown) are segmented where in the lower image there is just one amino acid (red). Region size is approximately 26 atoms which is similar in size to a residue. (1A2K_2)

5.2 Segmentation

In order to address the problem identified in the previous section, we introduce a novel method that computes surface regions that do not adhere to typical residue boundaries. To compute regions, a segmentation algorithm needs to be selected. The question in segmentation problems is often one of how to decompose the data. There are several methods that lie in two major categories: hierarchical and partitioning (flat). Since clustering relies on a similarity metric and there isn't an obvious way to say that two atoms are in a "similar" grouping especially given the inherent structural regularity, using a hierarchical algorithm would be a poor choice. Instead we need to find the protein interface without any prior knowledge except spatial locality (which suggests spatial clustering is a proper choice). Common and effective partitioning algorithms include k-means[63], mixture of Gaussians[64], and Spectral Clustering[65].

This section presents a clustering method inspired by an image segmentation algorithm that is an extension of k-means clustering, called SLIC Superpixels[66]. The idea is a simple adjustment to the algorithm: a random or equally spaced set of initial clusters are selected over the surface of a molecule and clusters are updated using a weighted distance metric. In this case, we compute both Euclidean and geodesic distances.[1] These surface regions have a number of benefits. In addition to not being limited by the residue boundaries, they are still predominantly

[1]Geodesic distance is the shortest surface-path between two points whereas Euclidean distance is the shortest path between two points. The tips of a horseshoe are a good example: low Euclidean distance, high geodesic distance. See Figure 5.6.

contiguous surfaces that can be classified. This is more true to real life interactions where docking sites depend on the specific exposed atoms and not the whole amino acid.

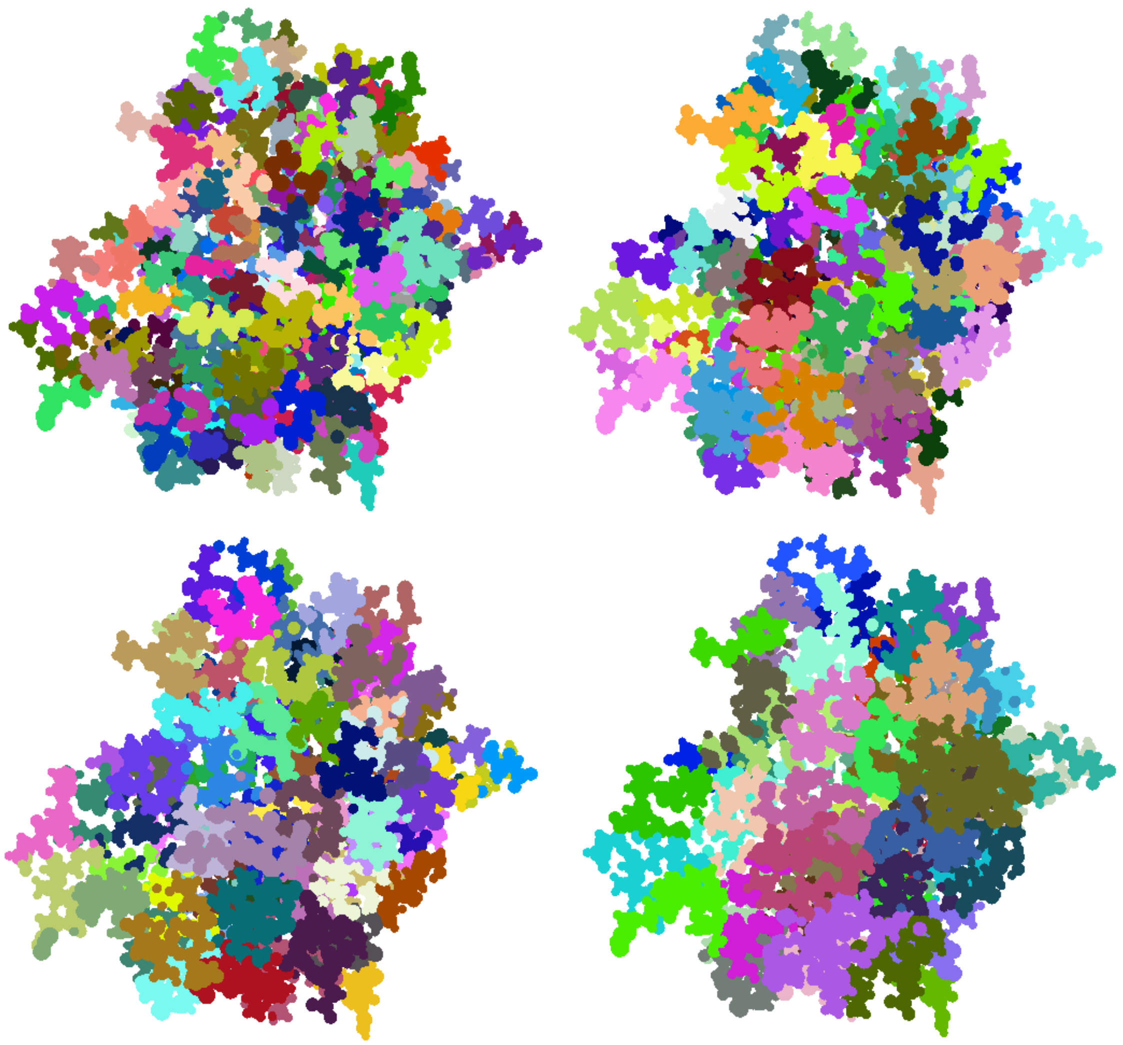


Figure 5.5: From left to right, top to bottom the approximate region size in number of atoms is 20, 34, 46, 70. (1A2K_1).

This method of segmentation is best contrasted with patch-based methods. Bordner et al.[30] used 15 residue patches based on an Euclidean distance computation between C_α atoms[2]. Obviously, this introduces a bias because the C_α could be buried and not reflective of surface geometry. This novel method for surface segmentation has tremendous advantages over current methods, especially because this allows for multiple scale segmentation and classification.

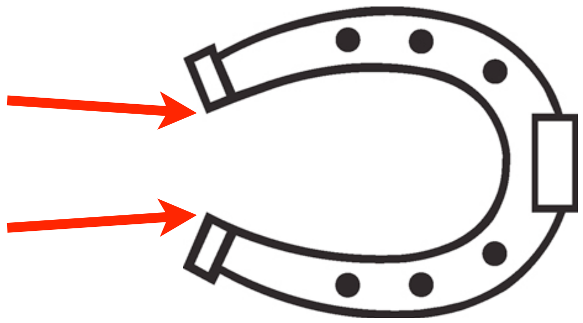


Figure 5.6: Geodesic distance: the distance between the tips of the horseshoe traveling along the surface of the object.

Picking initial cluster centers was improved by employing kmeans++[3] which has the benefit of producing typically better clusterings[67]. Example segmentations at various sclaes are shown in Figure 5.5. The difference between segmentations at the same scale is shown in Figure 5.7 which shows high stability in the clustering. This means that to successfully over-segment the surface, clustering at different scales is a requirement because we wouldn't get meaningfully different segmentations by clustering multiple times at the same scale.

The choice of scale for clustering is a challenging one. Bordner et al. used patches composed of 15 residues.[30] Using an approximate size of 20+ atoms per

[2] C_α is the primary backbone Carbon for a given amino acid.

[3] kmeans++ is an adaptation to traditional kmeans where the $2, \ldots, k$th centers are picked with a probability proportional to their distance to the nearest cluster center (instead of randomly).

amino acid, this suggests a region size of about 300 atoms. Also, given that an interface should be 10-30% of the surface, regions should not be computed unless at least four can be created at a given scale. This suggested an approximate range for segmentations of 20-300 atoms per cluster. However, regions on the larger side of this range results in zero predictions. This shows that there isn't enough distinctive information in the feature set to reliably predict any portion of any interface using larger regions as it hides local effects. This is due to the fact that you'd have to segment, by luck, in alignment with the true interface. Based on visual analysis, different size regions tend to allow different areas to be predicted, so a multiple scale segmentation is recommendable.

Refer to Section 6.1 for a discussion of the segmentation granularity that is recommended by this problem. In practice, 10 segmentation levels are used (Eqn. 6.1).

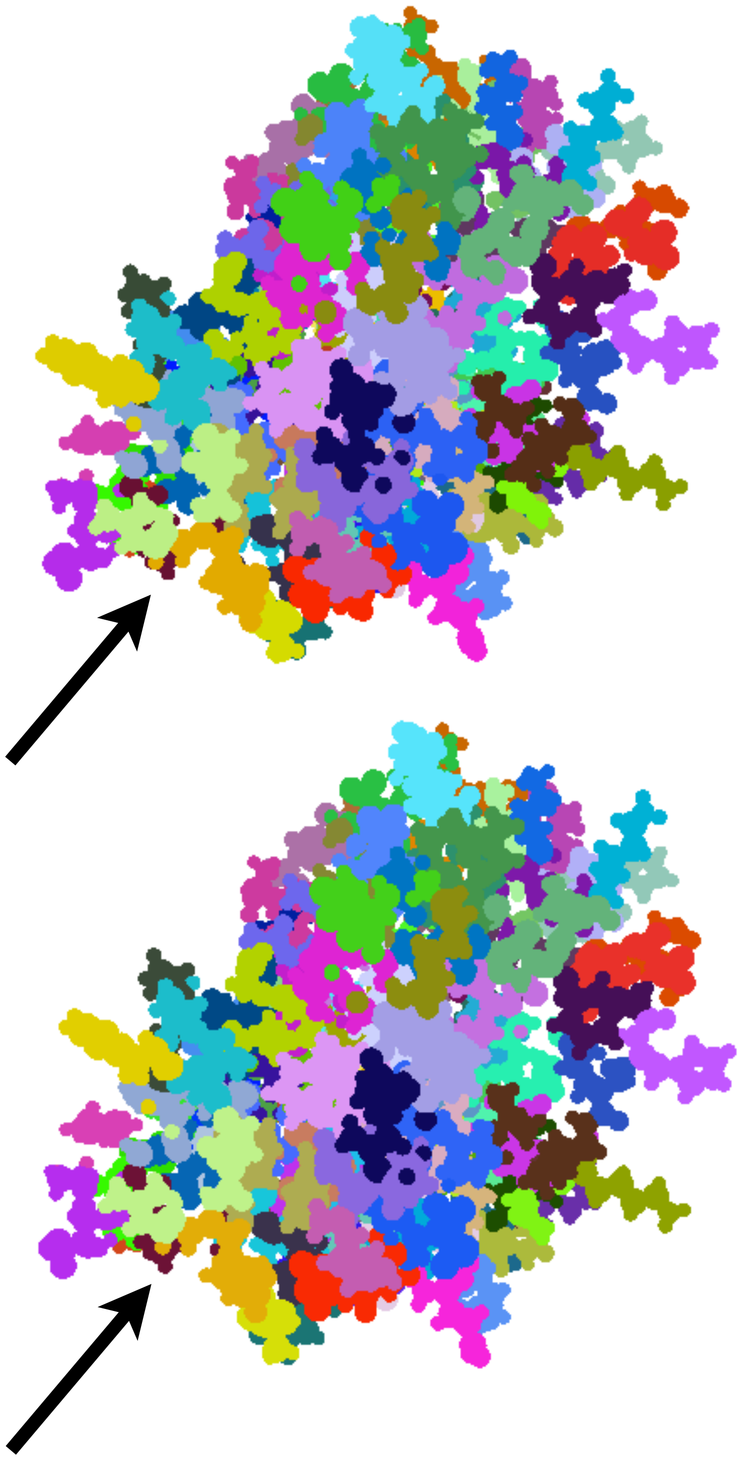


Figure 5.7: Clustering stability using kmeans++ seeding, the arrow points out a small difference. Region size, $n = 26$. (1A2K_2).

5.3 Dynamic Property Computation

The next step is to compute properties for each surface region. Feature selection is a large problem in machine learning, so the choice in this effort is to include as much information as possible and use a classification method that works well with weak features. In image analysis there are many common choices including edges, corners, texture, color, and higher level features including shape and structure metrics. It is not clear that there are obvious parallels to the surface of a protein, so the features we employ are biological. While we could have chosen to include shape features this could be misleading as conformational change can easily modify the final shape. (Addressing conformational change is left to the second tier problem of docking). The proxy for this specific concept is exposed surface area (SASA) as that is the amount of initial physical exposure during docking. Additionally, some ideas from image analysis could be applied to these features, such as gradients (a proxy for "edges"). This is left for future work because to produce the most accurate set of continuous values one would need to compute not just the feature for every region, but for every atom's local neighborhood which is out of the scope of this thesis.

Features are computed for each segmented region. Some are normalized based on the nominal region size which is defined as exposed surface area. Many of these features included were chosen due to the relative ease of computation, especially as some are supported by OpenBabel. Footnotes reference SMARTS descriptors computed using OpenBabel, see Appendix C for more information.

It is worth noting that generally features are experimentally derived for amino acids and therefore computing them for an arbitrary collection of atoms, instead of for the whole molecule or single amino acids, is unique within the field.

In order to see the different features, the following pages illustrate each feature. As the values are biological, to render them in each case, the values are normalized and trimmed manually in order to show the range of highest variability in the feature, for the specific molecule depicted. The idea here is just to see that each feature presents a different view of the molecule. These images use a red to blue to black color scale. The scale is linear from red to blue with a normalization factor, however in almost every image this caps large values. Black is used to indicate extreme values at the edge of blue (typically where the number is zero).

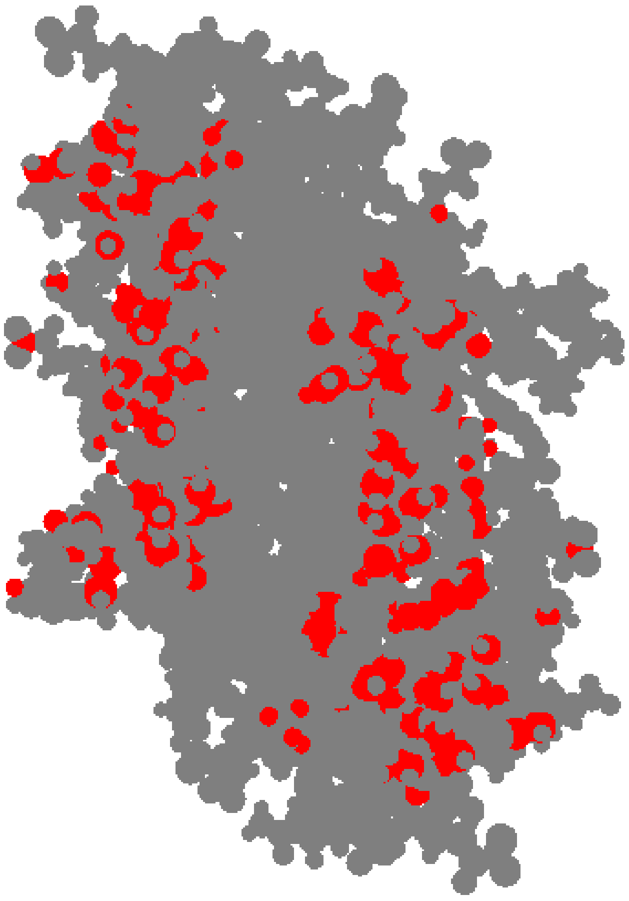


Figure 5.8: Interface of 1A2K_1.

- Solvent accessible surface area (SASA) - What is revealing in this image is that a large swath of the surface (purple tone) has only a moderate amount of surface area exposed relative to the number of atoms in the clusters.

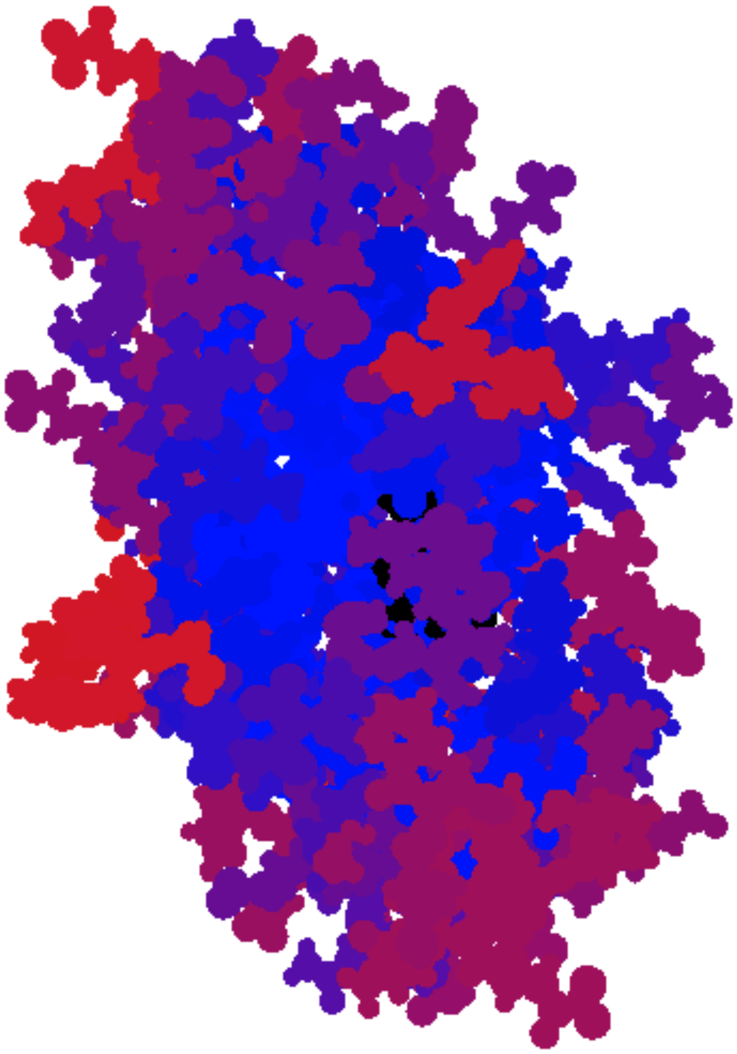


Figure 5.9: Solvent accessible surface access (SASA). Red to Blue (maximum to minimum), Black means zero access. (1A2K_1)

- Solvation energy weighted by surface area - The energy potential for water to bind to a surface. Two versions are computed (Kyte & Doolitle and Sharp et al. see [68]). Notice here there are only a few bright spots indicating a large potential.

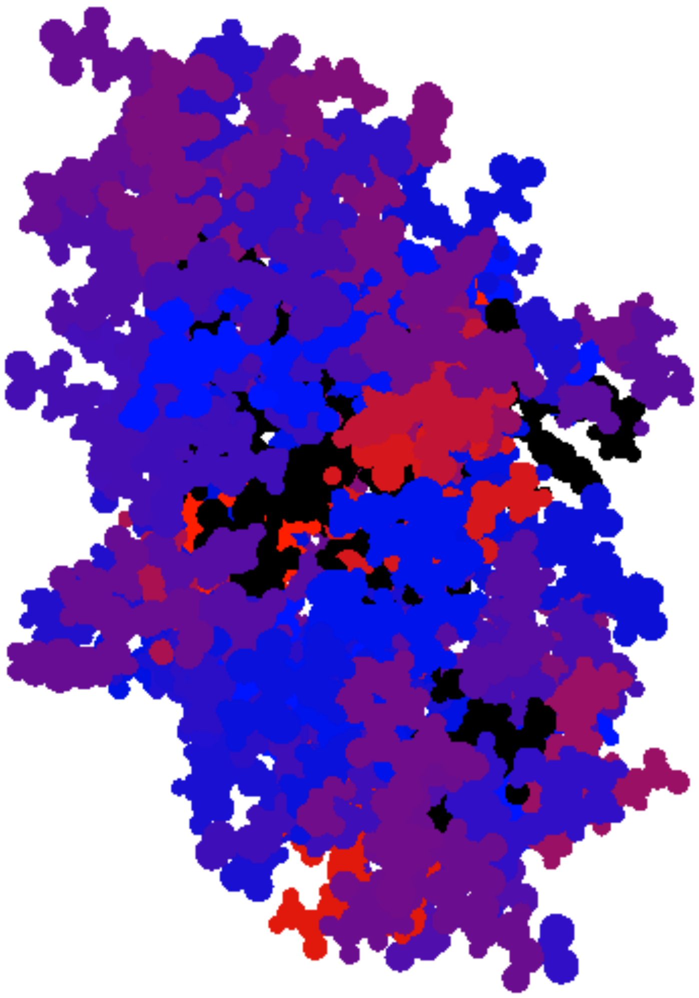


Figure 5.10: Solvation energy normalized by surface area. Red to Blue (negative maximum to minimum), Black means positive. (1A2K_1)

- Hydrophobicity [69, 70, 71] - Repulsive potential of a surface region, relates to the electron cloud locations.

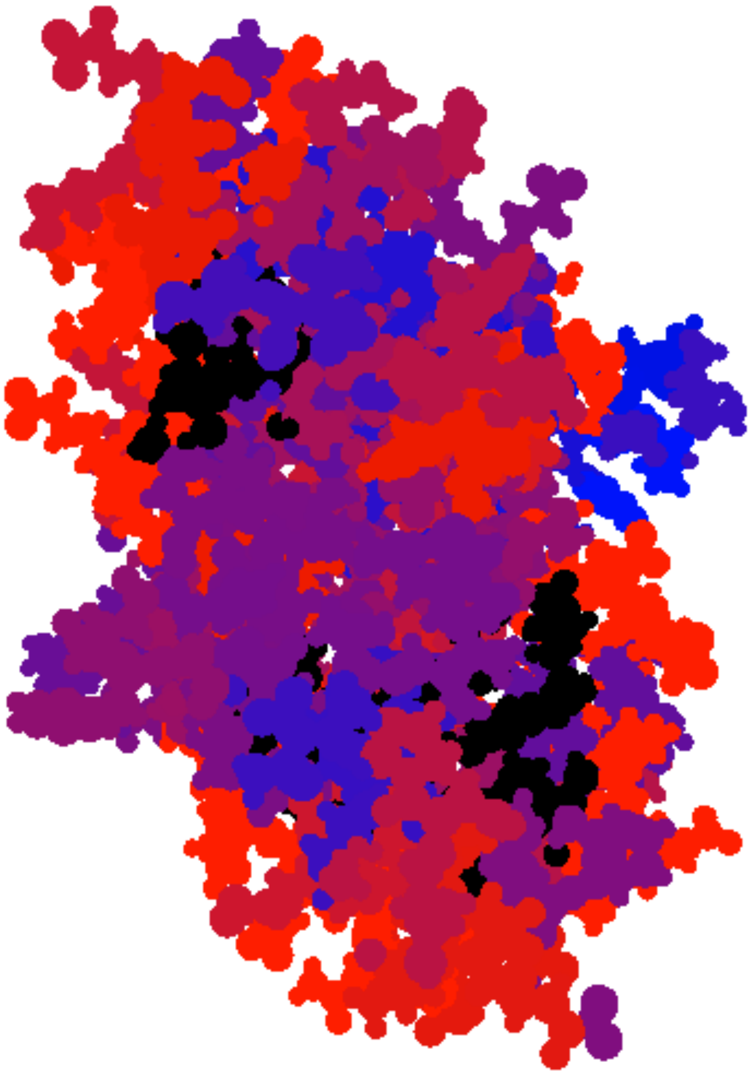


Figure 5.11: Hydrophobicity. Red to Blue (negative maximum to minimum), Black means positive. (1A2K_1)

- Hydrophobicity weighted by surface area

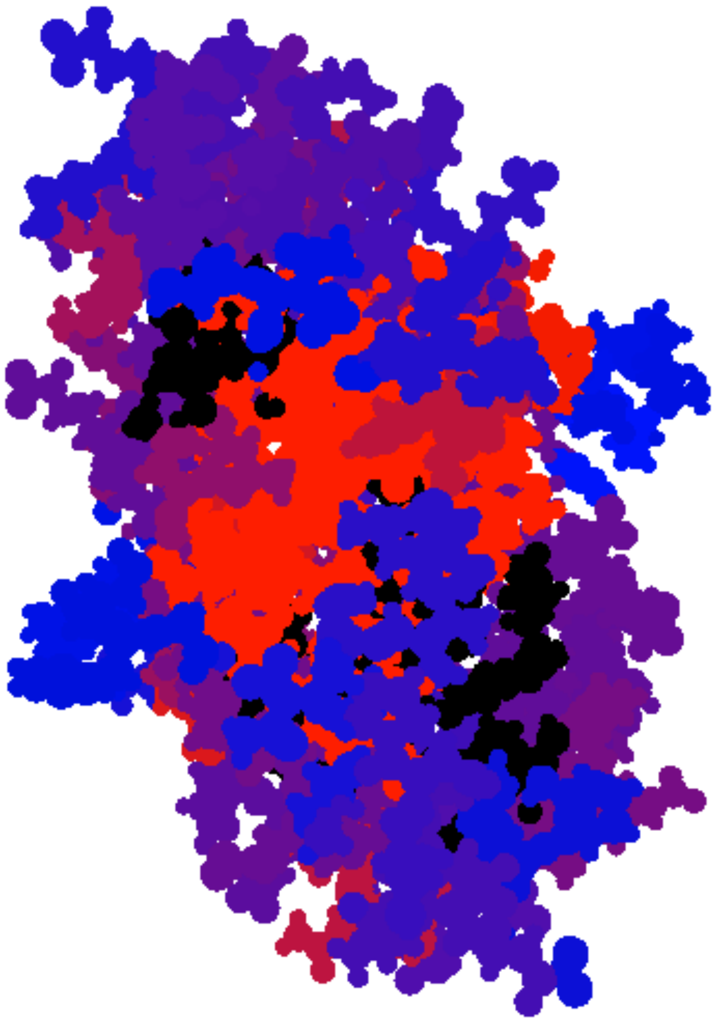


Figure 5.12: Hydrophobicity normalized by surface area. Red to Blue (negative maximum to minimum), Black means positive. (1A2K_1)

- Standard deviation of weighted hydrophobicity - Intuition: variance within a region contributes additional information.

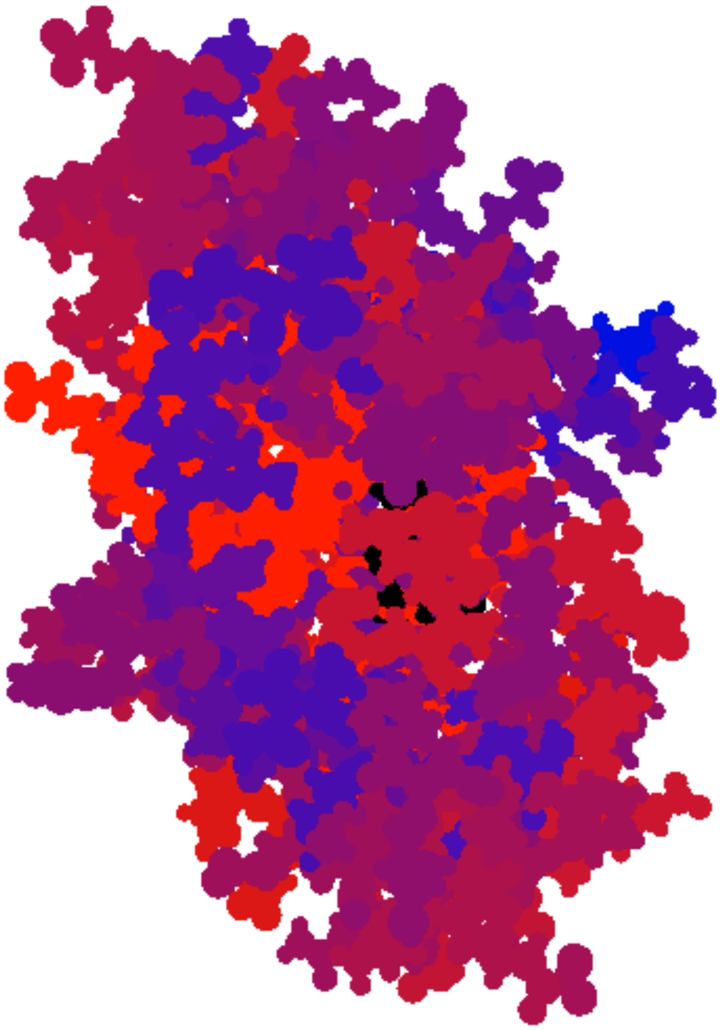


Figure 5.13: Standard Deviation of Hydrophobicity Weighted by SASA (surface area). Red to Blue (maximum to minimum), Black means zero. (1A2K_1)

- Hydrogen Bond Donors[4] - Two versions computed.

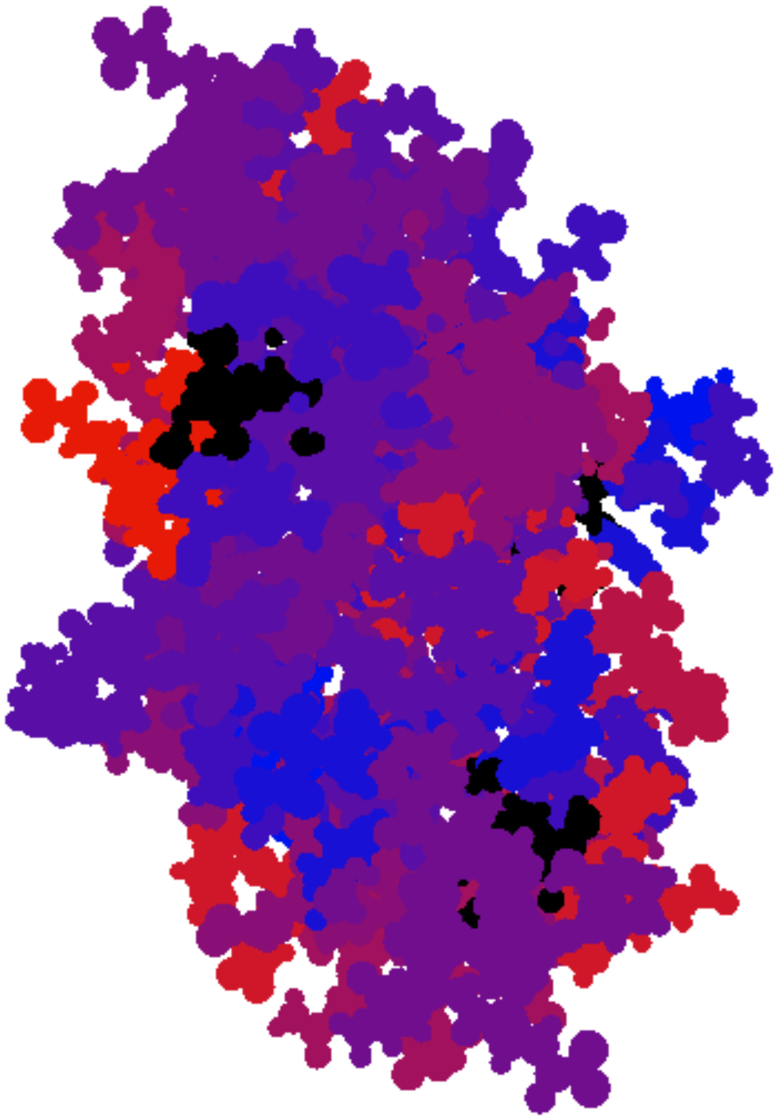


Figure 5.14: Number of Hydrogen Bond Donors. Red to Blue (maximum to minimum), Black means zero. SMARTS [!#6;!H0] (1A2K_1)

[4]http://www.ra.cs.uni-tuebingen.de/software/joelib/tutorial/descriptors/descriptors.html

- Hydrogen Bond Acceptors[72][5] - Two versions computed.

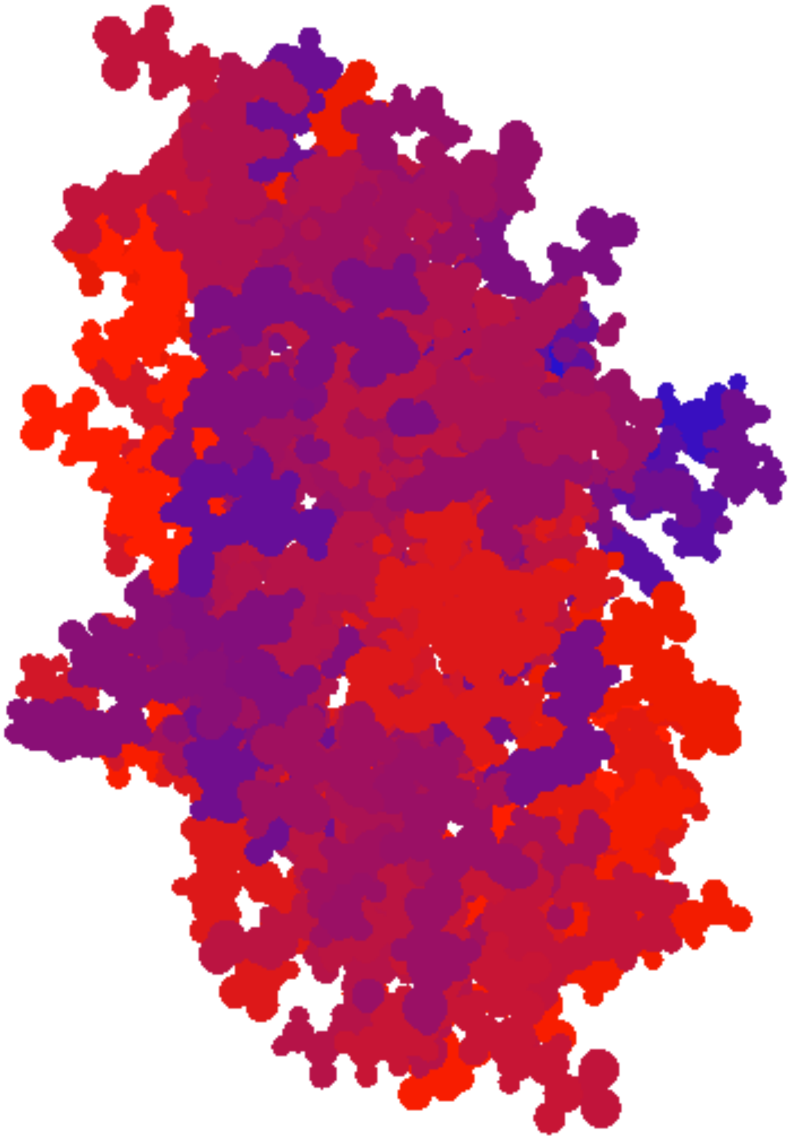


Figure 5.15: Number of Hydrogen Bond Acceptors. SMARTS [$([!#6;+0]);!$([F,Cl,Br,I]);!$([o,s,nX3]);!$([Nv5,Pv5,Sv4,Sv6])] Red to Blue (maximum to minimum). (1A2K_1)

[5]http://www.ra.cs.uni-tuebingen.de/software/joelib/tutorial/descriptors/descriptors.html

- Rotatable Bonds[6] per atom. This is an indicator for possible conformational change.

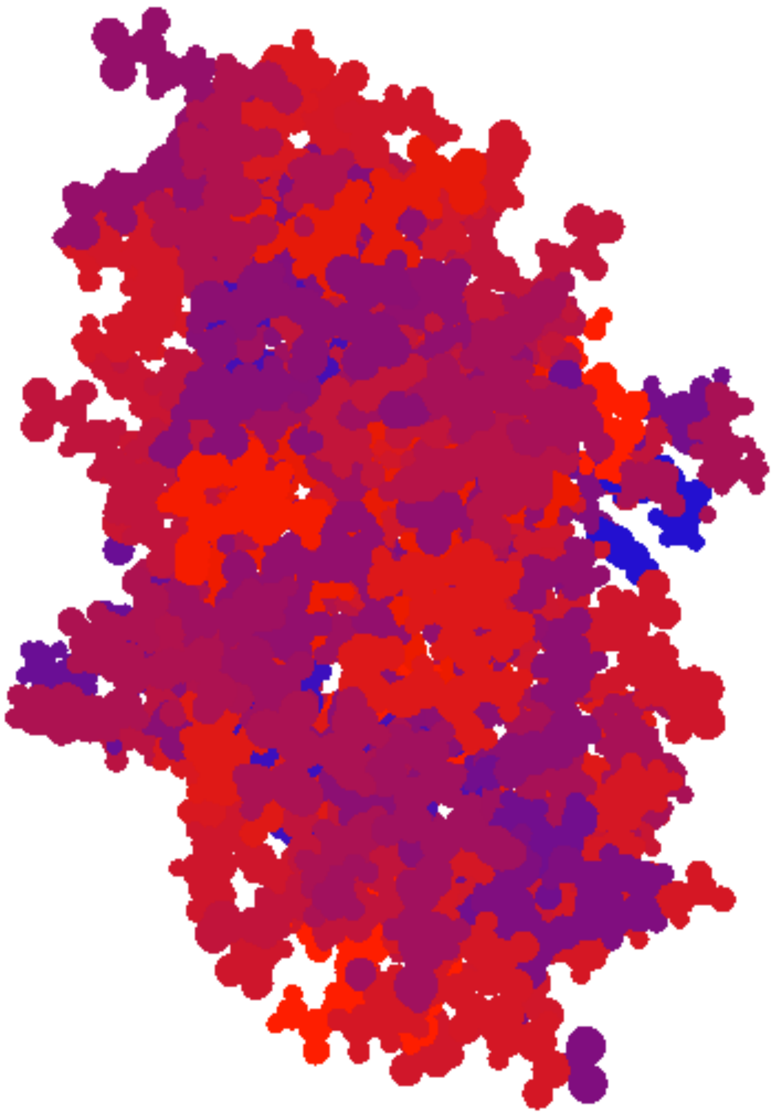


Figure 5.16: Number of Rotatable Bonds per atom. Red to Blue (maximum to minimum). SMARTS [!$(*#*)&!D1]-!@[!$(*#*)&!D1] (1A2K_1)

[6]http://www.daylight.com/dayhtml_tutorials/languages/smarts/smarts_examples.html#ROTATE

- Molar Refractivity[71] - Total polarizability of a mole of a substance

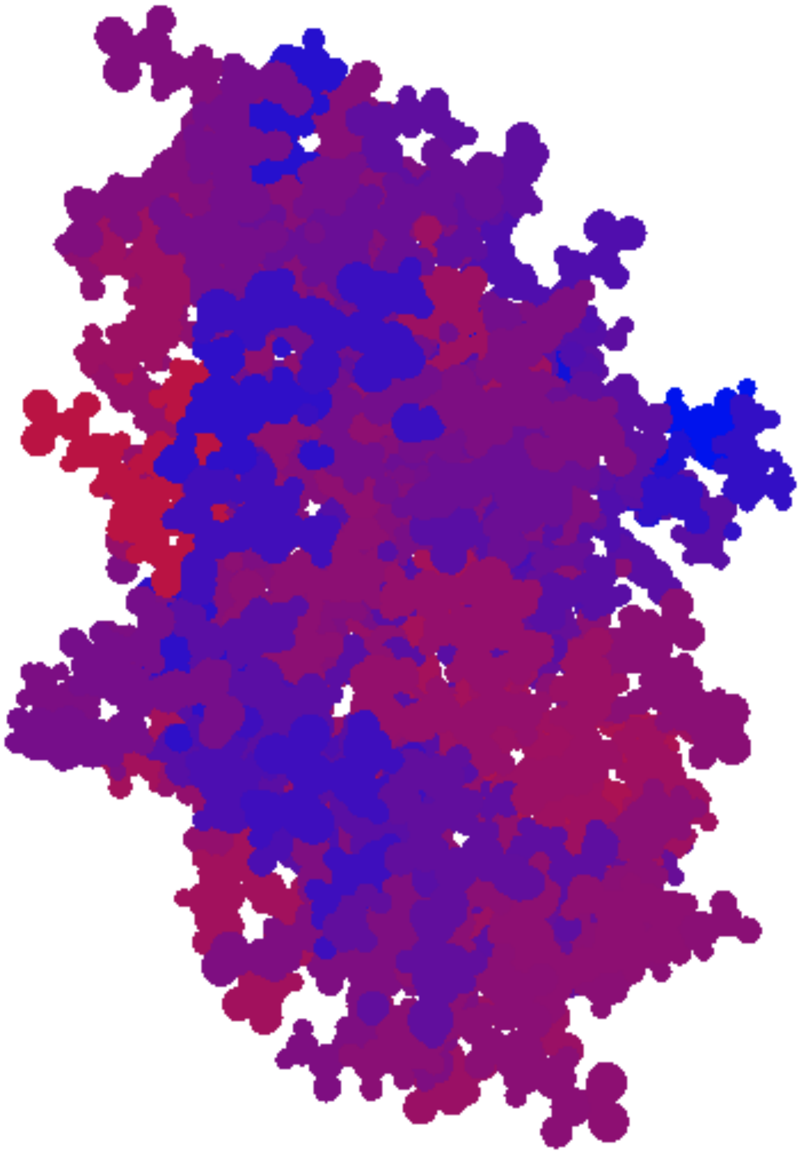


Figure 5.17: Molar Refractivity. Red to Blue (maximum to minimum). (1A2K_1)

- Total Polar Surface Area[73] - Typically useful in medicinal chemistry, polar surface area effects interactions.

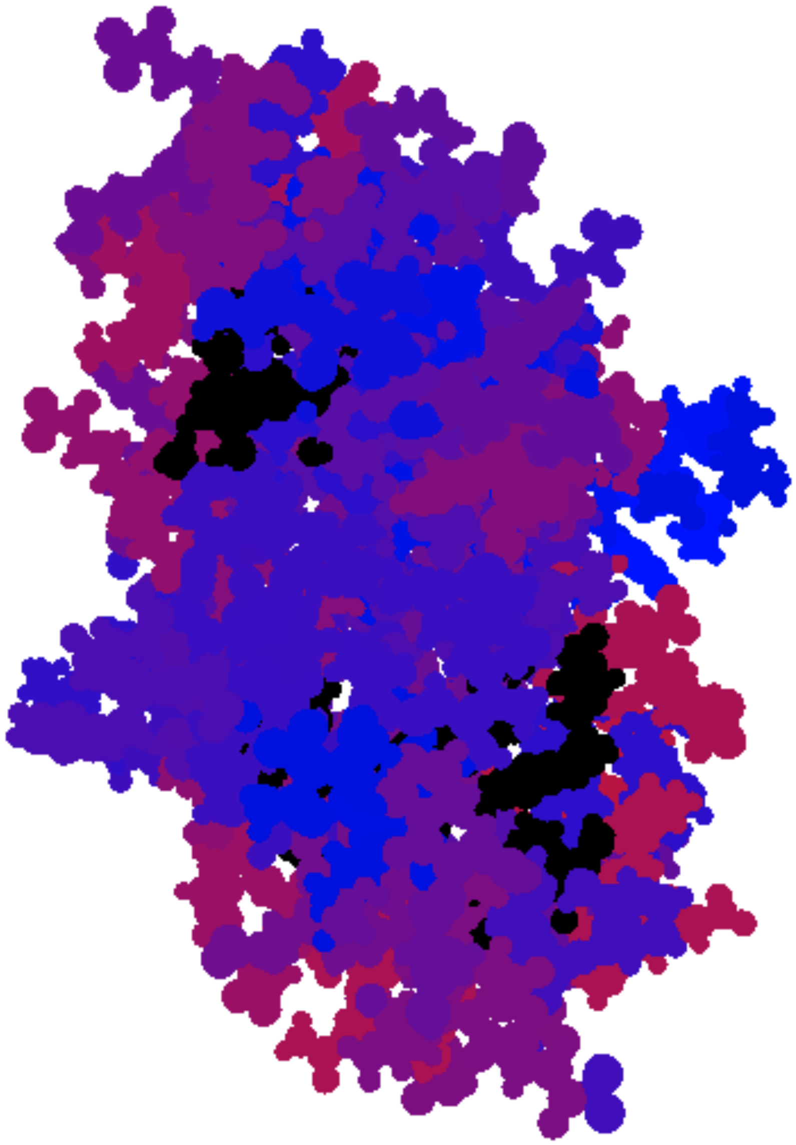


Figure 5.18: Molar Refractivity. Red to Blue (negative maximum to minimum), Black is positive. (1A2K_1)

- Complex type - This is the interaction category (4): Enzyme/Inhibitor or Enzyme/Substrate, Antibody/Antigen, Antigen/Bound Antibody, Others. [53]

These 13 real valued features plus complex type make up the entirety of the information used to learn about protein interfaces. See a complete list in Figure 5.19. It is obviously not an exhaustive set, but the goal is to have included enough to successfully localize interfaces. See the Conclusions for other possible inclusions. For the majority of the graphs reported, only the first 13 features were included due to the late addition of the 14th feature. The final results presented use all 14 features.

	Feature	**Reference**
1	Solvent Accessible Surface Area	[49, 50]
2	Solvation Potential	Kyte & Doolitle [68]
3	Solvation Potential	Sharp et al [68].
4	Hydrophobicity ($\log p$)	Wildman et al. [71]
5	Normalized Hydrophobicity ($\frac{\log p}{SASA}$)	Tribelhorn 2013
6	Standard Deviation of $\frac{\log p}{SASA}$	Tribelhorn 2013
7	Hydrogen Bond Donors 1	SMARTS
8	Hydrogen Bond Donors 2	SMARTS
9	Hydrogen Bond Acceptors 1	SMARTS
10	Hydrogen Bond Acceptors 2	SMARTS
11	Rotatable Bonds per atom	SMARTS
12	Molar Refractivity	Wildman et al. [71]
13	Total Polar Surface Area	Ertl et al. [73]
14	Interaction Category	Hwang et al. [53]

Figure 5.19: Summary of included features.

5.4 Classification

Given that this problem suffers from a clear lack of strongly predictive features, choosing a learner that is effective with weak features is a necessity. The choice of a Random Forest classifier fits this requirement well because it is effective with both weak features and missing data which lets us reduce the bias from the training set in theory.[74] Random Forests are also very fast classifiers and can even be implemented on the GPU.[75] Although SVMs are also competitive with Random Forests for many classification tasks, they are difficult to use in practice as selecting the right kernel function is an open problem. They can also be more susceptible to noise in the training dataset. Also, as the size of an interface is between 10%-30% of the surface, it can create a training bias that is difficult to overcome in both SVMs and Neural Networks. Usually this is addressed by randomly discarding data-which is undesirable given the small size of the dataset.

5.4.1 Random Forests

A Random Forest is a collection of decision trees that classify based on a majority vote. The implementation in this work creates trees depth-limited to ten[7], uses bagging[8][76], and uses Gini impurity[9][77] to determine the best split. Since we use bagging with replacement, each tree will see $1 - \frac{1}{e} \approx 63.2\%$ unique examples. One additional modification is included which is to limit the number of features that can be tested at each split (usually $\sqrt{F}$ or $\log_2 F$). This helps create a set of different trees because the algorithm is limited in the choice of split.[10] In practice this means that each split gets to choose randomly between 4 features (out of $F = 13$ possibles).

Algorithm 5.4.1: CreateRandomForest($S = \{\text{Data}\}^N$, t)

```
$f = \log_2(|S_i|) + 1$    // Num features to try at every split
for $i \leftarrow 1$ to $t$
    $S'$ = Draw $N$ samples from $S$ with replacement
    $T_i = \text{Tree}(S', f)$
RF = $\{T_1, \ldots, T_t\}$
return RF
```

[7]Used to avoid over fitting.

[8]Bootstrap aggregating (bagging) is designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It reduces variance and helps to avoid overfitting.

[9]This is a metric that determines how well a given split distinguishes between classes. Defined as $1 - \Sigma f_i^2$ where f_i is the fraction of examples of the ith class. Weighting these on the two sides of a proposed split by the number of examples on each side is minimized to choose the best split.

[10]Without this adaptation trees are more uniform because only bagging introduces randomness. Also, this helps the Random Forest utilize weaker features.

The optimal number of trees in a forest has been suggested to be no more than 128.[78] Typical rules of thumb suggest about 100 is the optimal maximum. Figure 5.20 shows performance vs. the number of trees learned. Based on these results a set of 21 trees is used for most data analysis (Figures 5.21, 6.1, 6.2). An odd number of trees are chosen to avoid tie votes.

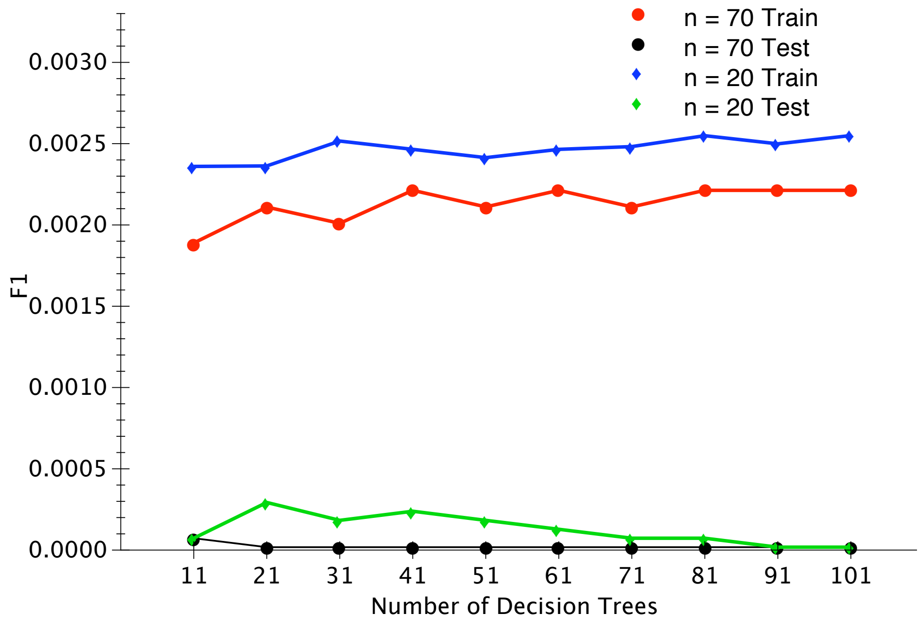


Figure 5.20: Classification performance (F_1) vs. number of trees. n is the approximate cluster size in atoms. Higher is better. Average of 10 trials. Train vs. Test selected to be opposite halves of the easy dataset.

Since the testing results are so poor, it suggests that additional effort is required to solve this problem. One method to improve the application of random forests is to adjust the feature space. Often using structural clues from the feature space can improve classification performance.

5.4.2 Codebook Learning

One method to extract and utilize structure within the feature space is to apply a codebook-style processing to that space. Other common methods include PCA, SVM kernels, and k-Nearest Neighbors. Since decision trees are limited to splits on the axes of the data, changing the feature axes can produce improvements. As there are a limited number of features, dimension reduction is not a concern; instead the goal is to acquire more information from the features.

Traditionally, codebook learning focuses on labeling bags of features, *i.e.* document classification. Since the obvious bag in this problem is a protein, the problem of labeling is uninteresting as every protein should have at least one interface. Instead of generating random interfaces (bags of adjacent regions) to fit into this paradigm, the adaptation is to use the idea of codewords to change the feature space and ignore the concept of bags and term frequencies. In light of this, what is presented here as codebook learning can also be considered a case of distance metric learning.[79, 80]

The algorithm used here starts by clustering, using kmeans++, all of the feature data which creates a set of codewords (cluster centers). The feature data is then mapped as a distance to each cluster center to create a new feature vector that is given to the learner. In this case a Random Forest with 21 trees. See Section 6.3 for more discussion of the number of trees used. Then the resultant random forest and set of codewords are used to classify future inputs. See Algorithm 5.4.2. There is one other parameter that needs to be tuned which is k, the number of codewords.

```
Algorithm 5.4.2: CreateCodebook(Learner, $S = \{\text{Data}\}^N$, $k$)

$C = \textbf{k-means++}(S, k)$
for $i \leftarrow 1$ to $N$
    $S'_i = \text{distance}_{1,\ldots,k}(S_i, C_{1,\ldots,k})$
$H = \text{Learner}(S')$
return $\langle H, C \rangle$
```

Figure 5.21 shows the success of this method depending on how many codewords, k, are chosen. Heuristically, picking points where there is a knee joint is a good choice because error in clustering decreases monotonically as you increase the number of clusters, so a location where the performance changes a larger amount indicates a good clustering. (This is similar in practice to the Gap statistic[82]). There are also unsupervised methods for picking k that consider purity, normalized mutual information, stability, or cross-validation likelihood.[81, 82, 83] In this case, a visual inspection shows $k = 15$ is the best choice because the F_1 performance peaks. As there are 13 features this shows that there is some additional structure within the features that is captured using this method. This method also fails to produce any results when the number of clusters is decreased below 13, demonstrating that each feature is strong enough on its own to be of utility.

Notice that in Figure 5.21, the red line shows an improvement using codebooks over the classification performance shown in Figure 5.20 using a Random Forest (by the F_1 metric). However, with inspiration from Zhang et al.[84], there is reason

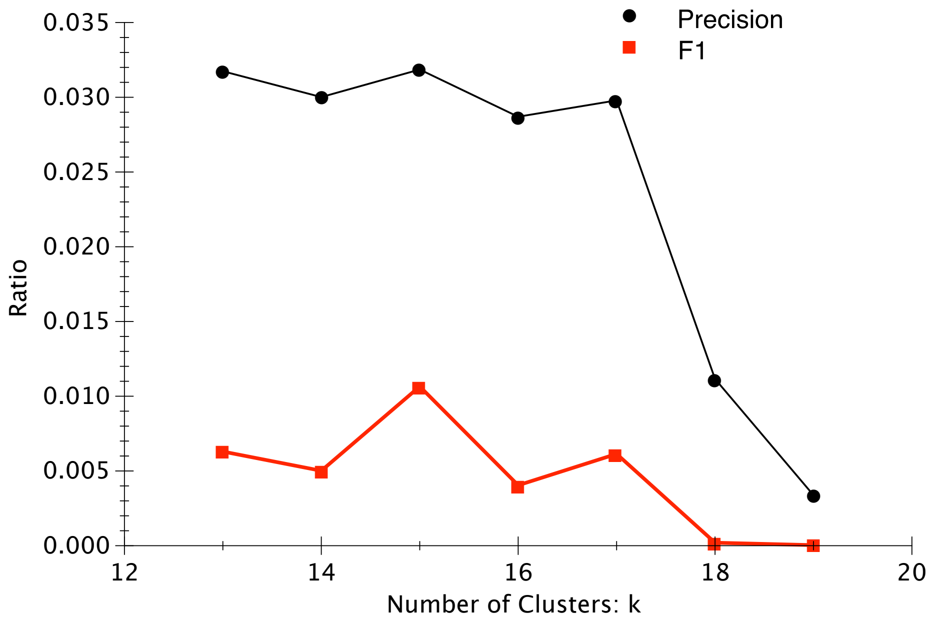


Figure 5.21: Classification performance on the test set vs. k the number of clusters. Average of 10 trials. Train vs. Test selected to be opposite halves of the easy dataset. Codebook using 21 tree Random Forest, $n = 46$ is segmented region size.

to believe that this can be improved even further. The key idea is to move the codebook creation process into a boosting cycle to dramatically decrease the error rate.

5.4.3 Boosting

The next step to improving the results in classifying regions of a protein interface is to incorporate boosting.

Adaptive Boosting, aka AdaBoost[85, 86], is a method for reducing over-fitting and creating a strong learner from a set of weak learners. Detailed in Algorithm 5.4.3, the general idea is to learn repeatedly using a set of weights. At each iteration, misclassified examples are weighted more, and correctly classified examples are weighted less. This process converges exponentially quickly. The set of weak learners trained are then combined using the α values as a weighting for their vote.

```
Algorithm 5.4.3: AdaBoost(Learner, $S = \{\text{Data}\}^N$)

initialize $W_1(i) = 1/N$, for all $i \leftarrow 1$ to $N$
for $\ell = 1, 2, \ldots, L$
    $h_\ell = \text{Learner}(S, W_\ell)$
    $\epsilon_\ell = \sum_i^N \begin{cases} W_\ell(i) & h_\ell(S_i) \neq \text{Label}(i) \\ 0 & h_\ell(S_i) = \text{Label}(i) \end{cases}$
    $\alpha_\ell = \frac{1}{2} \cdot \ln \frac{1 - \epsilon_\ell}{\epsilon_\ell}$
    $W_{\ell+1}(i) = W_\ell(i) \cdot \begin{cases} e^{\alpha_\ell} & h_\ell(S_i) \neq \text{Label}(i) \\ -e^{\alpha_\ell} & h_\ell(S_i) = \text{Label}(i) \end{cases}$
    normalize $W_{\ell+1}$
return $H = \langle \{h_1, \ldots, h_L\}, \text{WeightedVote}(\alpha_1, \ldots, \alpha_L) \rangle$
```

This implementation of AdaBoost uses weighted sampling with replacement to simulate learning with weights. In cases of very large datasets there are better methods[87], but this suffices for the relatively small dataset in this problem.

Boosting usually employs a simple weak learner like a decision stump (meaning limited depth, typically $h = 1$ or sometimes $h <= 5$). It is uncommon to boost random forests as the effect is presumably similar. However regarding Figure 5.22, when training neural networks there can be a benefit of training using momentum to speed convergence, in this case that grouping many trees causes a similar effect as choosing too large of a momentum. It is interesting to note that in this case of boosting having more than a single tree improves performance. Finally, what we can deduce from this graph is that with 110 trees we get better performance than 10 trees, but increasing to 210 or more trees causes serious over-fitting on possibly noisy data.

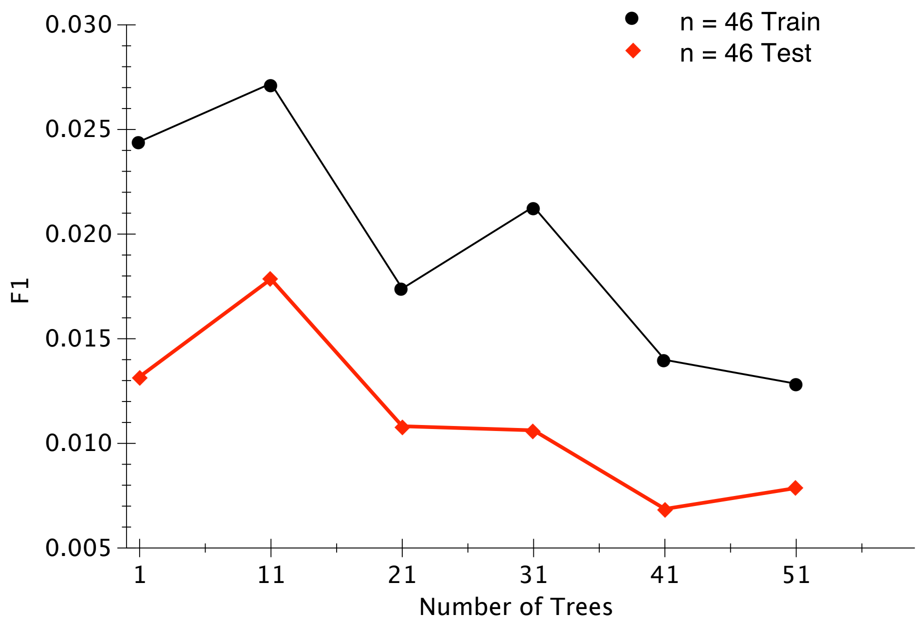


Figure 5.22: Classification performance (F_1) vs. number of trees in each round of boosting that create a boosted random forest. Average of 10 trials. Train vs. Test selected to be opposite halves of the easy dataset. $L = 10$ the number of iterations, $n = 46$ is segmented region size.

Boosting is applied at a given segmentation level just like the previous classification methods. The reason here is that performance is severely negatively impacted by trying to boost using all of the segmentation levels in one large ensemble.

The parameters to tune for boosting are the number of trees per iteration and possibly the number of boosting iterations. Boosting is also applied to codebooks. See Section 6.3 for discussion of the number of boosting iterations and boosting codebooks.

5.5 Interface Prediction

Presuming that this process has classified a large number of individual regions, the method for selecting a final region is straightforward. Each region that is predicted to be an interface adds a vote to its member atoms. Then some traditional computer vision techniques are applicable: blur, thresholding, opening, and connected components.

However, in many cases this system still favors under-prediction, so a manual determination can be made. The following algorithm is proposed for unsupervised output, but is premised on cases where there are multiple regions receiving many votes. In practice this does not occur often enough to require such a generalized system.

```
Algorithm 5.5.1: MERGE($S = \{\text{Atom}\}^N, D = \text{blur rad}, g = \text{gap rad}$)

for $i = 1, 2, \ldots, N$
    for $j = 1, 2, \ldots, N$
        if $\text{dist}(S_i, S_j) < D$
            $\sigma_i = \sigma_i + \text{Votes}(S_j)$
$T = \text{IterativeThresh}(\sigma)$
$R = \text{ConnectedComponents}(S, T, g)$
return $\langle R, \sigma \rangle$
```

Algorithm 5.5.1 starts by "blurring" votes; within euclidean radius $D = 6\text{Å}$, count all of the votes within that sphere and update σ_i (sum of neighbor's votes and own votes) for each atom i. The intuition here is similar to a blur or opening operation where we are assuming that the probability of being in an interface increases if it is near other portions of an interface and vice versa. The next step is to find a threshold for which an atom is unlikely to be in the interface despite receiving votes. This is a common method for increasing confidence of a final segmentation. To accomplish this, we use iterative thresholding which is essentially a special case of k-means with $k = 2$. Everything below the threshold T is reduced to $\sigma_i = 0$. Finally, since we know that interfaces are not atom-by-atom contiguous, we use a connected components algorithm using an acceptable gap radius, $g = 12\text{Å}$, to allow for holes or gaps within a larger interface region.

This algorithm returns all of the connected components and the total number of votes for each region. Regions are ranked by the total number of votes which should be proportional to the confidence in the final segmentation. These output regions can then be given as an input into a docking system such as HADDOCK as the key locations in order to seed a prediction. In the case of HADDOCK, these can be set as ‘mandatory’ components of an interface and it will be able to include those regions in its predictions.

Chapter 6: Analysis & Results

This chapter investigates some of the parameter choices used in the previous chapter. After discussion of how these impact the overall results, we present the results of the Multiple Scale Analysis method and show its predictions applied to CAPRI targets.

6.1 Segmentation Analysis

Figures 6.1 and 6.2 show the training performance averaged over 10 trials (using the first half of the easy dataset) of the pipeline presented for a given scale of region segmentation. We expect precision to decrease as the number of atoms per cluster increases given that the likelihood of a region being a complete subset of an interface decreases. It is this tradeoff between precision and recall that also motivates a multiple scale analysis. Notice that the precision for the Random Forest classifier is much less smooth than the boosted case. This implies that there is considerable instability based in this process. The instability is expected and suggests the use of boosting to remedy the problem.

Notice in Figure 6.1, for individual Random Forests, precision begins to increase starting around $n = 100$ until it declines precipitously. This is misleading as the increase is actually reflecting an increase of zero correct predictions which biases

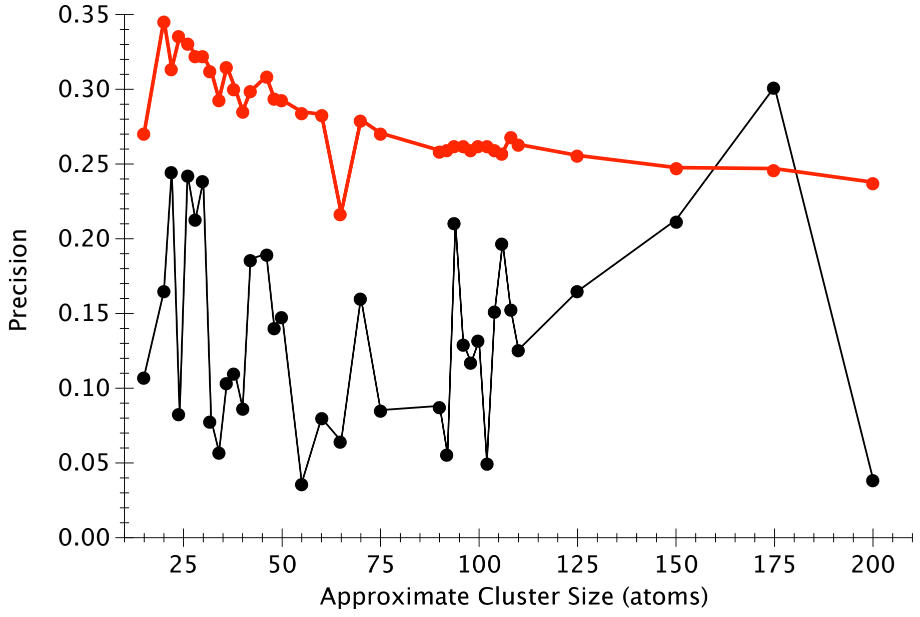


Figure 6.1: Graph showing precision versus the number of atoms in each segmented region. Average of 10 trials using the first half of the easy dataset. Black depicts a Random Forest with 21 trees, Red shows boosted Random Forests with Early Stopping.

this metric. As Figure 6.2 shows that the success peaks at $n = 94$, it suggests something is causing a "sweet-spot".

Figure 6.3 shows the minimum and maximum F_1 scores over ten trials using error bars (highlighting just the peak values of the boosted case). It is interesting to note that for different sizes the range can be drastically different. What is the difference between $n = 92$ and $n = 98$? We see that the stability of clustering changes at different sizes and that is the strongest impact between cluster sizes.

Picking points where both precision and F_1 peak (implying a high recall) are likely to combine for the best overall prediction when including multiple scales.

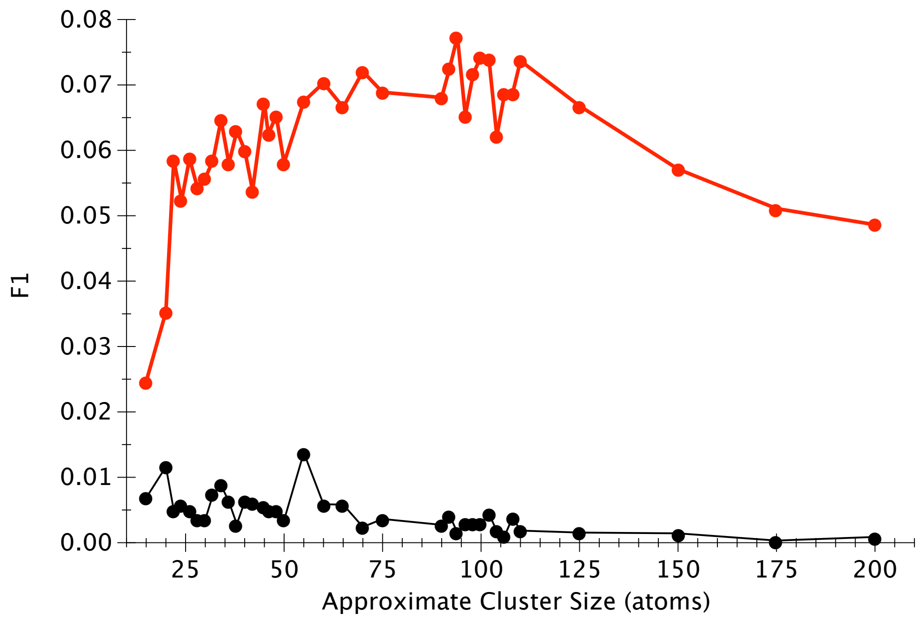


Figure 6.2: Graph showing classification success (F_1) versus the number of atoms in each segmented region. Average of 10 trials using the first half of the easy dataset. Black is a Random Forest with 21 trees, Red shows boosted Random Forests (ES).

Looking at these graphs, it is apparent that there are a number of interesting points to pick for segmentation levels. These are the ten scales at which segmentation is done with the hope that these scales are biologically relevant.

$$20, 22, 26, 34, 36, 46, 55, 60, 70, 94 \tag{6.1}$$

The utility of the small region sizes of 20, 22, 26, 34, and 36 atoms is high because they cause a segmentation size of approximately one or two amino acids. Since these are the functional units of a protein, it is not surprising that smaller

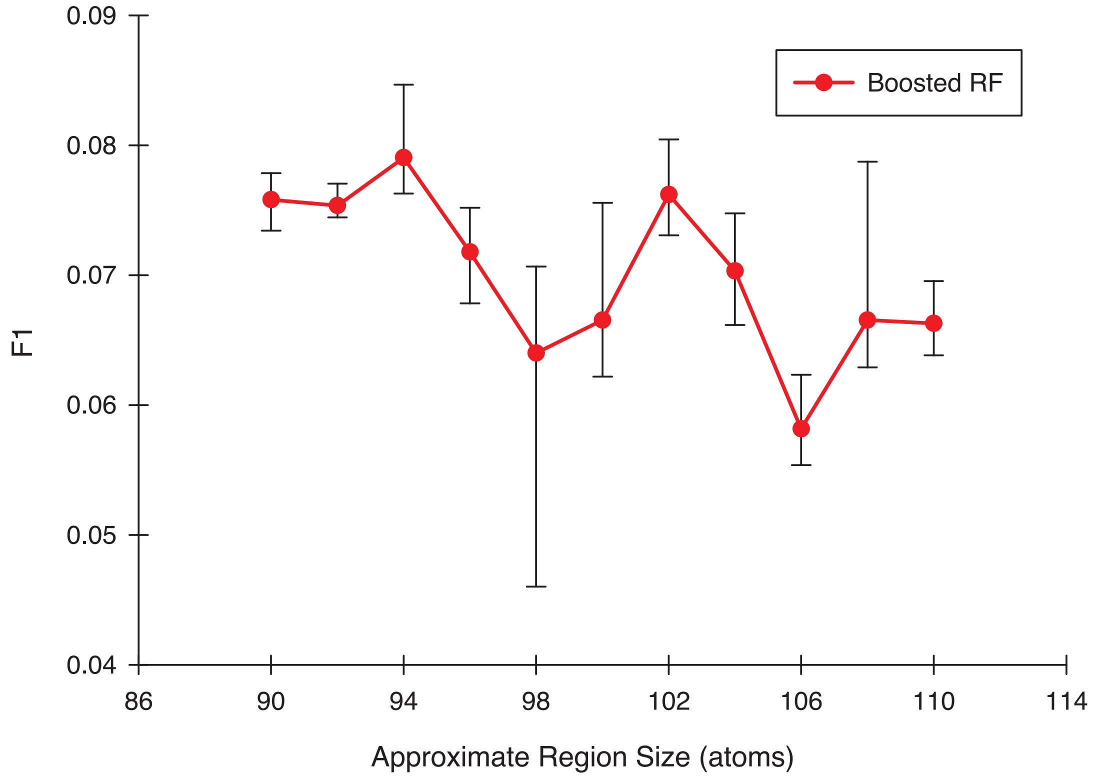


Figure 6.3: Graph showing classification success (F_1) vs the number of atoms in each segmented region. Error bars show minimum and maximum over 10 trials. Boosted Random Forests (ES).

regions do not provide enough distinguishing information while the scale of an amino acid does provide information. On the high end, a region of 94 atoms is about the size of 6 amino acids. This is small compared to the segmentation size of many other works. For example, compare to [30] which used 15 residue patches. We find that patches larger than a certain size are less effective presumably due to the loss of smaller signals. These graphs also show that collections larger than a single amino acid are useful for distinguishing an interface which validates patch-based methods.

6.1.1 Segmentation Validation

We can demonstrate the contribution of each of these segmentation levels by using a leave one out validation methodology. Figures 6.4 and 6.5 show the performance of the system with each patch size left out. In these graphs, the lower the performance without the segmentation level the more important it is because it indicates a stronger contribution to the overall system. Notice that the the strongest contributor to precision, $n = 36$, actually detracts from the F_1 metric. Given this tradeoff, we chose to emphasize precision because in this domain we are most concerned with narrowing down the location at the expense of finding the entire interface. As we know that interfaces are spatially localized regions, a higher confidence set of predictions is more desirable than a larger set of predictions as these results can be fed into a docking system to predict protein complexes.

Figure 6.4 shows that each segmentation level, when not used, reduces the system's performance. This shows that every level that we have chosen to include makes a positive contribution to our system.

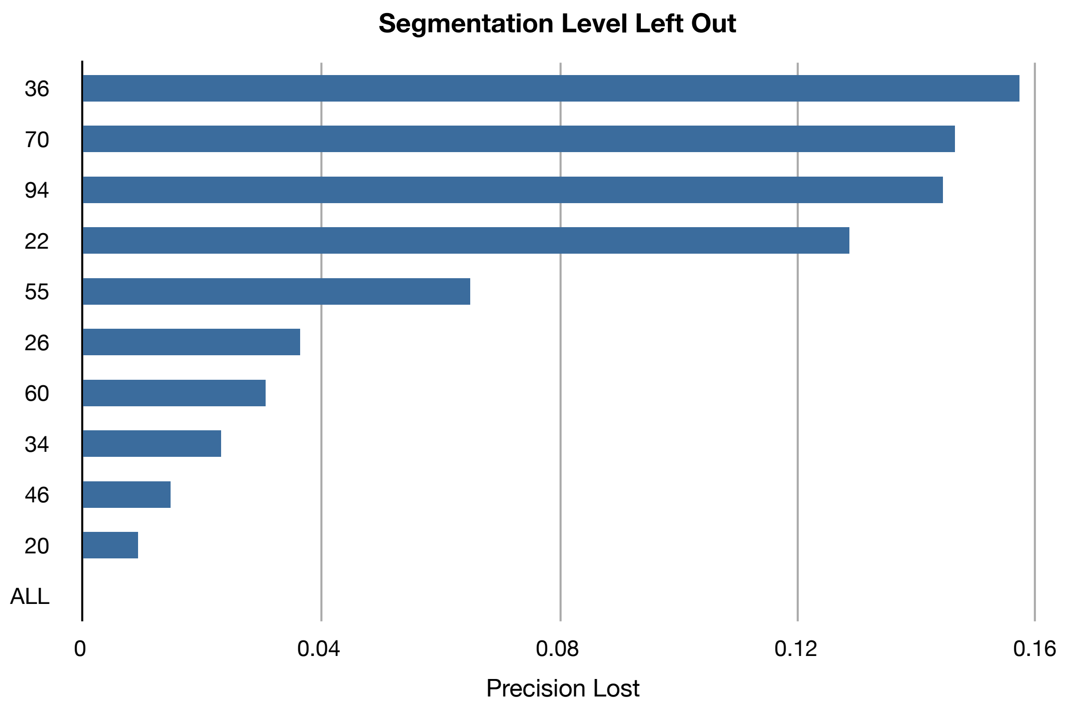


Figure 6.4: Classification performance lost (precision lost) of the pipeline using a Random Forest where the labeled segmentation level is left out. The higher values show the strongest contributors to overall performance. Average of 5 trials over first half of the easy dataset using a Random Forest with 21 trees.

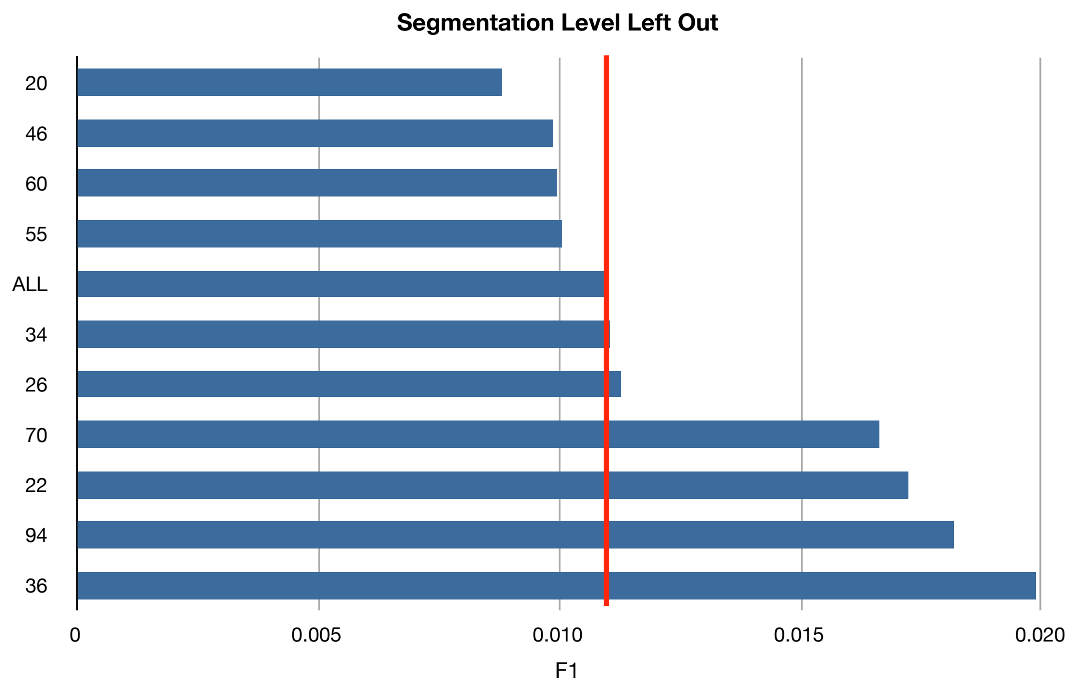


Figure 6.5: Classification performance (F_1) of the pipeline using a Random Forest where the labeled segmentation level is left out. The red line indicates the performance using all of the segmentation levels. The lower values (at the top of the graph) show the strongest contributors to overall performance. Average of 5 trials over first half of the easy dataset using a Random Forest with 21 trees.

6.2 Feature Validation

For each feature included in the system, we show its contribution to performance in Figure 6.6 using the leave one out validation method. In this figure, the lower values indicate the strongest contributors. We see that the different methods of determining H-Bond Donors or Acceptors (using SMARTS see Appendix C) can change the utility of the metric especially in the case of Acceptors. It is not surprising that H-Bond Donors are the most useful features because they represent the most reactive regions on the surface of a molecule. It is also interesting to see that the two methods for estimating the Solvation energy have largely different impacts on the performance which we would expect given the difficulty of estimating this value. Also, the inclusion of Molar Refractivity offers minimal benefits as it barely changes the system performance by either the precision or F_1 metric, so we would advise against its inclusion in future systems. This was not unexpected, as it is almost completely redundant when using TPSA, Hydrophobicity, and SASA.

This leave one out validation shows that the features we use are not independent; in that case, the performance gaps would be expected to exactly sum to the total system-wide performance. As a corollary, we can detract from the performance of the system by employing certain features. For example, using the standard deviation of hydrophobicities without first normalizing the values reduces the system performance by almost 20% (by the precision metric). So only the standard deviation of the weighted hydrophobicity[1] is included in our set of features

[1]Abbreviated as "STDDEV N log p" in Figure 6.6

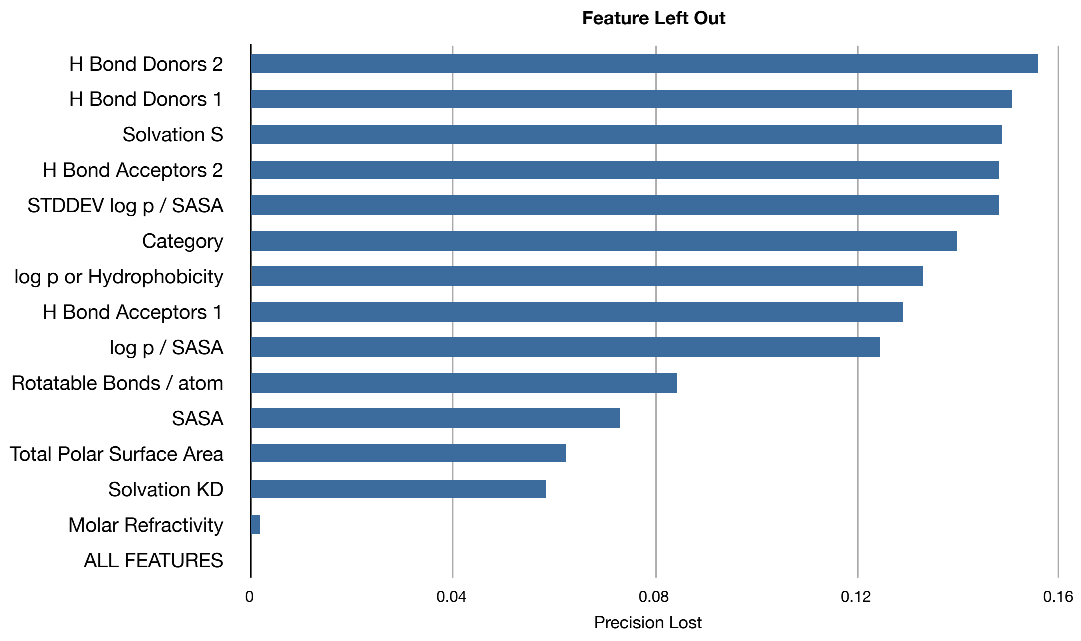


Figure 6.6: Classification performance lost (precision lost) of the pipeline using a Random Forest where the labeled feature is left out. The higher values show the strongest contributors. Average of 5 trials over first half of the easy dataset using a Random Forest with 21 trees.

which is the first use of this feature in this problem and we have shown that it is a very strong feature. The other feature that we have introduced, Normalized Hydrophobicity (log p / SASA), also demonstrates considerable utility in this problem.

6.3 Boosting Analysis

Figures 6.7 and 6.8 show the performance of boosting on Random Forests. Early stopping is a method where boosting stops after an individual member of the

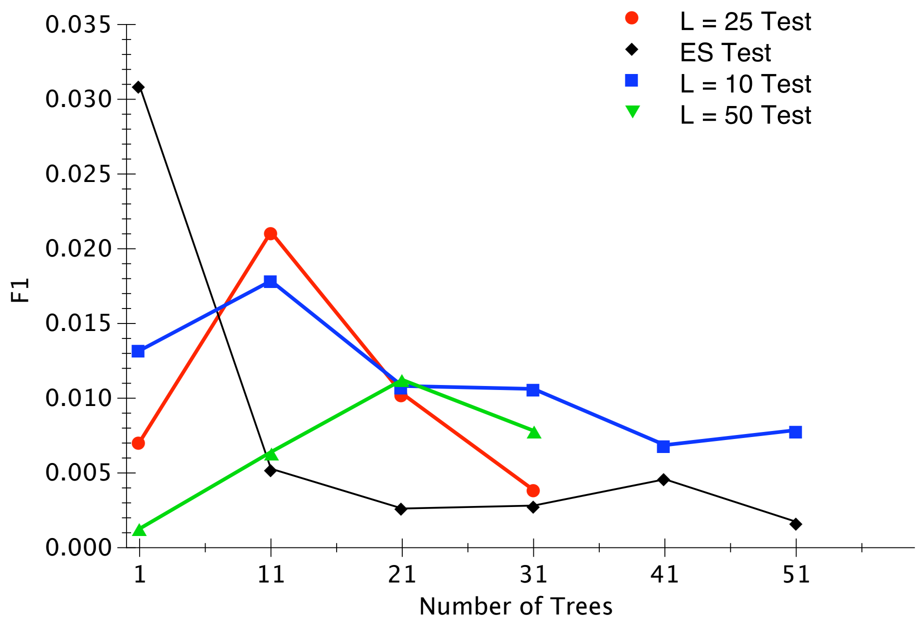


Figure 6.7: Classification performance (F_1) vs. number of trees in the boosted random forest. Average of 10 trials. Train vs. Test selected to be opposite halves of the easy dataset. L is the number of iterations, ES means early stopping, $n = 46$ is segmented region size.

ensemble correctly classifies all of the training examples. In many cases this is with the number of iterations being $L = 2$ or $L = 3$ and in virtually every case $L \leq 5$. The testing and training sets were chosen arbitrarily to be the first half and second half of the easy dataset (listed in Appendix A). Each data point represents an average of 10 trials on the same clustering (region size $n = 46$), so only the training of the Random Forests varies.

It should be clear that the choice of the number of trees, t, for the optimal boosted Random Forest is a difficult one. Is it better to choose for a higher (F_1) number of correct classifications even if that increases our incorrect predictions as

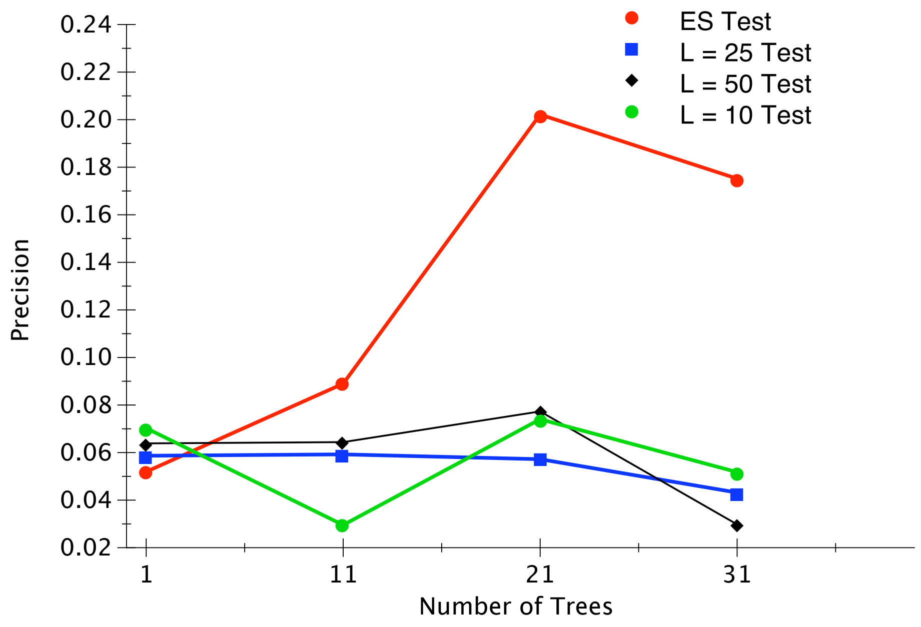


Figure 6.8: Classification performance (precision) vs. number of trees in each forest in the boosted random forest. Average of 10 trials. Train vs. Test selected to be opposite halves of the easy dataset. L is the number of iterations, ES means early stopping, $n = 46$ is segmented region size.

well? Or is it better to focus on small sets of highly correct predictions (Precision) with the tradeoff of under-prediction? Seeing that in Figure 6.7 all of the boosting rounds except early stopping perform almost identically at $t = 21$, which shows that this choice promises a certain stability. We also know that increasing the rounds of boosting should slowly reduce testing error, so choosing a point where the most rounds has the lowest test error is also likely to be indicative of a good choice. However, it is also difficult to ignore the early stopping case which performs very well with $t = 1$ which is very similar to traditional boosting.

Boosting is also applied to codebook learning inspired by the work presented in

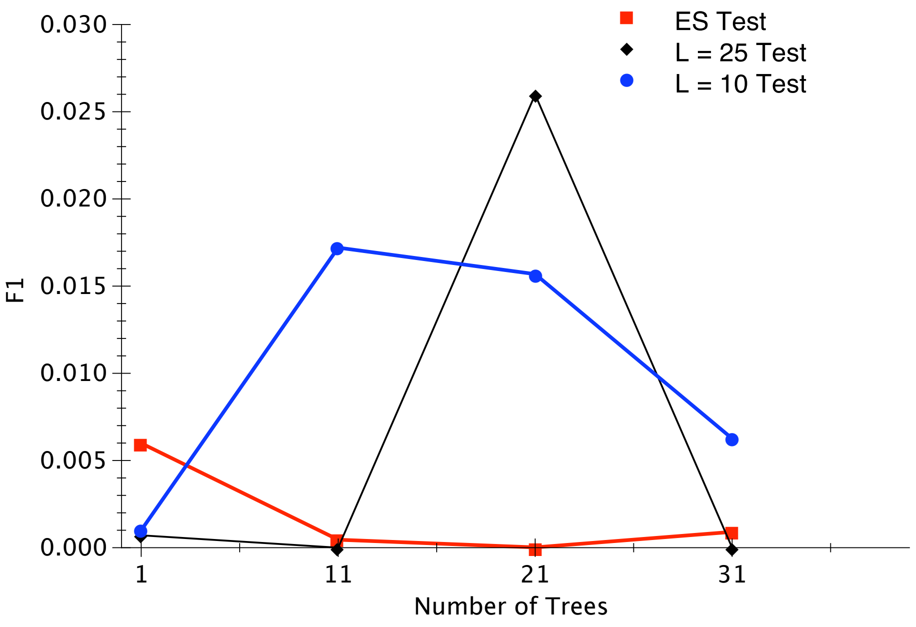


Figure 6.9: Classification performance vs. number of trees in the boosted Codebook. Average of 10 trials. Train vs. Test selected to be opposite halves of the easy dataset. L is the number of iterations, ES means early stopping, $n = 46$ is segmented region size. Codewords: $k = 15$.

Zhang et al.[84]. However, with larger number of boosting rounds training becomes impossible because the weights focus on the hardest cases which eventually become a set of non-separable classes. See Figure 6.9 to see the effectiveness of boosting. In this case, choosing the number of trees to be $t = 21$ is a far more obvious choice. It is also worth noting how the peak value in this result compares to Figure 6.7 where it handily beats every data point except the early stopping case. This validates the use of this more complex learning paradigm as it manages to pull out additional structure from the training data. The sensitivity to the number of boosting rounds is due to the small size of the dataset.

6.4 Illustrated Results

Every training run results in highly variable results. This implies that the data doesn't support a strong classification which is well supported in the literature. Given the instability of classification and the need to search parameters, the results included here allow for a sense of what the system is capable of accomplishing. What we show here is that in many cases the correct result is mixed in with false predictions. This shows that we can reduce the search space towards the interface location in many cases (Same point as shown in Figure 2.3 comparing the best possible result at each stage). Since the segmentation is different every time we run the software these values for Precision and F_1 are just one pass using the resultant classifier. It will perform differently between runs (although more similarly than re-training the classifier).

	RF $t = 11$	RFE ES, $t = 21$	CB $t = 21$, $k = 15$	CBE $t = 21$, ES, $k = 16$
Precision	0.0697	0.0502	0.0464	0.0502
F_1	0.0107	0.0222	0.0040	0.0248
Localized Interface Easy	94/232	113/232	44/232	26/232
Localized Interface Med/Hard	48/102	37/102	38/102	21/102

Figure 6.10: Results trained on all data. RF = Random Forest, RFE = Random Forest Ensemble, CB = Codebook, CBE = Codebook Ensemble. Used t trees, k codewords, and ES means early stopping during boosting.

Figure 6.10 shows the overall results for four different training runs (one for each type of classifier). This shows the tradeoff between methods: do you pick a higher precision or a higher F_1? What we see is that the un-boosted classifiers perform considerably lower when looking at the F_1 score. This is because the boosting cases made more predictions (due to Early Stopping). If you disable early stopping then the boosted methods drastically under predict, in many cases making zero predictions given enough boosting rounds. This more clearly illustrates that the features are not a strong enough set to solve this problem.

Let's take the idea of precision one step farther. Since it is the ratio of true positives to false positives, a value of 1 would mean every prediction correctly finds part of the protein interface. Since we achieve in the best cases of about 0.05-0.1, **that means that we are correct in localizing the interface in about 5-10% of our predictions.** This is a good way to compare these results to other methods. Recall the state of the art in protein docking (the other half of this problem) is currently upper bounded by a success rate of about 10%. We've achieved results that are on par with other methods. Also other works (such as Assi et al.[4]) have focused on identifying critical sub-regions of protein interfaces (aka "hot-spots"), which is what we have also achieved in our predictions. We can narrow the possible interactive sites down from thousands of areas to a few regions with definite certainty in the cases where this system makes a prediction.

Figure 6.11 shows the value of multiple scales. It depicts the difference between all predictions and a threshold (atom by atom) requiring at least 2 votes. In most cases this threshold increases our confidence in a specific considerably. We did not measure this directly, in part because not enough regions received multiple votes from any given classifier. Note that the random forest was the classifier that generally produced the most predictions.

Figure 6.12 shows an example segmentation by the four classification methods discussed in the previous chapter. It is worth noting that on the right are the boosted variants show a much more conservative segmentation. We observe this in every testing run which illustrates the boosting benefit of avoiding over-fitting. It is especially clear in the boosted random forest case where the difference is that multiple regions get 4/10 votes with the random forest, but only a single region gets a single vote in the boosted case. Interestingly, the variation between a single codebook and the boosted version tends to be minimal. This result is better than most because it doesn't identify any region that doesn't include at least one interface atom and regardless of the classification method it localized the interface. This level of performance should be attainable for most structures. The next chapter discusses the extensions that could lead to this goal.

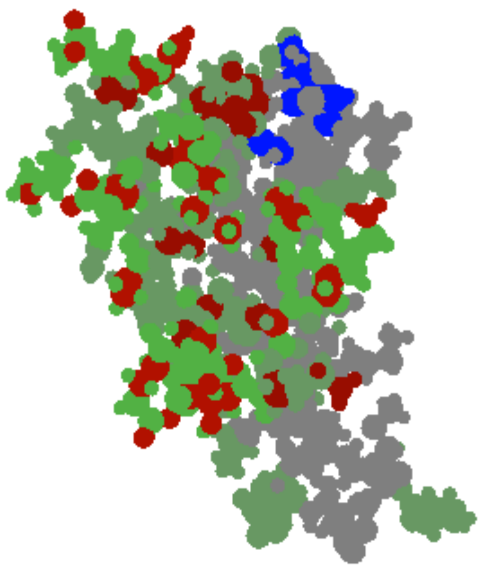
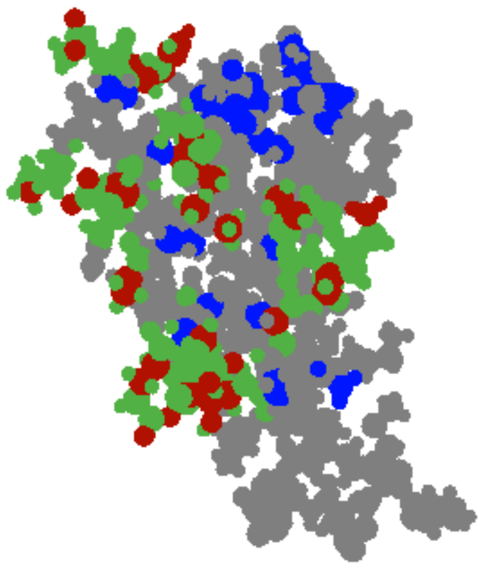



Figure 6.11: Results using a boosted random forest (detailed in Figure 6.10). Left shows all predictions with at least one vote and the right shows atoms that received at least two votes. This demonstrates the value of the multiple scale analysis. Green means over prediction, red correct, and blue under prediction. (2VDB_2)

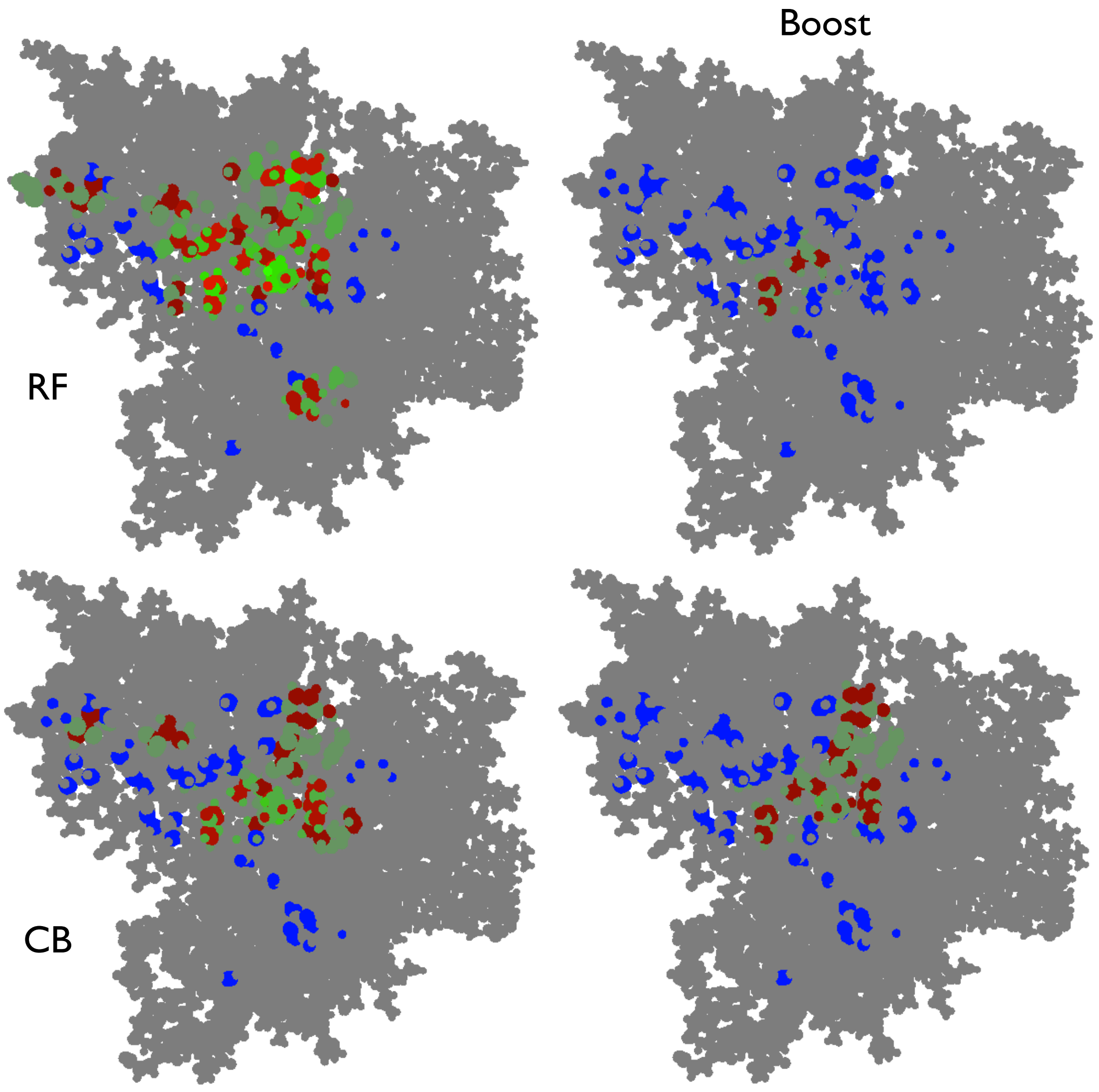


Figure 6.12: From left to right, top to bottom the different segmentations by a Random Forest, Boosted Random Forest, Codebook, Boosted Codebooks. The predicted interface is colored in red (correct prediction) and green (over-prediction), under-predicted interface is in blue. Brighter colored red or green reflects additional votes (higher confidence). This example used 21 trees, $k = 16$ codewords, and $L = 10$ boosting rounds, trained on all data. (3CPH_2)

6.5 Multiple Scale Analysis System Summary

We can illustrate the performance of this system using a receiver operating characteristic (ROC) curve, by tuning the decision threshold on the classifiers. To demonstrate this, we use the basic Random Forest where the typical decision threshold is a majority vote of the constituent trees. In the case of a 21 tree Random Forest this means at least 11 trees must agree on a given classification. However, it is possible to tune this threshold to increase the confidence of a prediction. Often the operating point, the threshold in this case, is chosen to be when recall equals precision. This assumes that the cost of a False Positive is equal to the cost of a False Negative, which is not true in this case. For the interface prediction problem, it is much more important to avoid False Positives as they increase the error in the docking prediction problem when used as initial guesses.

Figure 6.14 shows the ROC curve for the Random Forest classifier and highlights the chosen operating point. We chose the majority vote threshold as it offers the highest number of correct predictions while maintaining more than a 5-fold margin on the cost of a False Positive to a False Negative. Intuitively, given that the size of an interface is 10-30% of the surface atoms[2], if we predict all of the interface atoms and the atoms nearby we have selected very roughly about 20% of the atoms incorrectly (False Positive) and would have achieved 100% True Positives which is a simple 5-fold ratio. Since we are classifying regions with the Random Forest, this is a possible outcome. However, it hides some of the

[2]Counted by residue. This includes atoms outside of the 6Å interaction zone that are part of a residue that contains an interacting atom, or true interface atom.

complexity regarding the number of segmentation levels. This assumes that any positive prediction for a region automatically designates its component atoms as "interface." This need not be the case.

Figure 6.15 shows the impact of increasing the number of times an atom must be predicted as "interface" (up to 10 times given 10 segmentation levels). We illustrate that using a higher threshold in the merge stage allows for higher confidence predictions at the expense of the number of predictions. For this threshold we chose 1 or more votes as the operating point because the number of predictions becomes too sparse, typically only one interface in the whole dataset is identified when increasing this threshold.

# Complexes	167
Average molecule size	3826 atoms
2-Class problem	Binary "Interface" (per atom)
True Positive %	3.1382 %
False Positive %	0.6075 %
RF Operating Point	0.5 (majority vote)

Operating Point	**# of atoms**
True Positive	9,876
False Positive	63,184
True Negative	12,366,167
False Negative	340,222

Figure 6.13: Multiple Scale Analysis System Summary. Sample results using a Random Forest with 21 trees.

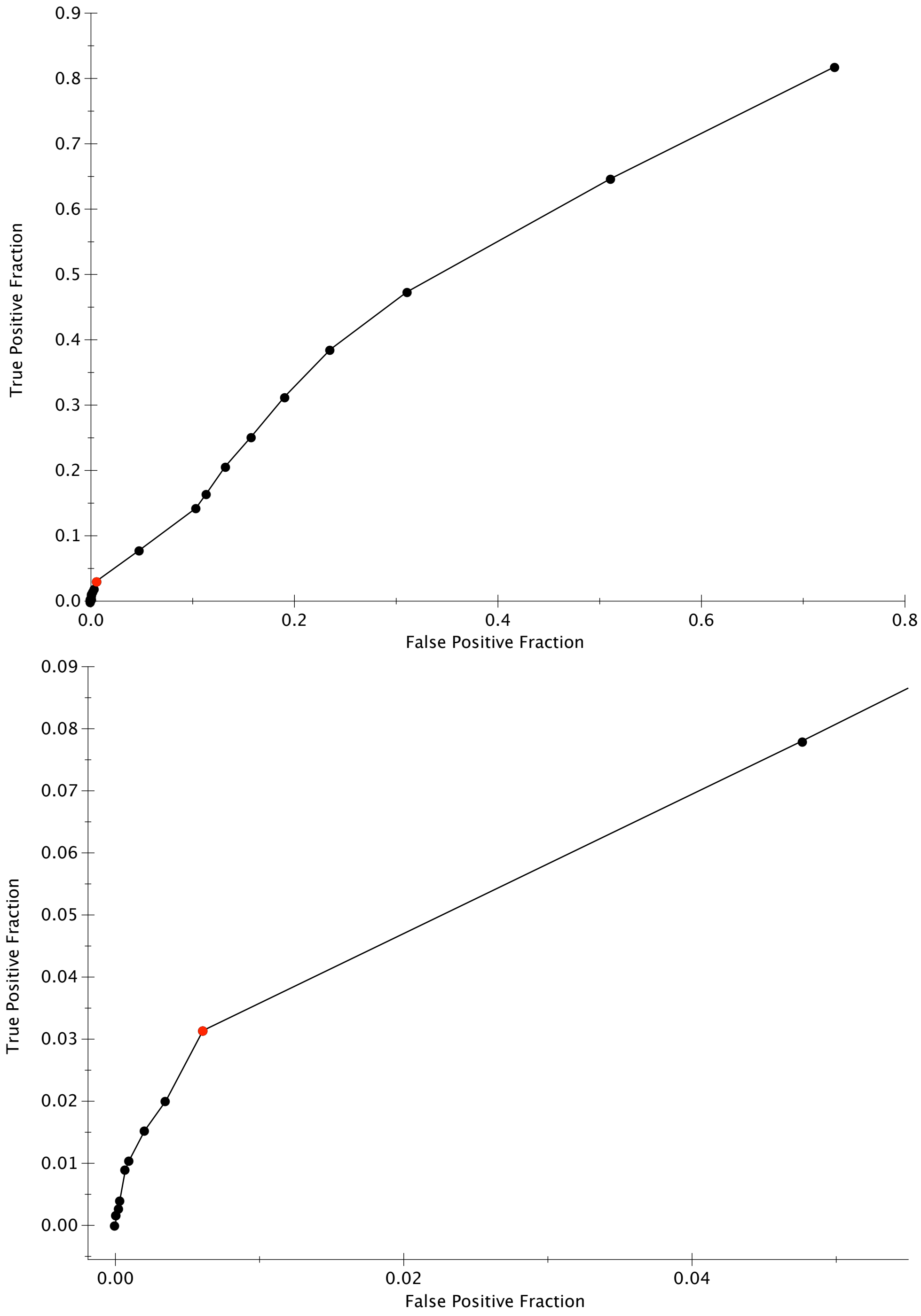


Figure 6.14: ROC curve showing the tradeoff for different thresholds in the Random Forest for making a prediction. The red dot indicates the chosen operating point for a Random Forest ($t = 21$). This point is the majority vote threshold. The lower curve is a zoom of the top curve. One run using first half of easy dataset.

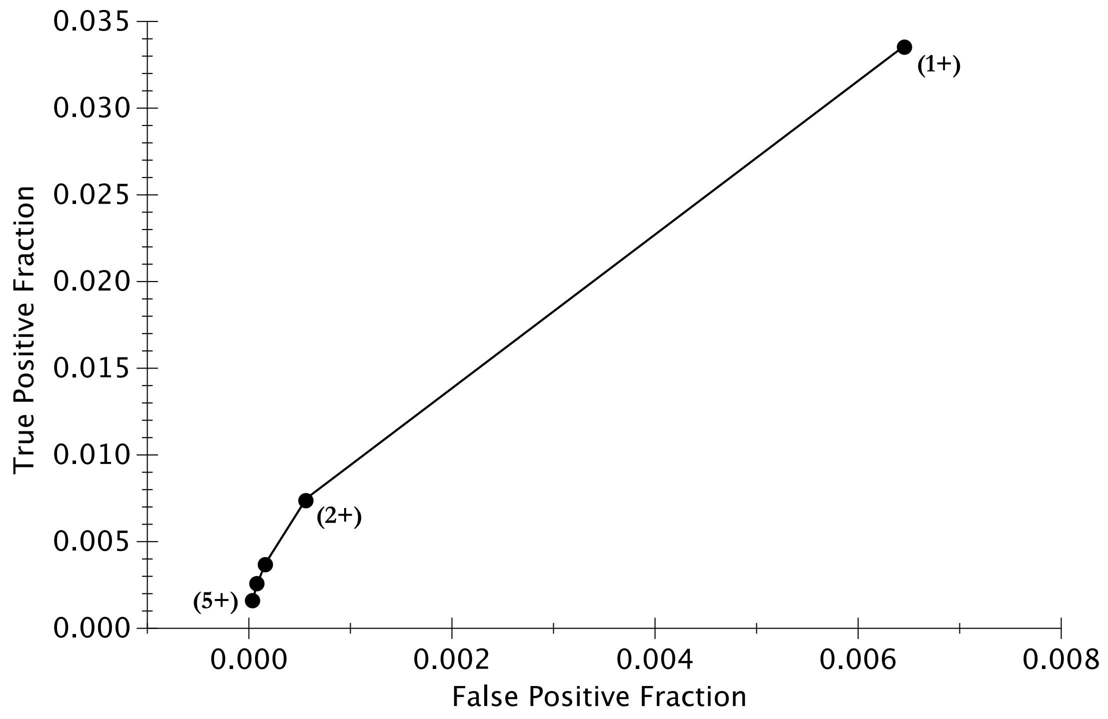


Figure 6.15: ROC curve showing the tradeoff for different thresholds for how many segmentation levels an atom needs to be identified in to be predicted as an interface atom. As there are 10 segmentation levels, a single atom can get up to 10 votes. Average of 5 trials using first half of easy dataset. The operating point is 1+ votes which produces less than 1% false positives.

6.6 CAPRI Targets

6.6.1 28th Round

Target 59 is a complex between the LSm domain of the fission yeast Edc3 and ribosomal protein Rps28b. The target is a gift of Dr. Remco Sprangers (Max Planck Institute for Developmental Biology, Tubingen, Germany). The coordinates used are for NMR model 1 of entries 4A53 (Edc3) and 1NE3 (a homolog of Rps28b). In collaboration with Dr. Victor Hsu and Hari Caushik[3] we submitted complexes predicted using our predictions with HADDOCK.

Using only the first 13 features (no complex interaction type), we found results illustrated in Figures 6.17 & 6.18. Of the 10 predictions we submitted we used different key sites between runs based on the different predictions from our classifiers. The reason for this manual action is that in very few cases did the predictors agree. The only case that had multiple agreements was on the tip of Rps28b, see Figure 6.16.

We also submitted predictions for four protein-peptide interactions (not shown). Results from this event are expected later in 2013.

[3]Oregon State Univerisity

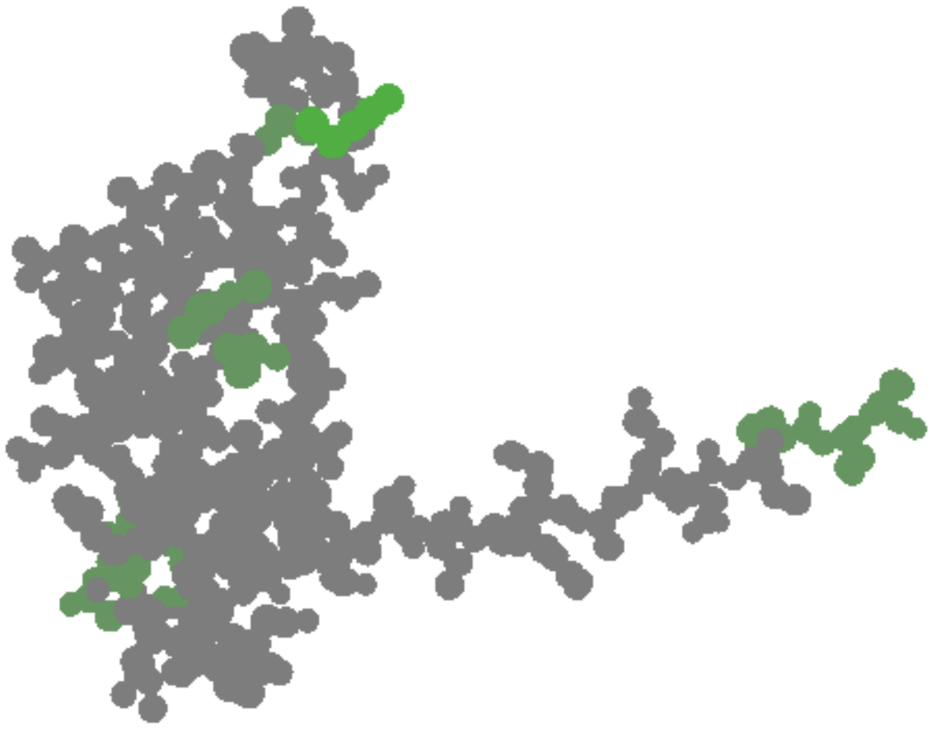
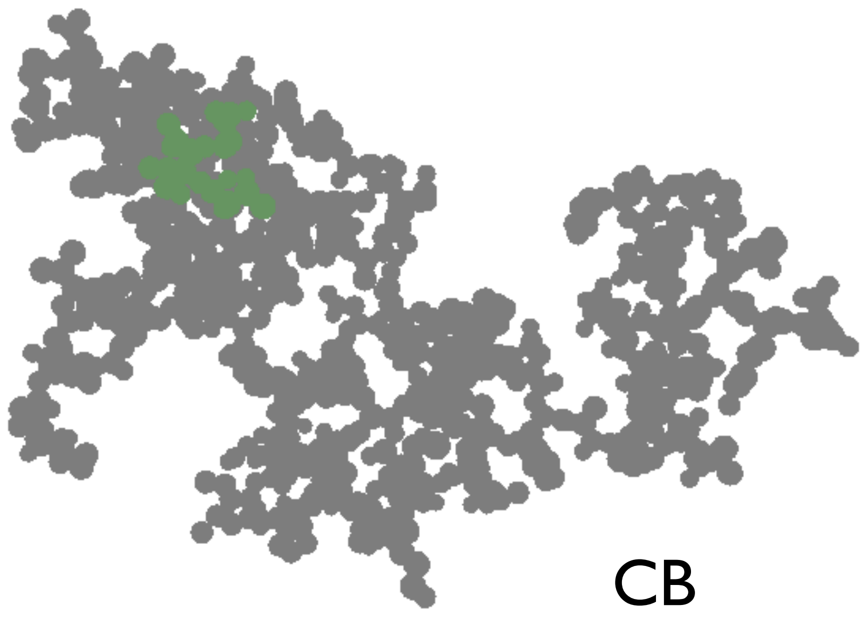


Figure 6.16: CAPRI Round 28 - Target 59 (Rps28b). Green shows predicted interface regions (not merged) using a Random Forest with 11 trees, trained on the easy dataset.



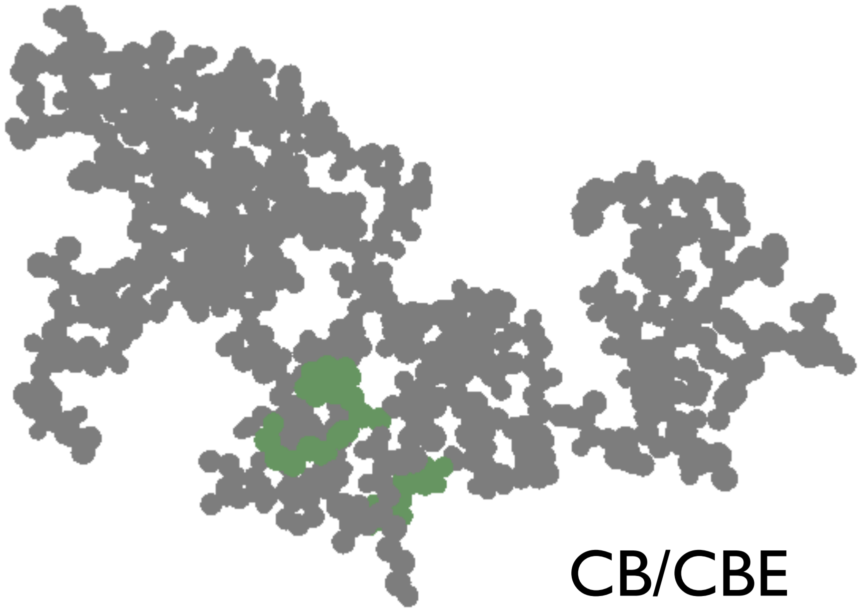
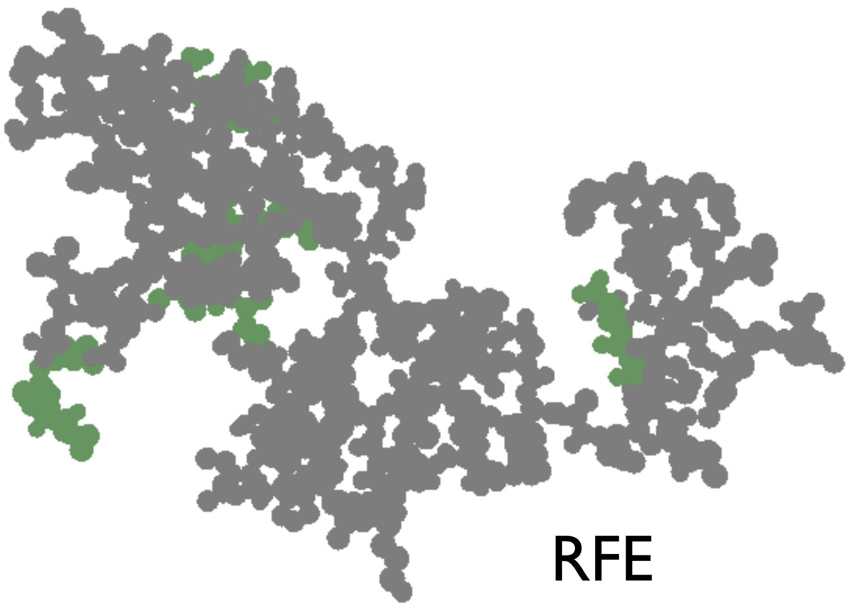


Figure 6.17: CAPRI Round 28 - Target 59 (Edc3). Green shows predicted interface regions (not merged). Top: Codebook prediction using 11 trees, $k = 15$, trained on the easy dataset. Bottom: Codebook and Boosted Codebook prediction using 21 trees, $k = 15$, trained on all data.



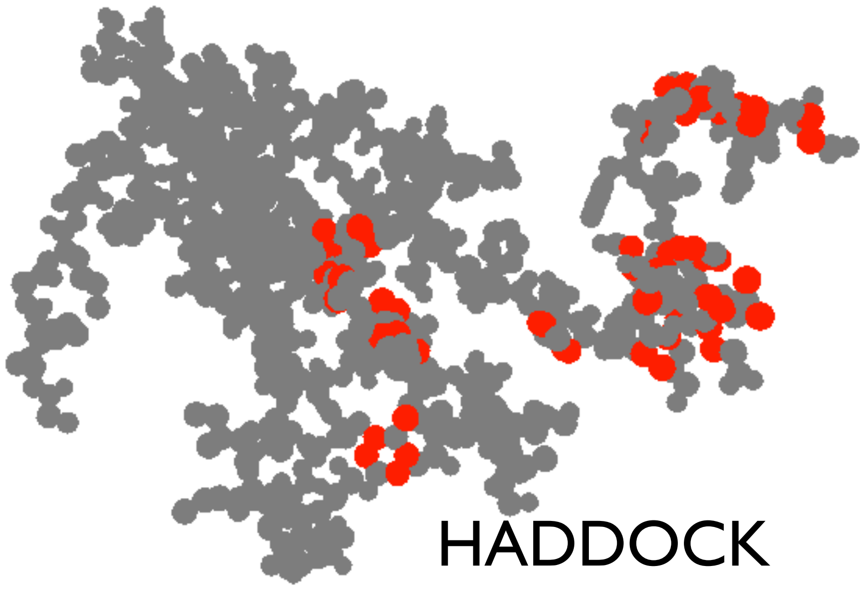


Figure 6.18: CAPRI Round 28 - Target 59 (Edc3). Green shows predicted interface regions (not merged). Red shows the final interface and structure as predicted by HADDOCK using a subset of predictions from this figure and previous figure. Top: Boosted Random Forest with 21 trees, $L = 10$ boost iterations, trained on all data. Bottom: Top ranked interface using HADDOCK score (notice conformational change is predicted because the shape has changed from the original model).

6.6.2 Previous Target

Here is the most recent CAPRI Target (#58) with published results (2013): PliG /SalG lysozyme complex with SAXS data aka 4G9S[7]. Figure 6.19 shows the best results we can achieve using the same classifiers (same training run) we used for our predictions on Target 59. We think that it is interesting that the prediction on the right side circles the interface. This suggests that something around the site might be contributing to the interface. In the future it might be worth investigating if using more interface "edge" information might increase our ability to find an interface.

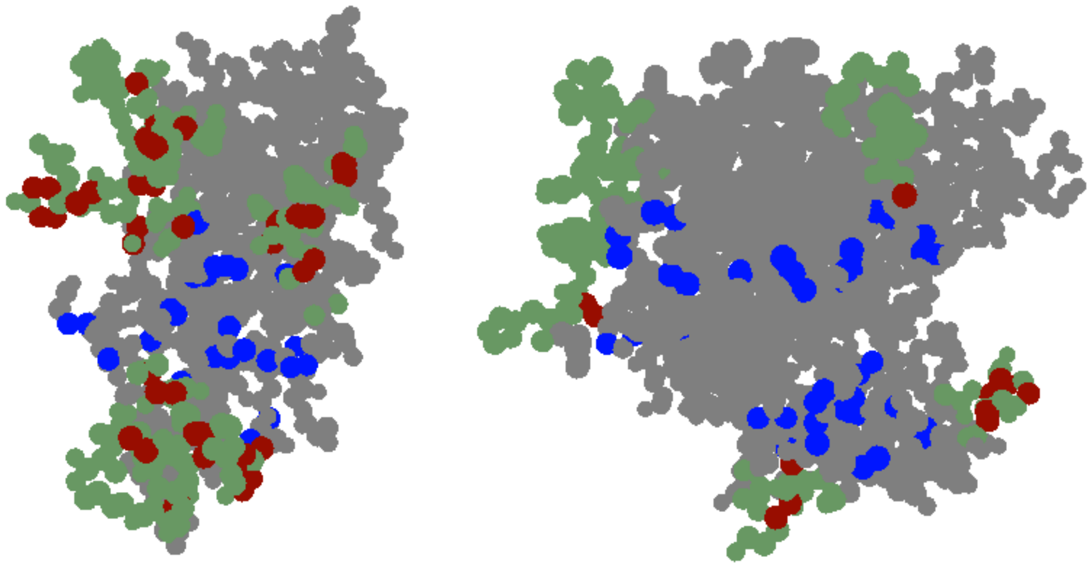


Figure 6.19: CAPRI Target 58[7]. Results on left using boosted random forest. Results on right using codebook. (4G9S) Classifiers trained on all data, same as used in Figures 6.17 & 6.18

Chapter 7: Conclusion

Predicting protein interactions is an extremely challenging problem. The facet addressed in this work, identifying protein interfaces, is a key step to solving a biological process that nature ensures functions perfectly. Since these mechanisms function at every moment in every living organism, it should be a computationally solvable problem. This presents great opportunity, and in this field of study, the desire for success is so great that both demonstrations of methods that do not work and of those that do are considered of particular interest to the scientific community. The work presented in the previous chapters demonstrate entirely new results that extend the knowledge in this field.

7.1 Contributions

Overall this paper constitutes a meaningful set of contributions to both the disciplines of Computer Science and Biophysics. This research combines multiple techniques that have not been investigated in concert within this problem space previously.

We have presented two novel methods for addressing the protein interface prediction problem. In Chapter 4 we demonstrated that atomic level information merged with residue information was not enough to produce results beyond the state of the art using a single class SVM. This suggested a number of extensions that were pursued to improve our results. The multiple scale analysis, articulated in Chapter 5, makes significant contributions to the field and produces competitive results. This approach also led to an entry to the 28th Round of the CAPRI event. This work presents a first for each of the following items.

1. New segmentation method that eliminates *a priori* bias. Prior methods depended on residue-level segmentations which is shown to be misleading in solving this problem. We also highlight the range of potentially optimal patch sizes.
2. Dynamic property computations. For each segmented region we determine properties including some that have not been employed previously. We also include new meta features.
3. First use of codebooks for this problem. We show that codebooks are able to find additional structure in the data.
4. First use of boosting in this problem. We show that boosting improves the confidence of localizing an interface at the expense of under-prediction.
5. Employed and contrasted a variety of ensemble learning methods, including random forests which display positive characteristics for this problem.
6. Submitted protein docking predictions to the CAPRI event by combining results from this work with the HADDOCK software.
7. Created an integrated and modular prediction system that matches other top performers on this problem by identifying critical regions within interfaces.

8. Presented (in the next section) a set of actionable extensions to the multiple scale analysis pipeline that could improve results.

Prior work typically formulates this problem as a single question of "What is the docking interface for a complex?" This paper argued that the problem is actually a two-tier question. We have demonstrated that by focusing on the interface prediction tier that we can elucidate critical portions of the interface in many cases which would improve results in docking simulation systems, such as HADDOCK. Our system produces results that are different than any existing system which emphasizes our contribution to the field as our work generates orthogonal predictions that can improve consensus driven docking (recall the state of the art driven by a consensus in Figure 2.3). Ultimately, our system has the following advantages over other top performers:

- Speed - Near-realtime (under 30 seconds) compared to the hours for most other systems. Also can benefit from newly solved complexes with a simple retraining.
- Modularity & extensibility (see next section)
- No requirements for expert-definitions since machine learning optimizes parameters automatically, unlike other "hot-spot" predictors (i.e.[4]).
- Can be leveraged to improve existing restraint-based docking simulators by predicting the critical regions of an interface since there is a very low false positive rate.
- No *a priori* bias on interface location.

Given the extraordinary difficulty of this problem it is important to continue to pursue novel ideas. While this work explored every avenue to achieve the best results, there are a few small adaptations that could be explored in the future thanks to the modularity of this system. In that vein, the next section details some possible avenues for future research.

7.2 Extensions to Multi-Scale Analysis

The first place to improve this work would be to incorporate additional data by harvesting homodimer/multimer complexes. If we assume that all types of docking provide information there are possibly hundreds of additional data points that could be leveraged to improve this process. Some researchers do include this data to increase their available training data.

7.2.1 Segmentation

We found that the segmentation using k-means++ is generally stable, so repetitive over-segmentation, a desirable action, is not feasible. A more principled segmentation could drastically improve results. Spectral clustering could create more geometrically relevant clusterings. Also, by looking at the set of atoms in an interface it resembles a set of polka-dots which suggests that creating regions that are not contiguous might be able to better align to the physical reality. This could be achieved by computing a larger form of surface accessibility: instead of using the radius of a water molecule, using a sphere 2-3 times larger would allow for the emphasis of the primary surface actors.

7.2.2 Features

Additional features are required to increase this system's ability to identify interfaces. One area for including more features would be to include more versions of the hydrophobicity calculations. These are empirically derived values and there are many different versions published by researchers. Secondly, although conformational changes can make initial shape less relevant to the final complex, shape is a part of this problem that should be considered. There are many ways to define shape; typically a FFT is applied (i.e. over a spherical volume). In the 2D case this is a texture descriptor for image analysis. This suggests that a similar effort could be a useful feature in this problem. Other shape feature that could be adapted include aspect, waviness, etc. Finally as discussed previously, a gradient meta feature for the current features could be incorporated. Ideas like a Histogram of Gradients (HOG) are commonly used in image processing and are an obvious place for adaptation to this problem.

Outside of the pipeline to predict interfaces without knowing the opposite actor, inter-molecular propensities is an area that has been highly studied and could be incorporated in a second pass. In the same spirit there is also some existing work on intra-molecular propensities that could be specifically included (as opposed to hoping that the learning system identifies these correlations).

7.2.3 Classification

Classification methods allow for almost limitless improvements. Below are some "low hanging fruit" for variations worth investigating and ideas that showed up when reading related literature.

Random Forest: Since we found a large instability in the output of the random forests[1], either the set of features was too weak or that the randomness inserted was not well utilized. One way to address this problem would be to use a different split criteria, the gini coefficient used can be foiled in cases where there are too few options. Since this method used too few features, the split choice could have been too limiting. One alternate heuristic for splitting is called ReliefF that improves the detection of correlational dependencies between features.[88, 89] (Recall we claim these exist based on the results from the application of Codebooks.)

Codebooks: The first place to improve this classifier would be to employ spectral clustering or Gaussian Mixture Modeling instead of kmeans++. Both of these methods can result in better approximations of the feature space, which is difficult to manually determine.

The version of codebook learning used in this work is actually a simplification of the typical approach. A modification of the pipeline to add an agglomerative phase to create "bags" would transform this method into something more reminiscent of traditional codebook learning. In this extension, you could then implement features

[1]Larger than is typical or desirable for this learner.

as tf-idf (term-frequency inverse-document-frequency, see Zhang et al.) instead of as a distance metric.

Boosting: AdaBoost, the version of boosting implemented in this work, is the standard "first pass" method for incorporating boosting. However, there are many other versions that have been shown to improve upon this version, among them gradient boosting can perform significantly better.

Additionally, boosting could be applied to SVMs. Boosting SVMs is also a successful paradigm as SVMs are some of the best classifiers. We could incorporate the single class SVM into boosting and perhaps see more stable results than the random forest classifier.

7.3 So long, and thanks for all the fish

This document presented a set of contributions that extend the field of protein docking prediction. Our method is naturally collaborative, as it focuses on providing key insight, the interface location, to existing models and methods (including HADDOCK). In creating an orthogonal methodology, the consensus in protein docking is improved by this work.

Bibliography

[1] J. Janin, "Assessing predictions of protein-protein interaction: The CAPRI experiment," *Protein Science*, 2005. http://www.ebi.ac.uk/msd-srv/capri

[2] S. J. de Vries and A. M. Bonvin, "How proteins get in touch: Interface prediction in the study of bio-molecular complexes," *Current Protein and Peptide Science*, 2008.

[3] S. Vajda and D. Kozakov, "Convergence and combination of methods in protein-protein docking," *Current Opinion in Structural Biology*, 2009.

[4] S. A. Assi, T. Tanaka, T. H. Rabbitts, and N. Fernandez-Fuentes, "PCRPi: Presaging critical residues in protein interfaces, a new computational tool to chart hot spots in protein interfaces," *Nucleic Acids Research*, 2010.

[5] R. Mendez, R. Leplae, L. De Maria, and S. Wodak, "Assessment of blind predictions of protein-protein interactions: current status of docking methods." *Proteins*, 2003.

[6] S. J. de Vries and A. M. J. J. Bonvin, "CPORT: A consensus interface predictor and its performance in prediction-driven docking with HADDOCK." *PLoS ONE 6(3): e17695.*, 2011.

[7] S. Leysen, L. Vanderkelen, S. Weeks, C. Michiels, and S. Strelkov, "Structural basis of bacterial defense against g-type lysozyme-based innate immunity," *Cellular and Molecular Life Sciences*, 2013.

[8] R. Unger and J. Moult, "Finding the lowest free energy conformation of a protein is an np-hard problem: Proof and implications," *Bulletin of Mathematical Biology*, 1993.

[9] B. Sadjad and Z. Zsoldos, "Toward a robust search method for the protein-drug docking problem." *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2011.

[10] K. Dukka Bahadur, E. Tomita, J. Suzuki, and T. Akutsu, "Protein side-chain packing problem: a maximum edge-weight clique algorithmic approach." *Journal of Bioinformatics and Computational Biology*, 2005.

[11] K. Wüthrich, *NMR of Proteins and Nucleic Acids.* Wiley, New York, 1986.

[12] J. C. Kendrew, G. Bodo, H. M. Dintzis, R. G. Parrish, and H. Wyckoff, "A three-dimensional model of the myoglobin molecule obtained by x-ray analysis." *Nature*, 1958.

[13] Y. Zhang and Z. Xu, "Atomic radii of noble gas elements in condensed phases," *American Mineralogist*, 1995.

[14] O. Lichtarge, H. Bourne, and F. Cohen, "An evolutionary trace method defines binding surfaces common to protein families." *Journal of Molecular Biology*, 1996.

[15] I. Mihalek, I. Res, and O. Lichtarge, "A family of evolution-entropy hybrid methods for ranking protein residues by importance," *Journal of Molecular Biology*, 2004.

[16] S. Madabushi, H. Yao, M. Marsh, D. Kristensen, A. Philippi, M. Sowa, and O. Lichtarge, "Structural clusters of evolutionary trace residues are statistically significant and common in proteins." *Journal of Molecular Biology*, 2002.

[17] D. Caffrey, S. Somaroo, J. Hughes, J. Mintseris, and E. Huang, "Are protein-protein interfaces more conserved in sequence than the rest of the protein surface?" *Protein Science*, 2004.

[18] I. Kufareva, L. Budagyan, E. Raush, M. Totrov, and R. Abagyan, "Pier: protein interface recognition for structural proteomics." *Proteins*, 2007.

[19] Y. Ofran and B. Rost, "Proteinprotein interaction hotspots carved into sequences," *Journal of Molecular Biology*, 2003.

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

[21] K. Wiehe, B. Pierce, W. Tong, H. Hwang, J. Mintseris, and Z. Weng, "The performance of ZDOCK and ZRANK in rounds 6-11 of CAPRI." *Proteins*, 2007.

[22] D. Kozakov, R. Brenke, S. Comeau, and S. Vajda, "PIPER: an FFT-based protein docking program with pairwise potentials," *Proteins*, 2006.

[23] D. Schneidman-Duhovny, R. Nussinov, and H. Woldson, "Automatic prediction of protein interactions with large scale motion," *Proteins*, 2007.

[24] N. Kowalsman and M. Eisenstein, "Inherent limitations in protein-protein docking procedures," *Bioinformatics*, 2007.

[25] J. Gray, S. Moughon, C. Wang, O. Schueler-Furman, B. Kuhlman, and D. Baker, "Protein-protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations," *J Mol Biol*, 2003.

[26] J. Fernandex-Recia, M. Totrov, and R. Abagyan, "ISM-DISCO docking by global energy optimization with fully flexible side-chains." *Proteins*, 2003.

[27] C. Dominguez, R. Boelens, and A. M. Bonvin, "HADDOCK: a protein-protein docking approach based on biochemical and/or biophysical information." *Journal of the American Chemical Society*, 2003.

[28] S. J. de Vries, A. D. J. van Dijk, M. Krzeminski, M. van Dijk, A. Thureau, V. Hsu, T. Wassenaar, and A. M. J. J. Bonvin, "Haddock versus haddock: New features and performance of haddock2.0 on the capri targets," *Proteins*, 2007.

[29] I. Res, I. Mihalek, and O. Lichtarge, "A structure and evolution-guided monte carlo sequence selection strategy for multiple alignment-based analysis of proteins." *Bioinformatics*, 2005.

[30] A. J. Bordner and R. Abagyan, "Statistical analysis and prediction of protein-protein interfaces," *PROTEINS: Structure, Function, and Bioinformatics*, 2005.

[31] H. Neuvirth, R. Raz, and G. Schreiber, "Promate: a structure based prediction program to identify the location of protein-protein binding sites." *Journal of Molecular Biology*, 2004.

[32] H. Chen and H. Zhou, "Prediction of interface residues in protein-protein complexes by a consensus neural network method: test against nmr data." *Proteins*, 2005.

[33] J. Bradford, C. Needham, A. Bulpitt, and D. Westhead, "Insights into protein-protein interfaces using a bayesian network prediction method." *Journal of Molecular Biology*, 2006.

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

[35] F. Pettit, E. Bare, A. Tsai, and J. Bowie, "Hotpatch: a statistical approach to finding biologically relevant features on protein surfaces." *Journal of Molecular Biology*, 2007.

[36] A. Porollo and J. Meller, "Prediction?based fingerprints of proteinprotein interactions," *Proteins*, 2007.

[37] Y. Ofran and B. Rost, "Proteinprotein interaction hotspots carved into sequences." *Bioinformatics*, 2007.

[38] H. Zhou and Y. Shan, "Prediction of protein interaction sites from sequence profile and residue neighbor list." *Proteins*, 2001.

[39] B. Wang, P. Chen, D.-S. Huang, J. jing Li, T.-M. Lok, and M. R. Lyu, "Predicting protein interaction sites from residue spatial sequence profile and evolution rate," *FEBS Letters 580*, 2005.

[40] B. Wang, H. Wong, and D. Huang, "Inferring protein-protein interacting sites using residue conservation and evolutionary information." *Protein Peptide Letters*, 2006.

[41] A. Koike and T. Takagi, "Prediction of proteinprotein interaction sites using support vector machines." *Protein Eng. Des. Sel.*, 2004.

[42] J. Chung, W. Wang, and P. Bourne, "Exploiting sequence and structure homologs to identify protein-protein binding sites." *Proteins*, 2006.

[43] Q. Dong, X. Wang, L. Lin, and Y. Guan, "Exploiting residue-level and profile-level interface propensities for usage in binding sites prediction of proteins." *BMC Bioinformatics*, 2007.

[44] O. Martin and D. Schomburg, "Efficient comprehensive scoring of docked protein complexes using probabilistic support vector machines," *Proteins*, 2008.

[45] J. R. Bradford, C. J. Needham, A. J. Bulpitt, and D. R. Westhead, "Insights into protein-protein interfaces using a bayesian network prediction method," *Journal of Molecular Biology*, 2006.

[46] M.-H. Li, L. Lin, X.-L. Wang, and T. Liu, "Protein-protein interaction site prediction based on conditional random fields," *Bioinformatics*, 2007.

[47] S. J. de Vries and A. M. J. J. Bonvin, "Intramolecular surface contacts contain information about protein-protein interface regions," *Bioinformatics*, 2006.

[48] C. J. van Rijsbergen, *Information Retrieval.* Butterworth-Heinemann, 1979. http://www.dcs.gla.ac.uk/Keith/Preface.html

[49] S. Hubbard and J. Thornton, "NACCESS," 1993, [Computer Program]. Department of Biochemistry and Molecular Biology, University College London.

[50] B. Lee and F. Richards, "The interpretation of protein structures: estimation of static accesibility." *Journal of Molecular Biology*, 1971.

[51] A. Brunger, P. Adams, G. Clore, W. DeLano, P. Gros, R. Grosse-Kunstleve, J. Jiang, J. Kuszewski, M. Nilges, N. Pannu, R. Read, L. Rice, T. Simonson, and G. Warren, "Crystallography & NMR system (CNS): A new software suite for macromolecular structure determination." *Acta Crystallographia Section D: Biological Crystallography*, 1998.

[52] H. Hwang, B. Pierce, J. Mintseris, J. Janin, and Z. Weng, "Protein-protein docking benchmark version 3.0," *Proteins*, 2008.

[53] H. Hwang, T. Vreven, J. Janin, and Z. Weng, "Protein-protein docking benchmark version 4.0," *Proteins*, 2010.

[54] F. Armougom, S. Moretti, V. Keduas, and C. Notredame, "The iRMSD: a local measure of sequence alignment accuracy using structural information." *Bioinformatics*, 2006.

[55] K. Pearson, "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine*, 1901.

[56] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[57] G. Trinquier and Y. Sanejouand, "Which effective property of amino acids is best preserved by the genetic code?" *Protein Engineering*, 1998.

[58] "OpenCV," [Computer Program]. http://opencv.org/

[59] "Scalable library for eigenvalue problem computations," [Computer Program]. http://www.grycap.upv.es/slepc/

[60] "Portable, extensible toolkit for scientific computation," [Computer Program]. http://www.mcs.anl.gov/petsc/

[61] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, 2011.

[62] N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, and G. R. Hutchison, "Open babel: An open chemical toolbox," *Journal of Cheminformatics*, 2011.

[63] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, 1982.

[64] D. Titterington, A. Smith, and U. Makov, *Statistical Analysis of Finite Mixture Distributions.* Wiley, 1985.

[65] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems (NIPS).* MIT Press, 2001, pp. 849–856.

[66] A. Radhakrishna, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk., "SLIC Superpixels compared to state-of-the-art superpixel methods." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[67] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," *SODA '07 Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007.

[68] L. Wesson and D. Eisenberg, "Atomic solvation parameters applied to molecular dynamics of proteins in solution," *Protein Science*, 1992.

[69] T. Masuda, T. Jikihara, K. Nakamura, A. Kimura, T. Takagi, and H. Fujiwara, "Introduction of solvent-accessible surface area in the calculation of the hydrophobicity parameter log p from an atomistic approach," *Journal of Pharmaceutical Sciences*, 1997.

[70] A. Sarkar and G. E. Kellogg, "Hydrophobicity - shake flasks, protein folding and drug," *Curr Top Med Chem*, 2010.

[71] S. A. Wildman and G. M. Crippen, "Prediction of physicochemical parameters by atomic contributions," *Journal of Chemical Information and Modeling*, vol. 39, no. 5, pp. 868–873, 1999.

[72] V. Gillet, P. Willett, and J. Bradshaw, "Identification of biological activity profiles using substructural analysis and genetic algorithms." *Journal of Chemical Information and Computer Sciences*, 1998.

[73] P. Ertl, B. Rohde, and P. Selzer, "Fast calculation of molecular polar surface area as a sum of fragment-based contributions and its application to the prediction of drug transport properties," *Journal of Medical Chemistry*, 2000.

[74] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.

[75] T. Sharp, "Implementing decision trees and forests on a gpu," *Computer Vision - ECCV 2008*, 2008.

[76] L. Breiman, "Bagging predictors." *Machine Learning*, vol. 26(2), 123-140, 1996.

[77] L. Breiman, J. Friedman, C. J. Stone, and R. Olshen, *Classification and Regression Trees.* Belmont, California: Wadsworth Inc., 1984.

[78] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, "How many trees in a random forest?" in *Machine Learning and Data Mining in Pattern Recognition*, 2012.

[79] L. Yang, "Distance metric learning: A comprehensive survey," Michigan State University, Tech. Rep., 2006.

[80] E. Xing, A. Y. Ng, M. Jordan, and S. Russell, "Distance metric learning, with application to clustering with side-information," *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[81] P. Smyth, "Model selection for probabilistic clustering using cross-validated likelihood," University of California, Irvine, Tech. Rep., 1998.

[82] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B*, 2001.

[83] E. Levine and E. Domany, "Resampling method for unsupervised estimation of cluster validity," *Neural Computation*, 2001.

[84] W. Zhang, A. Surve, X. Fern, and T. Dietterich, "Learning non-redundant codebooks for classifying complex objects," *International Conference on Machine Learning (ICML)*, 2009.

[85] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," 1995.

[86] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," *Machine Learning: Proceedings of the Thirteenth International Conference*, pp. 148–156, 1996.

[87] Z. Kalal, J. Matas, and K. Mikolajczyk, "Weighted sampling for large-scale boosting," *British Machine Vision Conference*, 2008.

[88] M. Robnik-Sikonja and I. Kononenko, "Theoretical and empirical analysis of ReliefF and RReliefF," *Machine Learning Journal*, 2003.

[89] M. Robnik-Sikonja, "Improving random forests," *Machine Learning, ECML*, 2004.

APPENDICES

Appendix A: List of Protein Data

Benchmark v4

1A2K, 1AHW, 1AK4, 1AKJ, 1AVX, 1AY7, 1AZS, 1B6C, 1BJ1, 1BUH, 1BVK, 1BVN, 1CGI, 1CLV, 1D6R, 1DFJ, 1DQJ, 1E6E, 1E6J, 1E96, 1EAW, 1EFN, 1EWY, 1EZU, 1F34, 1F51, 1FC2, 1FCC, 1FFW, 1FLE, 1FQJ, 1FSK, 1GHQ, 1GL1, 1GLA, 1GPW, 1GXD, 1H9D, 1HCF, 1HE1, 1HIA, 1I4D, 1I9R, 1IQD, 1J2J, 1JPS, 1JTG, 1JWH, 1K4C, 1K74, 1KAC, 1KLU, 1KTZ, 1KXP, 1KXQ, 1MAH, 1ML0, 1MLC, 1N8O, 1NCA, 1NSN, 1OC0, 1OFU, 1OPH, 1PPE, 1PVH, 1QA9, 1QFW, 1R0R, 1RLB, 1RV6, 1S1Q, 1SBB, 1T6B, 1TMQ, 1UDI, 1US7, 1VFB, 1WDW, 1WEJ, 1XD3, 1XU1, 1Z0K, 1Z5Y, 1ZHH, 1ZHI, 2A5T, 2A9K, 2ABZ, 2AJF, 2AYO, 2B42, 2B4J, 2BTF, 2FD6, 2FJU, 2G77, 2HLE, 2HQS, 2I25, 2J0T, 2JEL, 2MTA, 2O8V, 2OOB, 2OOR, 2PCC, 2SIC, 2SNI, 2UUY, 2VDB, 3BP8, 3D5S, 3SGQ, 4CPA, 7CEI //END OF EASY 116

1ACB, 1ATN, 1BGX, 1BKD, 1DE4, 1E4K, 1EER, 1F6M, 1FAK, 1FQ1, 1GP2, 1GRN, 1H1V, 1HE8, 1I2M, 1IB1, 1IBR, 1IJK, 1IRA, 1JIW, 1JK9, 1JMO, 1JZD, 1K5D, 1KKL, 1LFD, 1M10, 1MQ8, 1NW9, 1PXV, 1R6Q, 1R8S, 1SYX, 1WQ1, 1XQS, 1Y64, 1ZLI, 1ZM4, 2C0L, 2CFH, 2HMI, 2HRK, 2I9B, 2IDO, 2J7P, 2NZ8, 2O3B, 2OT3, 2OZA, 2Z0E, 3CPH

Appendix B: Internal Data

```
struct Complex
{
// Energies
float tot_energy;
float bond_energy;
float angle_energy;
float improper_energy;
float dihedral_energy;
float vdw_energy;
float tot_intermolecular_energy;
float intermol_vwd_energy;
float intermol_electrostatic_energy;
float desolvation_energy;
float alpha_carbon_RMSD;

// Buried Surface Area
float act_bsa;
float pct_bsa;
float pct_cpx_bsa;
float p1_delta_sa;
float p1_pct_delta_sa;
float p2_delta_sa;
float p2_pct_delta_sa;

// Summed Buried Surface Area
float all_bsa[5];
float tot_bsa[5]; // Sidechain
float main_bsa[5]; // Mainchain
float np_bsa[5]; // Nonpolar
float pol_bsa[5]; // Polar
```

```
int   complex_type; //Eg. enzyme:ligand, antigen:antibody, etc.

int   num_nb_contacts; // Nonbonded contacts
};

struct NBC
{
// Nonbonded contact between two atoms in different proteins.
int amino_acid_type[2]; // 20 Types
int amino_acid_polarity[2]; // Polarity rank 1-20 from Trinquier et al 1998.
int amino_acid_charge[2]; // -1, 0, 1
int residue_num[2];
int atom_type[2]; // 40 types (3 are Hydrogens: 60/61/62)
int strand_id[2]; // ? how to scale these numbers?
float residue_dec[2];

float dist; //A
};

struct Atom
{
float x, y, z;
float solvent_access; //A^2
float surface_area;
float vdw_radius; //A
int residue_num; // Use this to index into parent amino acid to get
// that data if needed
float residue_dec;
int atom_type;

float logP; //hydrophobicity
float TPSA; //topological polar surface area
float MR; //molar refractivity

bool HBD;
bool HBD2;
bool HBA1;
bool HBA2;
bool PHBD;
int RB;

bool interface;
int votes;
};

struct Residue
{
int strand_id;
int amino_acid_type;
int residue_num; //Self
int polarity_rank;
int charge;
int secondary_struct_element; // 7 Types

// Values go up to about 250?
float all_abs, all_rel;
float tot_abs, tot_rel; //Sidechain
float main_abs, main_rel; //Mainchain
float np_abs, np_rel; //Nonpolar
float pol_abs, pol_rel; //Polar

float residue_dec; //Stange cases where we get 181.1 or whatever
};
```

Appendix C: SMARTS Descriptors

From Wikipedia[1]:

> *SMiles ARbitrary Target Specification (SMARTS) is a language for specifying substructural patterns in molecules. The SMARTS line notation is expressive and allows extremely precise and transparent substructural specification and atom typing.*
> *SMARTS is related to the SMILES line notation that is used to encode molecular structures and like SMILES was originally developed by David Weininger and colleagues at Daylight Chemical Information Systems. The most comprehensive descriptions of the SMARTS language can be found in Daylight's SMARTS theory manual, tutorial and examples.*

Below are the specific searches that this work uses. For each of these are are more than one method to specify a query with the named intent. The ones used are hopefully representative of the desired results. We also attempted to compute "Possible Intra-molecular Hydrogen Bonds" but this tended to cause stack overflows on many of the proteins, so it was not included.

[1]http://en.wikipedia.org/wiki/Smiles_arbitrary_target_specification

Number of Hydrogen Bond Donors (HBD) 1[2]

```
[!#6;!H0]
```

Number of Hydrogen Bond Donors (HBD) 2[2]

```
[$([O;H1,-&!$(*-N=O)]),$([S;H1&X2,-&X1]),$([#7;H;!$(*(S(=O)=O)C(F)(F)F);
!$(n1nnnc1);!$(n1nncn1)]),$([#7;-])]
```

Number of Hydrogen Bond Acceptors (HBA) 1[2]

```
[$([!#6;+0]);!$([F,Cl,Br,I]);!$([o,s,nX3]);!$([Nv5,Pv5,Sv4,Sv6])]
```

Number of Hydrogen Bond Acceptors (HBA) 2[2]

```
[$([$([#8,#16]);!$(*=N~O);!$(*~N=O);X1,X2]),$([#7;v3;!$([nH]);!$(*(-a)-a)])]
```

Rotatable bond[3]

```
[!$(*#*)&!D1]-!@[!$(*#*)&!D1]
```

An atom which is not triply bonded and not one-connected i.e.terminal connected by a single non-ring bond to and equivalent atom. Note that logical operators can be applied to bonds ("-&!@"). Here, the overall SMARTS consists of two atoms and one bond. The bond is "site and not ring". *#* any atom triple bonded to any atom. By enclosing this SMARTS in parentheses and preceding with $, this enables us to use $(*#*) to write a recursive SMARTS using that

[2]http://www.ra.cs.uni-tuebingen.de/software/joelib/tutorial/descriptors/descriptors.html
[3]http://www.daylight.com/dayhtml_tutorials/languages/smarts/smarts_examples.html

string as an atom primitive. The purpose is to avoid bonds such as c1ccccc1-C#C which wo be considered rotatable without this specification.