Guilherme José Gago Fial

Bachelor of Computer Science and Engineering

Optimizing Data Selection for Contact Prediction in Proteins

Dissertation submitted for the degree of

Master of Science in
Computer Science and Engineering

Adviser: Ludwig Krippahl, Assistant Professor,
NOVA University of Lisbon

Examination Committee

Chairperson: Jorge Carlos Ferreira Rodrigues da Cruz
Raporteur: Daniel Vieira N. Silva Sobral

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

September, 2019

Optimizing Data Selection for Contact Prediction in Proteins

This document was created using the (pdf)LaTeX processor, based in the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

To my friends, family and love.
Those that make life worth living.
And lastly…for Science!

Acknowledgements

To begin with, I would like to thank the computer science department (*Departamento de Informática*) of the University *Faculdade de Ciências e Tecnologia da Universidade NOVA de Lisboa* for providing me with the opportunity, environment and teachers that allowed me to learn a great deal about the field of computer science.

I would also like to express my gratitude to my supervisor, Ludwig Krippahl, for introducing me to the interesting fields of bioinformatics and machine learning, and for providing his knowledge and supervision that have guided me through this challenging project. To my colleagues, José Carneiro and Pedro Almeida, for the support with the structure of this final document. To everyone that contributed to this field and developed all the necessary tools, for without them, this work would not have been possible.

Lastly, I would like to thank my family, for their interest and support, particularly my Mother, and to my girlfriend Natalia, for always being there with an endless supply of motivation.

Abstract

Proteins are essential to life across all organisms. They act as enzymes, antibodies, transporters of molecules, structural elements, among other important roles. Their ability to interact with specific molecules in a selective manner, is what makes them important.

Being able to understand their interaction can provide many advantages in fields such as drug design and metabolic engineering. Current methods of predicting protein interaction attempt to geometrically fit the structures of two proteins together by generating a large amount of potential configurations and then discriminating the correct pose from the remaining ones.

Given the large search space, approaches to reduce the complexity are often employed. Identifying a contact point between the pairing proteins is a good constraining factor. If at least one contact can be predicted among a small set of possibilities (e.g. 100), the search space will be significantly reduced.

Using structural and evolutionary information of the interacting proteins, a machine learning predictor can be developed for this task. Such evolutionary measures are computed over a substantial amount of homologous sequences, which can be filtered and ordered in many different ways. As a result, a machine learning solution was developed that focused in measuring the effects that differing homolog arrangements can have over the final prediction.

Keywords: Contact prediction, Machine learning, Bioinformatics, Protein-Protein Interactions

Resumo

As proteínas são uma componente fundamental da vida, atuam como enzimas, anticorpos, transporte molecular, elementos estruturais, entre outros. A capacidade de interagirem de uma forma seletiva com outras moléculas é o que as faz importantes.

Perceber como as proteínas interagem pode trazer benefícios em certas áreas, como a da farmacêutica e a de engenharia metabólica. Os métodos atuais que tentam prever interacções entre proteínas começam por gerar um grande número de encaixes possíveis entre as duas proteínas, de entre os quais se tenta identificar o encaixe correto.

Dado o grande espaço de procura, métodos de redução de complexidade são frequentemente empregues. A identificação de um ponto de contacto entre as proteínas é um bom factor de restrição. Se pelo menos um contacto for correctamente identificado dentro de um pequeno conjunto de previsões (e.g. 100), o espaço de procura será bastante reduzido.

Para prever contactos, usam-se classificadores juntamente com informação estrutural e evolucionária das proteínas que interagem. No entanto, métodos de informação evolucionária extraem essa informação com base num grande número de sequências de proteínas homólogas, as quais podem ser filtradas e ordenadas de diferentes maneiras. Após desenvolver classificadores de contactos, a tese foca-se em medir os efeitos de diferentes configurações de homólogos.

Palavras-chave: Proteína, Previsão de Contactos, Aprendizagem Automática, Bioinformática

Contents

List of Figures

List of Tables

Listings

Glossary

contact	Residue-residue contacts, inter-contacts, or simply contacts, are pairs of residues that belong to different proteins, and have a distance no longer than 5Å.
docking	Docking is a computational procedure that seeks to find the position and orientation between two molecules.
hyperparameters	Necessary parameters that control how a machine learning model learns from the data.

Acronyms

ASA Accessible Surface Area.

Chapter 1

Introduction

In this chapter, the overall motivation and objective of this work will be provided, as well as the necessary biochemical background concepts such as the protein and its structure, protein docking and the attributes within proteins that facilitate the intended contact prediction.

1.1 Motivation

Proteins are essential to life across all organisms. They are responsible for most of the work within the cell by regulating its processes and enabling signals to be imported and exported into and out of the cell. They also act as enzymes that catalyze chemical reactions, transporters for small molecules, antibodies, messengers (hormones) and structural components that provide movement, structure and support [Bur08; Pro].

The biological function of the protein depends on its 3-dimensional structure and how it connects to other structures and proteins, not simply on the sequence of building blocks that compose it [Bra+99]. It is therefore fundamental to understand how proteins interact and what is the assembled structure they form, designated a protein complex. Such knowledge would grant scientific advantages in areas like drug design or metabolic engineering. As an example, anti-retro-viral drugs supply molecules that bind to the viruses' reverse transcriptase and block its ability to convert viral RNA to DNA, stopping the viruses' replication capability.

Current practical methods for determining protein structures, such as X-Ray crystallography, are complex and lengthy laboratory processes, with an increase in difficulty depending on the size of the protein. For this reason, despite having resolved structures of individual constituent proteins, the complexes they form are often unknown. As a result, methods that attempt to computationally solve this problem were developed, known

as protein docking.

In the most general form, the docking algorithms attempt to geometrically fit the two protein molecules together. However, doing so with no additional information leads to an overwhelming number of potential conformations to be tested, among which, only a select few correspond to a native pose, as found in the actual complex.

It is computationally unfeasible to thoroughly test each potential pose, and so there is a critical need to filter out bad conformations. Typically, this is done by applying fast filtering functions which reject likely wrong conformations. However, the small number of native poses might still end up being filtered out and the docking will produce no correct results.

One way of filtering would be to develop a prediction mechanism that would issue a set of possible contacting points between the two interacting proteins, ideally guaranteeing at least one correct prediction within the set. From here, conformations can be restricted to those including one of the predicted contact points.

Besides using structural and chemical features, an interesting way of possibly determining the location of such contact points is to look at evolution. That is, by analyzing homologous proteins that carry the same function across different organisms, one can determine evolutionarily conserved or co-evolved points, which can be indicative of contact locations.

1.2 Objectives

The motivation requests the development of a protein-protein contact predictor by using machine learning and features extracted from the constituent proteins' three-dimensional structures and homologous protein sequences.

The homolog protein data is the basis for the extraction of evolutive information, as a result, practical questions were raised relative to the collection and organization of the homolog protein data, and how it affects contact prediction.

For this reason, the optimization of the homolog data selection and a measurement of its contribution are the goal of this thesis. Thus, while there is a variety of structure based attributes that are certainly helpful for contact prediction, the focus is on information that can be extracted from the homolog sequences.

Therefore, different ways of arranging the homolog data were devised and predictions were made under different setups in order to draw a conclusion regarding an ideal data preparation. To determine the predictive success, the protein docking benchmark dataset (DBMK 5.0) is used. It provides known complex structures. This way, predicted contacts are compared to the actual contacts in the structure.

1.3 The Protein

Proteins are molecules present in every living system, their capability of interacting with specific molecules in a very selective manner, including other proteins, makes them essential in most biological processes.
In this subsection we will provide an overview of the structure of the protein, namely the structural properties relevant to our goal of contact prediction.

1.3.1 Amino Acids

Proteins are essentially a chain of amino acids (or residues, when an amino-acid is incorporated into a chain, water is released, turning into an amino-acid residue, or simply residue) linked together by peptide bonds: a polypeptide chain, where the amino-acid is the building block.
Amino-acids consist of a central carbon (C_α), which connects to an hydrogen atom (H), a carboxyl group (COOH), an amino group (NH_2) and a sidechain, denoted by the letter R, as depicted in figure 1.1.
The sidechain sets difference between the 20 distinct amino-acids that can be found

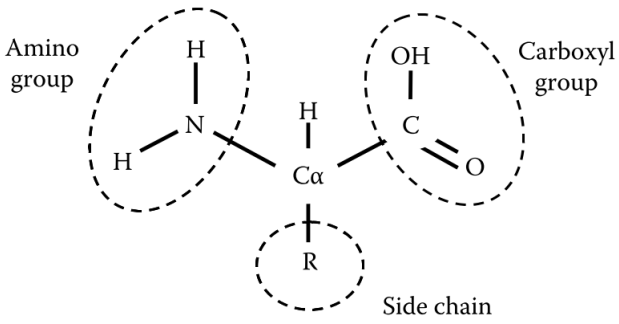

Figure 1.1: Anatomy of an amino acid [Bur08].

on most living systems. These sidechain differences manifest themselves as particular attributes per amino-acid, chemical functionality and structure vary from one to another, however, they are grouped by similar attributes [Gro10].
In this regard, we can arrange them into four main categories: non-polar hydrophobic, polar-uncharged, polar-charged positive and polar-charged negative. These properties can provide an insight about the structure and fold of the protein and possibly how it might bind to other molecules. For instance, hydrophobicity has been demonstrated to affect structural properties. Hydrophobic residues can usually be found on the hydrophobic core of the protein, or in contact with one another when in aqueous environment, due to their water repulsive properties. Hydrophilic residues may in turn remain in contact with water [Bur08].

1.3.2 Structure

The chains of amino-acids that constitute a protein can assume lengths spanning from as few as 20 to more than 5000 residues. In order to be able to portray them, there are four structural description levels. Namely Primary, Secondary, Tertiary and Quaternary (figure 1.2).

Primary Structure The first level, denoted as primary structure, is simply a list of the composing amino-acids in their order of appearance, analogously to reading the protein chain from one end to another. Their spatial distribution is not described.

Secondary Structure The secondary structure is a description of the spatial arrangements of amino-acids in the peptide chain induced by hydrogen bonds. Specifically, these arrangements cause the chain to form a particular recurrent shape, it is this shape that is described and not the physical positioning of the residues in the tridimensional space. Commonly, the secondary structure is categorized by one of three shapes: α-helices, β-strands and loops [Bur08].
The alpha helix takes place when the backbone of the chain winds in a helical conformation around the long axis of the molecule with the R groups of the amino-acids facing outward from the helix. This coil pattern develops due to the hydrogen bond that the residue n has with the residue $n+4$ ahead in the chain. As a result each turn in the helix is composed of 3.6 residues, however, there are other infrequent forms that allow more elongated helices, like the 3_{10} helix with 3 residues per turn that is usually found at the end of alpha helices, or the more uncommon pi helix with 4.4 residues per turn [Bur08].
A beta strand is a strip of the polypeptide chain that is almost stretched in a line. When several beta strands are parallel to each other and connected by hydrogen bonds between their carboxyl oxygens and their amide hydrogens, they form a beta sheet, where this sheet forms a plane-like structure (which may be twisted). When all connected strands rise and fall together, we have a beta-pleated sheet. Additionally, the beta strands can be classified as parallel or anti-parallel. If the strands have the same orientation, i.e. the amino acid indices on the parallel strands increase in the same direction, it is a parallel beta strand. Otherwise, it is labeled as anti-parallel [Bur08; Gro10].
The Loop, or omega (Ω-) loop, is categorized as a non-regular secondary structure, unlike the α-helix and β-strand. This is due to not having repeating hydrogen bonding patterns. The loops are characterized by the loop-like three-dimensional form contracted by the polypeptide chain, usually having the beginning and end residues of the loop proximal in space. Furthermore, they often connect other secondary structures and are in great part found on the surface. The loops have been found to be regularly involved in protein function and molecular recognition [Fet95], most likely as a result of their flexibility and surface availability, along with pattern free arrangement unlike regular structures.
The secondary structure of a residue may contribute to the likelihood of it being a contact.

For instance, it has been observed that contact residues appear to be more frequently part of a loop formation [OR07].

Tertiary Structure The tertiary structure allows the full visualization of a protein in three-dimensional space. Strictly speaking, it provides the full three-dimensional structure of the polypeptide chain with atomic detail. The interactions of residues that are far apart in the primary structure are represented here, mainly through non-covalent interactions, thus allowing the depiction of the positional relationships of the secondary structures [Bur08].

Quaternary Structure Lastly, in the same fashion, the quaternary structure is the spatial relationship of multiple tertiary structures that are grouped together, forming an oligomeric complex. Therefore, an oligomeric complex, or protein complex, is an assembly of several polypeptide chains. In other words, two or more proteins linked together by non-covalent bonds.
The surface area enclosed by two given proteins that are forming a complex is termed interface. However, in the interface only a small subset of residues is actually essential for the binding between the two protein partners, they are commonly referred to as hotspots, or contacts.
The aim of this work is therefore aiding the prediction of protein complexes [Bur08].
How the complex prediction is attempted and how contact residue prediction will assist is covered in the following section.

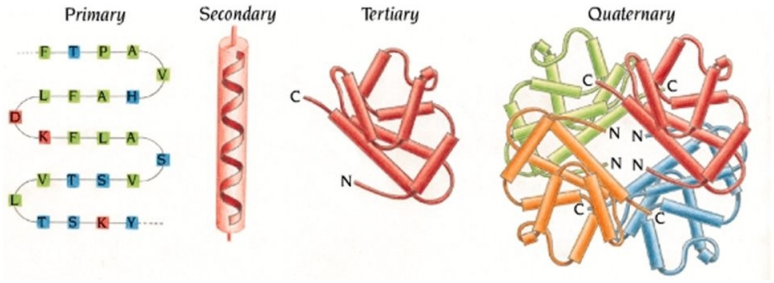

Figure 1.2: The four levels of protein structure [Bra+99].

1.4 Protein Docking

Docking is a computational procedure that seeks to find the best matching configuration between two molecules, designated receptor and ligand, in pursuit of predicting the bound molecule complex resulting from their association [Hal+02], therefore making protein docking the prediction of a bound protein-protein complex given two known protein structures (protein targets)[KB15]. These protein structures naturally have to be experimentally extracted and determined, and it can be done two ways. Either we derive the structure from a complexed state, that is, while it is coupled to any other molecule, or

when it is free in solution in an uncomplexed state (native structure). Consequently, the docking can then be attempted using bound or unbound molecules. The bound docking, the simplest, seeks to dock a receptor and a ligand considering the same shape they have when found connected to one another, disregarding the natural conformational changes that occur when they are separated, this is not realistic. The docking will be most useful when attempting to couple uncomplexed structures, as these are what we have at our disposal when the structure of the complex is unknown. Therefore, efforts should be focused on the unbound docking. Such docking works with molecular structures that may be uncomplexed (native structure), or bound to other molecule other than the one that is being attempted to dock with (pseudo-native structure), or even modeled ones [Hal+02].
In short, the docking problem can be described as follows: "Given the coordinates of two molecules, predict their correct bound association", as stated in [Hal+02].

1.4.1 Docking stages

Computing the docking can be a very expensive task, especially if attempted in its most basic form where we have no additional information other than the structure of the two molecules. This is due to the fact that there is a great number of ways of placing two molecules together while considering three levels of rotational and translational freedom. To illustrate, combining every residue on the surface of a protein with all the surface residues on the other could easily yield a number in the order of 10^7 possible combinations [Hal+02]. The difficulty rises further when taking into account the unbound state and the flexibility of the molecular structures, since each molecule can now pose in several different conformations.
The docking procedure divides the problem in two stages: the searching (or filtering) and the scoring stage [Hal+02].

Search/Filtering stage: With the current computational capabilities, it is unfeasible to perform a thorough evaluation (scoring) of all the possible molecular arrangements in order to sort out the best. To tackle this, a search stage is employed.
At this stage, the goal is to retain a subset of most likely correct molecule configurations from the large sample of possible molecular arrangements. To do so, fast and coarse functions are applied over the search space to weed out incorrect conformations all the while keeping the correct solutions [Hal+02].
The subset of filtered candidates is then passed onto the next stage.

Scoring stage: The goal of the scoring stage is to identify the correct or near-native conformations within the reduced set of candidates provided by the search phase.
Through the use of heavier and more complete functions that implement strategies like

energy minimization, each candidate is evaluated more in depth and attributed a rank. Ideally, the top ranked solutions would correspond to the correct models [Hal+02; KB15].

1.4.2 Improving the Docking

One way of improving the docking results would be to improve the scoring phase, in which case the scoring functions would then rank the correct conformations more accurately. Another way would be to improve the searching phase, which in turn provides more suitable candidates for the scoring phase.
While scoring is important, if the searching fails to select any correct conformations, then a correct model can never be found irrespective of how good the scoring functions are.
It is in this filtering stage that contact prediction may prove to be a useful addition. The locations of possible contacts between the two protein partners can be used as a constraint when picking a set of candidate conformations to pass onto the scoring phase. Even if there is only one correct contact among all the predicted contacts, then we can already increase the fraction of correct docking configurations in the set of candidate models [KB15].

1.5 Information for contact prediction

For distinguishing contacting from non-contacting residues, information is needed. The contacts between proteins happen by virtue of some form of compatibility. If we wish to predict such contacts, attributes that may lead to understanding this compatibility must be gathered and tested with machine learning in pursuit of bettering predictive capabilities.
The following aspects were deemed relevant for feature extraction regarding the prediction of protein-protein contact points.

1.5.1 Evolutionary information

All living beings have relatives and ancestors as a result of evolution, and so do the proteins within them. Proteins from different organisms that carry out the same function are usually similar, in this sense, if two proteins share ancestry between them, they are called homologous. Such homologs often share significant sequence identity, that is, a certain percentage of amino acids in their sequence are observed in the same sequential order.
Since contact residues are essential participants on maintaining and establishing the interaction between two proteins, and considering that the resulting protein may lose functionality or structure upon the loss of a contact, throughout evolution such residues tend to be conserved or correlatively mutated in order to maintain such contact points. In such a way, by analyzing several homologous sequences, insight on which residues have been mutated or conserved across protein relatives can be obtained. Such information

helps to pinpoint some potential contact locations on the target protein [MG08]. Additionally, evolutionary methods allow for a more universal and wide applicability due to conservation patterns being more easily recognizable across different functional sites, where physicochemical properties may vary [AA+15]. Hence, evolutionary data already proved useful for several contact, binding site and interface predictors [Gon+13; KB15; OR07].

1.5.1.1 Sequence alignments

The first step towards extracting useful information from several homolog sequences is to carry out a Multiple Sequence Alignment (MSA). The rationale behind MSA is to, ideally, align multiple homolog sequences in a way that equivalent residues in terms of structure and functionality end up aligned in a one-to-one correspondence across all sequences, with positions that are unable to align being represented by a (-) symbol, called gap [Rus14] (figure 1.3 illustrates an alignment).

```
P69891 HBG1_HUMAN          1  MGHFTEEDKATITSLWGKVNV--EDAGGETLGRLLVVYPWTQRFFDSFGNLSSASAIMGN     58
P02091 HBB1_RAT            1  MVHLTDAEKAAVNGLWGKVNP--DDVGGEALGRLLVVYPWTQRYFDSFGDLSSASAIMGN     58
P02089 HBB2_MOUSE          1  MVHLTDAEKSAVSCLWAKVNP--DEVGGEALGRLLVVYPWTQRYFDSFGDLSSASAIMGN     58
P02070 HBB_BOVIN           1  --MLTAEEKAAVTAFWGKVKV--DEVGGEALGRLLVVYPWTQRFFESFGDLSTADAVMNN     56
P69906 HBA_PANPA           1  -MVLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDL------SHGS     53
                                 ::  :*: :.  *.**     : *.*:* *::: :* *: :*  *.        ..
```

Figure 1.3: MSA of 5 hemoglobin proteins (Generated under the UniProt website using Clustal Omega).

The MSA is the multiple sequence counterpart of a Pairwise Alignment, where only two sequences are aligned.

1.5.1.2 Conserved amino acids

As briefly revealed in the introduction of this section, conservation can be indicative of a possible interaction residue. The logic behind it is that these contact residues are crucial to the whole function of the protein since they mediate the interactions directly, thus having a considerable impact if disrupted. Having taken this into consideration, such residues will likely be under evolutionary pressure against mutation, therefore being identifiable by their conservation across homologs [AA+15].

1.5.1.3 Coevolved amino acids

Residue conservation has had conflicting views regarding hotspot prediction. Some studies have found interacting residues not to be highly differentially conserved in comparison to the rest of the protein, or that their use brings little predictive improvement, while some other studies indeed found conservation to be greater among interface residues, where several methods based solely on conservation delivered significant results. Perhaps such a difference may originate from different datasets and methods [AA+15]. Irrespective of its significance, conservation does not provide mutual information between two interacting

partners, that is, given several conserved residues on one protein and several on the other, no clue is provided from the conservation alone about which pairs of conserved residues are potential contacts [MG08]. To this end, coevolution allows the inferring of potential contacting residues. The rationale is the following: different organisms can have homologous proteins, however, they often differ in amino acid composition, having different amino acids in certain positions [Ovc+14]. By comparing protein complexes existing in different organisms, it can be deduced which amino acids change in both protein partners in a compensatory way, leading to the uncovering of potential contacting residues, since a mutation in one side must often be accompanied by a compensatory mutation on the other side to maintain compatibility [KB15].

1.5.2 Structural and physicochemical information

In the previous subsection, it was explained how evolutionary insights allow us to deduce which residues might be part of protein-protein interactions.
In this subsection, residue structural and physicochemical attributes will be presented. Such properties will help further constrain the possibilities when evaluating potential contacts.

1.5.2.1 Solvent Accessibility

Solvent accessibility is often chosen as one of the most discriminatory structural features for interface and hotspot related predictions [AA+15]. The solvent-accessible surface area depicts how exposed to the solvent is a given area. No exposition to the solvent means that the aforementioned area is buried within the molecule and thus cannot be accessed on the surface, indicating its unlikeliness on making part of a protein-protein interaction, since these interactions take place upon the reachable surface. Solvent accessibility has been shown to be higher in inter-protein contact residues [OR07], when in an uncomplexed state. However, this does not always seem to be true. The O-Ring theory states that contact residues are usually located at the center of a particular interface area and surrounded by energetically less important residues that shape like an O-ring to occlude bulk water molecules from the contact residue [LL09]. Though this is applied over a bigger patch of surface, it has inspired the use of both the residue and its surrounding nearby residues accessibilities as features.

1.5.2.2 Hydrophobicity

Hydrophobicity has proven to be of great value to the prediction of protein interactions, there is an evident overall tendency of hydrophobic residues to interact with each other, as they constitute the most common residue pairing. Moreover, hydrophobic-hydrophilic residue pairings were associated low contact values as well [Gla+01].
Additionally, the hydrophobicity of neighboring residues might be of relevance considering their evident proximity to the residue at hand. That is, in an actual contact, the surface

neighborhoods of either contacting residue are likely to be proximally close and should have some form of complementarity. Therefore, both the hydrophobicity of residues and their neighbors constitute interesting properties to help determine the feasibility of a potential contact.
The following amino acids are hydrophobic [BR03]:

- Very hydrophobic: Valine, Isoleucine, Leucine, Methionine, Phenylalanine, Tryptophan and Cysteine.
- Less hydrophobic (or indifferent): Alanine, Tyrosine, Histidine, Threonine, Serine, Proline and Glycine.
- Part hydrophobic (i.e. the part of the side-chain nearest to the main-chain): Arginine and Lysine.

1.5.2.3 Other attributes

Other residue attributes were initially considered and implemented, such as residue type, secondary structure and charge. However, they were eventually left out as the objective shifted to analyze the impact of different homolog data arrangements. Different arrangements merely impact evolutionary features, thus, optimizing the contact detection by means of further structural and physicochemical features is suitable for future work. The research and rationale of choice for such features can be found under this section.

Residue type The amino acids differ from each other and have different representations when considering hotspot residue composition. It has been reported that the fundamental hotspot forming residues are tryptophan, arginine, and tyrosine, composing 21%, 13.3% and 12.3 % of hotspots found in interfaces, respectively. On the other hand, other types such as leucine, serine, threonine and valine, seem to be underrepresented as hotspots [Mor+07].

Secondary Structure It has been shown that the secondary structures to which the amino acids belong to are not equally represented in hotspots. In addition, their distribution was found to vary between hotspot interface residues and non-hotspot residues by [OR07; Wan+12], having 57% of hotspot residues belonging to a loop structure. A possible explanation being that loops are somewhat more flexible and might constitute good contact points. To this end, it is interesting to use the type of secondary structure associated to a given residue as an additional attribute that might help pointing out a most likely hotspot residue.

Charge Certain residues have either a negative or a positive charge, given that opposite charges attract each other and that same charges repulse one another, such properties

may help determine the likeliness of two given residues forming a contact. Oppositely charged residues have demonstrated a tendency to be in contact with each other while negatively charged residues displayed a low contact propensity. However, there was an average pairing tendency among positively charged residues, e.g. arginine and lysine. In this particular case, the arginine and lysine had a specific orientation so that their charged amino groups would be as far as possible from each other, seemingly to minimize the repulsion between them. They appear to constitute a pair due their solvent accessibility and close packing attributed to hydrophobic interactions, which diminish their mutual repulsion.
Additionally, there was a favorable connection between hydrophobic and charged residues [Gla+01], advocating for an interplay of both physicochemical attributes.

1.6 Summary

To summarize, the objective is to develop a contact predictor based on structural and evolutionary information, with a focus on the latter, and then test for differences in the predictive quality depending on the protein homolog arrangement, which is the basis for deriving evolutionary information.

To ready homolog protein data such that any evolutionary based attributes can be computed, the homologs of each protein must be aligned, as explained in detail above at 1.5.1.1. From these alignments, conservation and co-evolutive attributes are then computed.
For conservation (1.5.1.2), each residue in the protein is attributed a value describing how conserved it is across the homologs, it is therefore dependent on the quality of the available homologs for that protein, and the alignment produced.
For co-evolution (1.5.1.3), a value is attributed for each pair of residues between two connecting proteins, which is dependent on the homolog alignments of both proteins and which homologs from either protein are assumed to connect with each other. Thus being subject to not only the homolog quality, but the order between the two sets of homologs.

From the protein structure, the solvent accessibility (1.5.2.1) is computed for each residue. It is undoubtedly a good structural descriptor for this problem, as most contact residues are on the surface of the proteins, and can therefore rule out residues that are too buried into the protein. As discussed earlier, more structural attributes could be used, but those would not be affected by the homolog configurations.

Another attribute is hydrophobicity (1.5.2.2), residues are given a value representing how hydrophobic they are. This information can be used in conjunction with the homolog alignments, providing descriptors such as the average hydrophobicity across the homologs for a particular residue.

Multiple features are then computed based on this information. This is explained in detail during the development chapter, in the feature extraction section at 3.3.2.

Chapter 2

State of the art

In the introduction, several important concepts necessary for understanding this project were presented. A brief description of the techniques used to implement such concepts along with a comparison among different state-of-the-art implementations is provided in this chapter. In the first three sections, the specialized software and techniques required for extracting information from evolutionary, structural and physicochemical is discussed. Following, common machine learning methodologies are introduced that were initially considered for use. Afterwards, it is explained how to evaluate the prediction results on this particular problem, where and why the data is obtained from and lastly, other relevant contact prediction related methods are discussed.

2.1 Evolutionary information

In this section, the necessary methodology for the extraction of evolutionary information is described and analyzed.

2.1.1 Protein Sequence Comparison

Comparing protein sequences is the base procedure upon which homology can be inferred, and homology is the basis for extrapolating any kind of evolutionary information.

This subsection will present the principal mechanisms with which sequences are compared.

2.1.1.1 Pairwise Sequence Alignment

In order to be able to measure how identical two protein sequences may be, a two sequence alignment must be performed, termed the Pairwise Alignment. From such an alignment a score is then deduced that quantifies their similarity.

The two standard scores for analyzing the similarity between two sequences are the percent identity and percent similarity [Pea13], and the two types of alignment that can be performed are the global and the local alignment [Rus14].

```
HFCGGSLINENWVVTAAHCGVT----TSDVVVAGEFDQGSSSEKIQKLKI
|.|||||||..|||:||||.|.    ..:.:|||...:.....:|:::|
HQCGGSLINNLWVVSAAHCHVVFYGGGQNEIVAGLHRKSEVDSSVQRIEI
```

Figure 2.1: A pairwise alignment between two sequences.

Global and local alignments

There are two main approaches to sequence alignment, the global and the local alignment. The global alignment takes the entire sequences into account, that is, it seeks to match the sequences end to end with the highest possible score, it is best used when the sequences are of similar lengths and homologous. On the other hand, the local alignment aims at matching parts of the sequences without being forced to align them entirely, such an approach is more suitable for the discovery of conserved domains, or to align sequences significantly different in length.

In this work, the conservation and coevolution of residues is to be explored by aligning protein homologs that should interact in the same way, as such, global alignment is assumed to be the most suitable method [Rus14]. It is presumed that such proteins should be globally similar.

Similarity scores

After two sequences are aligned, residue with residue, we can evaluate the alignment with percent identity or percent similarity, the differences are explained below.

The percent identity uses an identity matrix that simply attributes 1 to identical residue pairs and 0 to all other pairs. This essentially results in counting the matching residues on the alignment that have the same amino-acid (perfect match) and dividing the result by the length of either the smaller or larger sequence to obtain a percentage. If divided by the smallest length, it will essentially rate how well does the smaller sequence match with any sub-part of the bigger sequence (e.g. sequence ABBACA and ABBA are 100% similar), this is suitable to rank local alignments. Contrarily, if divided by the bigger length, the extremity gaps and size differences matter, which is more suitable for the situation where one intends to find homolog sequences and not just fragments or regions, that is, a global alignment (e.g. ABBACA and ABBA are now 67% similar).

Percent similarity follows the same strategy as percent identity explained above, except that it uses a substitution matrix rather than an identity one.

In short, substitution matrices are matrices that quantify the interchangeability between any two residue types, that is, residues that often replace one another when analyzing mutations between homologous sequences. If any two residues map to a positive

value, they have been observed to replace one another whenever mutations occur in homologous sequences, if they map to a negative value instead, they have been observed to avoid being replaced by one another.

Hence, by using a substitution matrix, a value of 1 will be attributed whenever the aligned residue pair has a positive number, otherwise, 0 will be attributed.

Among substitution matrices, the most common ones are PAM (Point Accepted Mutation) [Day+78] and BLOSUM (Blocks Substitution Matrix) [HH92].

To summarize, percent identity has a very strict matching mechanism, where only the exact same residues are accepted as a match. Well rated sequences regarding identity are indeed very similar. Once homology has been established, it is a reasonable similarity score [Pea13].

On the other hand, in contrast with identity, percent similarity has a more relaxed matching mechanism, since it accepts residue pairs on the alignment that have been observed to substitute each other according to any given substitution matrix. Well rated sequences with this approach are not necessarily well rated on identity.

The standard algorithms for implementing local and global alignments are the Smith-Waterman and Needleman-wunsch algorithms [NW70], respectfully.

The pairwise2 module of the Biopython package [Coc+09] already implements the alignment algorithms.

2.1.1.2 Multiple Sequence Alignment

As briefed over the introduction under 1.5.1.1, the Multiple Sequence Alignment (MSA) procedure aligns three or more protein sequences with assumed evolutionary relationship and is the preceding step required to be able to discern evolutionary characteristics such as the conservation and coevolution of residues.

The most common algorithms used for global multiple sequence alignments have been the Clustal algorithms. The T-Coffee algorithms are also fairly popular due to their increased accuracy [SH14]. In this work the latest version of the Clustal algorithms, Clustal Omega, will be used for the alignments. The reason for choosing Clustal over T-Coffee, at least initially, is that the T-Coffee increased accuracy of 5-10% comes at the expense of performance, usually being limited to a few hundred sequences. Oppositely, Clustal Omega allows for sizable alignments (190,000 sequences took a few hours on a single processor) while maintaining accurate results. Furthermore, Clustal Omega revealed through benchmarking tests to be more accurate than most of the frequently used fast methods and still comparable to some of the slow intensive methods [Sie+11].

2.1.2 Conserved residues

The two most recognized tools for calculating evolutionary conservation at each site of a given MSA are AL2CO [PG01] and Rate4Site [Pup+02][AA+15; CP09].

The choice is Rate4Site since it is reportedly more accurate than previous methods [May+04], among the best for large alignments with more than 50 sequences [JT10] and it has been consistently used in research related to contact and hot-spot prediction [AA+15; Ezk+09; MG08; Tun+09].

The computed scores are a relative measure of evolutionary conservation, a low value indicates that the position in the MSA is most conserved.

2.1.3 Coevolved residues

In the coevolution approach, residue positions within a MSA are analyzed for correlated changes as a sign of coevolution and therefore direct residue contact. One issue is that residues might coevolve indirectly, i.e. if residue A is coupled to residue B and B to a residue C, then one may falsely conclude that A is in contact with C [Kam+13]. To tackle this, recent methods such as Direct Coupling Analysis (DCA) [Mor+11] and PSICOV [Jon+12] used an inverted residue-residue covariance matrix technique to accomplish the separation of indirectly and directly coevolved residues [Kam+13]. However, more recently, methods such as GREMLIN [Kam+13] and plmDCA [Eke+13] have achieved higher accuracy with a different approach (Pseudo-Likelihood Maximization) [Kam+13; See+14]. CCMPred [See+14] is an efficient open-source implementation of these top performing approaches and will be the chosen software for this task[See+14].

CCMPred calculates direct coevolution strength between any two residues of a MSA [Zho+17], as it is designed to predict intra-contact residues to help fold prediction. To measure the coevolutive strength between two partner proteins of a complex, both MSA's belonging to each one must be concatenated, linking sequences belonging to the same species (same complex), creating a paired sequence alignment. The paired alignment will then be fed into CCMPred to reveal direct coevolving residues within the complex, where only residue pairs (potential contacts) that are on the surface and belong to both partners are to be considered.

2.2 Structural characteristics

2.2.1 Solvent accessibility

Solvent accessibility, or Accessible Surface Area (ASA), is calculated based on an algorithm which simulates a probe rolling around the Van der Waal's surface of the molecule [LR71; SR73]. The probe is typically given the radius size of water (1.4 Å), which is the solvent, and then rolled over the 3-dimensional coordinate structure of the protein. The path traced out by the center of the probe defines the accessible area [HN93; LR71; SR73].

There are several tools to calculate ASA values [Cav+03; HN93; Mih+08; Mit16; Tou+15], however, they generally follow the same algorithm and therefore output similar values, the main differences lay in usability, format and calculation speed.

DSSP was initially considered, as it is a standard database/tool to use with PDB entries which provides secondary structure and accessibility information [Tou+15]. However, the structure files extracted from DBMK sometimes had non-standard residues which caused DSSP to ignore them and possibly not account for their interference with neighboring residues.

As a result, a python library that provides utility functions to analyze PDB structures was used to calculate the ASA values [Ho18].

2.2.2 Secondary Structure

To determine the secondary structures within a given protein structure, the *de facto* standard is the DSSP (Dictionary of Protein Secondary Structure) software and databank [KS83; Tou+15]. Essentially, the DSSP software determines the secondary structures and calculates other structural properties from the PDB entries and places them on the equally named databank [Tou+15]. DSSP does not predict the secondary structure, it assigns it based on hydrogen bonding patterns [KS83].

In a study published in 2005, the authors of SEGNO [Cub+05] compared it to DSSP and STRIDE [HF04], having concluded that the majority of assignments provided by the different programs are similar (80%), where SEGNO is slightly more accurate at defining the end of secondary structures. However, SEGNO appears not to be available online anymore. Furthermore, improvements were made to DSSP in 2011-2012 [Tou+15] and no comparison was found after. In 2015, in a study about approaches to protein-protein interactions, DSSP is mentioned as the sole recommended secondary structure assignment tool [AA+15].

2.3 Physicochemical characteristics

2.3.1 Hydrophobicity

In order to more accurately represent the hydrophobicity of the amino acids beyond the simplified categorization of either hydrophobic or non hydrophobic, hydrophobicity scales also can be employed. An amino acid scale provides a numerical value for each amino acid that represents its relative hydrophobicity [Bis+03].

There are, however, a great number of scales (hundreds) that oftentimes represent amino acids in a contradictory way due to variation in their calculation methods [Bis+03]. Due to the lack of consensus across the published scales, a study [TS98] already considered 144 scales and averaged the results, dividing the amino acids into three categories: Hydrophobic (WCMFILVGRS), Hydrophilic (EDKNQHY) and Ambivalent (ATP).

While that may provide an opportunity for further research, it is not the main focus on this thesis. For this reason, a frequently cited scale shall be considered. One such scale that determines hydrophobicity based on surface accessibility and combines two other well known scales is the Kyte-Doolittle index [KD82].

The hydrophobic scale can be taken from the AAIndex database [Ezk+09].

2.4 Machine Learning

Machine learning is a field of computer science and artificial intelligence that endows systems with the ability to learn from data, therefore allowing them to adapt and to generalize, as comparable to learning from experience. It is inherently multi-disciplinary since it allows a computer to learn about anything given that is backed up by data, thus being applicable to different contexts like physics, statistics or chemistry.

Machine learning comes to assist computational problems that cannot be solved by explicit programming due to the inability, incomplete understanding, or unacceptable computational costs and complexity on calculating the solution. In essence, machine learning algorithms attempt to learn how to predict the correct output for a given input without the knowledge of the actual true process that yields the result. Therefore, in order to achieve learning, an algorithm must adapt and modify its actions in the direction of improving the outcome. This can be accomplished with three main algorithmic approaches: Supervised learning; Unsupervised learning and Reinforcement learning [Alp10].

Essentially, in supervised learning, an algorithm learns by examples. A provided training set of examples (i.e. known as targets) that holds correct answers (i.e. intended categorization) allows an algorithm to detect a pattern and thus generalize correct answers for all input.

Supervised learning algorithms can be further divided into two main groups, classification and regression. For the classification problem a discrete number of values is being predicted, that is, given information about an element, predict to which class it belongs, e.g. when analyzing a protein surface residue, classify the residue as either a contact or not. Oppositely, for the regression problem continuous values are being predicted, e.g. when predicting the binding strength of a molecule with another, determine the value of the binding strength.

The supervised learning method will be the focus of this project since the intention is to use classified data for the training of a classifier.

In the following sections, different classification algorithms considered for contact prediction will be addressed.

2.4.1 Decision Trees

Decision trees have a low computational cost. They are inexpensive to create and to query (i.e. $O(\log N)$), which makes them an attractive choice. The basic idea behind them is that classification is processed in a branch like fashion. By starting at the root node, the input is successively split or broken down through test functions on the decision nodes, eventually narrowing into a leaf node where the definite classification is obtained. In

essence, the trees constitute a set of if-then logical disjunctions being applied over an input and can therefore be easily interpreted.

Since leaf nodes represent particular regions in the input space where instances that fall into the same regions belong to the same category, in a classification problem they would correspond to the same class whereas in a regression problem they would correspond to a similar numeric value.

In an univariate tree, the test functions that determine the branching, or split, manipulate a single input dimension at a time. If the dimension is discrete and contains n values, a n-way split will ensue, and the decision node will have n branches. Otherwise, for a numeric input, the test will apply a comparison to generate a binary split. This pattern continues with each decision node successively dividing further until it is no longer necessary, where a leaf node labels the output.

Multivariate trees can pick combinations of features, as opposed to one. This can lead to considerable smaller trees, but univariate trees are simpler to understand and visualize, and also tend to provide better results [Mar09].

The quality of a split is determined by an impurity measure. A split is considered pure if, for all the branches arising from the split, all the instances that choose a given branch belong to the same class. To illustrate, let m be a node, and N the number of training instances that reach m. If m is the root node, then all instances reach m, therefore $N_m = N$. N_m^i represents those with class C_i that have reached m, so $\sum_i N_m^i = N_m$. The estimated probability of class C_i given that an instance has reached node m is $p_m^i = \frac{N_m^i}{N_m}$.

Node m is considered pure if for each class either all or none of the instances belonging to that class reach m. Thats is, for all classes C_i, p_m^i is either 0 or 1. Consequently, a pure split eliminates the need for further splitting, since it is able to fully discriminate a class.

One way to measure impurity is through the entropy function:

$$I_m = -\sum_{i=1}^{K} p_m^i \log_2 p_m^i \tag{2.1}$$

This function describes the amount of information obtained by knowing the value of a feature. For a two class problem, if all of the examples score positive, then there there is no information gain and the entropy would be 0, since for any value the feature may take, the result is always positive. However, if a given feature separates the examples by 50% positive and negative, the entropy would be maximized, i.e. 1, as it represents the best discrimination possible. The same argument holds for $K > 2$, i.e. for when there are more than two classes, making the largest entropy in turn $log_2 K$ when $p^i = 1/K$.

There are other measuring options besides entropy. Lets assume a two class problem where $p^1 \equiv p$, $p^2 \equiv 1 - p$ and $\phi(p, 1-p)$ is a non-negative function that measures the impurity of a split and that satisfies the following properties:

- $\phi(\frac{1}{2}, \frac{1}{2}) \geq \phi(p, 1-p)$, for any $p \in [0, 1]$.
- $\phi(0, 1) = \phi(1, 0) = 0$.

- $\phi(p, 1-p)$ increases with $p \in [0, \frac{1}{2}]$ and decreases with $p \in [\frac{1}{2}, 1]$.

Options are:

1. Entropy
$$\phi(p, 1-p) = -p\log_2 p - (1-p)\log_2(1-p) \tag{2.2}$$
The entropy equation 2.1 generalizes for $K > 2$ classes.

2. Gini Index
$$\phi(p, 1-p) = 2p(1-p) \tag{2.3}$$

3. Misclassification error
$$\phi(p, 1-p) = 1 - max(p, 1-p) \tag{2.4}$$

These measures can be generalized for $K > 2$ classes.

If a node m is not pure, a split will be chosen to reduce impurity. There are multiple attributes to which a split might be applied, the attribute that reduces impurity the most after the split will be picked in an attempt to generate the smallest tree. This greedy behavior leads to locally optimal solutions, there is no guarantee that the smallest tree is found. Splitting also favors attributes with many values, which may pose a problem, since the impurity is reduced the most when creating a lot of branches despite the relevance of the feature. This property along with noisy data might cause the tree to grow very large and overfit. A possible solution is to stop the tree construction once the nodes become pure enough, that is, when a given threshold is surpassed, i.e. data shall not be further split when $I < \theta_I$. Additionally, the trees can also be pruned, namely prepruned or postpruned. Prepruning stops further splits whenever insufficient instances are reaching a given node, this avoids variance and generalization error by keeping decisions based on too few instances from taking place. Postpruning happens after a full tree construction and is responsible for the removal of subtrees that cause overfitting [Alp10].

Random Forest

Random Forests are an ensemble learning method. Such method improves predictive results by using multiple learning algorithms which would not perform as good on their own. Particularly, random forests use multiple decision trees.

This approach provides a solution to the tendency to overfit observed in individual trees and resistance to noisy data.

The algorithm works in the following way [LW02]:

1. n_{tree} bootstrap samples are extracted from the original data

2. Generate an unpruned tree from each of one the n_{tree} samples, with a different splitting technique: For each node, instead of choosing the best split amid all predictors, choose from m_{try}, where m_{try} is a random subsample from all predictors

3. Predict new data by querying all of the n_{tree} trees and computing a majority or average, for classification or regression purposes, respectively

2.4.2 Support Vector Machines

To introduce Support Vector Machines, or SVM, one may start with the concept of linear discrimination.

The easiest classification scenario is one where the classes, or labeled data, can be separated using a simple linear discriminator. In linear discrimination, a line, defined by a function

$$g(x) = w^T x + w_0 \tag{2.5}$$

separates the instances of both classes in a graph where each feature composes an axis. In the equation, x is a data point, w is a weight vector and w_0 is the bias, or threshold. Assuming y is a class label, and $y \in \{+1, -1\}$, then $g(x) > 0$ if $y_x = 1$, or $g(x) < 0$ if $y_x = -1$. Thus, all points belonging to one class will be in one side of the line, whereas points belonging to the other class will stand on the opposite side.

Support Vector Machines work with the concept of maximum margin for a linear separator.
By considering a set of linear separators with the same orientation (parallel to one another), the distance between the separators that are further apart will constitute a margin. In this regard, among all possible separating lines with various orientations, one will have a maximum margin. The data points that define the boundaries of the lines composing the maximum separators are called support vectors. The concept is shown in figure 2.2.

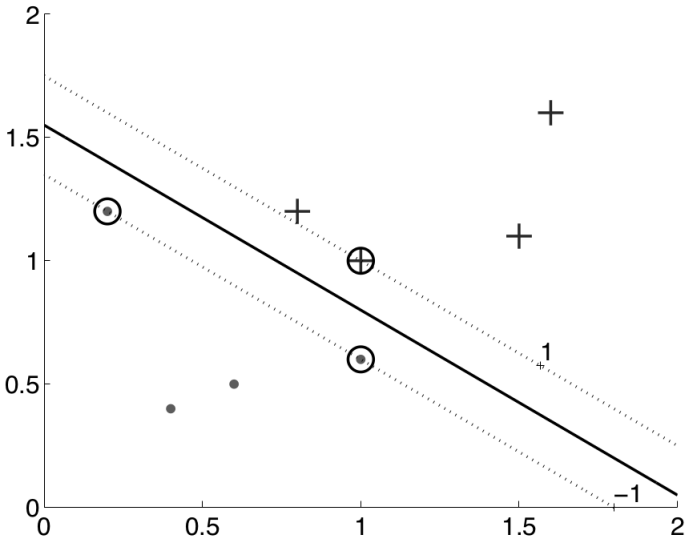

Figure 2.2: For a two class problem, the thick line defines the boundary and the dotted lines the limit of the margin on both sides. The dot and plus signs represent the classes, the ones encircled are the support vectors [Alp10].

The main idea behind the SVM algorithm is therefore to identify the $g(x) = 0$ linear separator with the maximum margin of separation.

The logic behind this is that the maximum margin separator is the optimal separator, as classifiers with lower margins have demonstrated a higher associated risk of misclassification.

When a set of data points cannot be linearly separated, a strategy called kernelling can be employed, where the number of dimensions is increased to try and find a linear separation in a higher dimensional space. Since the number of features determines the dimension of the input space (the data points), in order to obtain a higher dimension a new feature is computed based on the already existent features through a kernel function. Consequently, if a linear separation is attained at a higher dimensional space of $n + 1$ dimensions, the resulting separator in our n dimensional input space will be an hyperplane. However, it may happen that no amount of higher dimensions will be capable of separating the data points, in which case an hyperplane that minimizes the error has to be selected [Bur08].

Due to the general use of more than three features, it is not possible to visualize or understand the resulting vector data, as it resides in a Euclidean space of high dimension.

Although SVMs were initially considered, such techniques do not scale well with a large amount of samples. After encoding protein data into samples, a total of 3.282.479 samples were obtained. This number would result in very slow training times, especially considering the large number of hyperparameters to optimize.

2.4.3 Neural Networks

Neural networks are inspired by the brain. Since brains are able to learn, then, understanding them and copying their learning mechanisms should be of use to the machine learning field.

The brain is very complex, but its building blocks, the neurons, may be described in a simplistic way that approximates their behavior. Basically, the neuron is an electrically excitable nervous cell that transmits information through electrical impulses. An impulse is fired when the electric potential within the neuron rises above a certain threshold. Neurons can connect to each other through synapses and are typically connected to thousands of other neurons on two different ends, a receiving end, where impulses are collected from the connected neurons through branches called dendrites, and a giving end (axon), where its own impulse is sent to the other connected neurons. Together, they form a neural network.

Each neuron can be visualized as a separate processor that performs the simple computation of deciding whether or not to fire [Mar09], allowing it to be simulated on a computer. By interconnecting the artificial neurons together we obtain the neural networks machine learning technique.

In order to simulate the neuron, although simplistically, a mathematical model was introduced in 1943 to describe the bare essentials of its operation. A neuron is thus modeled as:

1. A set of weighted inputs w_i, meant to model the synapses (different synapses contribute differently)
2. An adder that sums the input signals (simulates the collection of electrical charge in the cell)
3. An activation function responsible for deciding whether or not a neuron fires (threshold)

A set of input nodes $\{x_1,...,x_m\}$ are fed into the neuron. They represent impulses coming from the synapses of the connected neurons. Each x takes a value, 1 means that the input neuron fired and 0 that it did not. They can also take up values like 0.5, despite not having a biological meaning. Synapses coming into the neuron have different strengths associated with them, represented by the weights. Hence, the strength gathered from input signals is calculated by

$$h = \sum_{i=1}^{m} w_i x_i, \tag{2.6}$$

If a weight w_i is 0, the corresponding input signal x_i is irrelevant for our neuron. If it is positive, it contributes towards firing by increasing the strength within the neuron, otherwise, if it is negative, it has an inhibitory effect. The altering of the weights allows the neuron to learn. Lastly, the neuron has to decide if it is going to fire, and that is if the collected strength surpasses a given threshold θ. If $\theta = 0$, the neuron fires whenever $h > 0$. Therefore, this simple activation function translates to

$$g(h) = \begin{cases} 1, & \text{if } h > \theta. \\ 0, & \text{if } h \leq \theta. \end{cases} \tag{2.7}$$

In the neural network, these neurons are typically arranged in layers. The input layer directly collects the inputs from the system, while the output layer provides the final results of the computation. Hidden layers exist between the former and the latter, collecting inputs from the previous layer and providing output to the next layer [Mar09].

Neural Networks can work really well, but they are time consuming to set up due the vast amount of hyperparameters and the very long time required to train. This does not make a good candidate for validating against different homolog settings.

2.4.4 Logistic Regression

Despite having regression in its name, Logistic Regression is a classification model. Likely due to the fact that it uses the output of a logistic function, commonly known as sigmoid function, to decide whether a given sample belongs to a class.

The sigmoid function has a characteristic S-shape 2.3, as it is defined by the formula 2.8.

$$\phi(z) = \frac{1}{1 + e^{-z}} \tag{2.8}$$

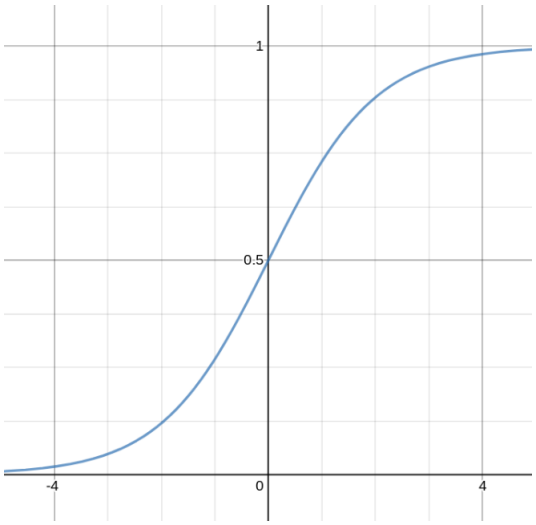

Figure 2.3: Sigmoid activation function.

This function converts any real number into the [0,1] range. Thus, when z is the the linear combination of the m features and their affected weights, $z = w^T x = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m$, the output can be interpreted as a probability of sample x belonging to the positive class [RM17].

The classifier is simple in itself, but it is one of the most widely used algorithms in the industry and it generally provides good results.

Its simplicity is a strong point, as it requires little configuration and is fast to train. It is therefore very suitable to use for the particular problem in this thesis where models have to be optimized and trained from the beginning with each different data configuration.

2.4.5 Naïve Bayes Classifier (NBC)

Naive Bayes models have been particular successful at the problem of information retrieval [Lew98; MN98], which makes it an interesting choice due to its similarity with contact prediction, where the disproportion of contacts in relation to non contacts resemble that of the disparity of relevant documents against non relevant documents when making a query on a search engine.

Unlike the previous classifiers, NBC follows a generative model and assumes that all features are independent of each other given the context of the class, i.e. conditionally independent. Probabilities are associated to each feature given the class, carrying out the

classification of an example according to the product of probabilities of the example's features in relation to each class [KB15; MN98]. In other words, classification occurs by choosing the class with the greatest probability of having generated the example [MN98].

While this assumption of independence is practically false on most real world examples, NBC often performs well at classifying, additionally giving insights about the relevance of features.

2.5 Evaluation

Once a classifier is set and predictions are being made, the next concern is how to evaluate its performance. One needs to be able to assess the expected error or predictive performance of a classification model to be able to do any sort of comparison or draw any conclusion.

Algorithms cannot be evaluated solely on the training data. An algorithm performing very well on training data might be overfitting or overtraining, that is, the model is learning not only the underlying function that generally matches the observed samples but also how to explain the noise in the particular training data being used [Alp10], thus failing to generalize and consequently leading to greater prediction errors in data outside of the training set. Similarly, different algorithms cannot be compared based on training errors since the more complex model allowing more parameters will adjust better and yield less errors [Alp10], e.g., a higher polynomial function can adapt better than a lower one, but that simply proves that one can overfit better than its counterpart, not that it actually generalizes better. For this reason we need a validation set that is different from the training set, thus guaranteeing that the data used to evaluate the error and prediction has not been seen before by the learning mechanism. However, a single validation set might be insufficient when testing different models: First, both the training and validation sets may be small and subject to noise and outliers, leading us to false conclusions; Second, factors of random nature might affect the generalization, i.e., algorithms often start with random initial conditions (like the weights in the previously introduced neural networks), thus leading to possibly very different results in different runs over the same data. To solve this, multiple runs can be employed to create an average, since training and validating only once cannot account for the effect of the mentioned factors [Alp10].

Before continuing, one should understand that conclusions obtained from analyzing the performance of machine learning techniques are dependent on the particular dataset being used. The comparison between different methods is therefore not domain independent, but related to a specific application. Therefore, it cannot be said that one algorithm is generally better than any other, it is only quantified how well it fits our particular data.

An additional factor to consider is that the validation set indirectly affects the training of the algorithm, since it is still used as a form of guidance to achieve better predictions, for example, to decide which algorithm performs better, or to determine when to stop learning to avoid overfitting. Therefore, after having trained and decided on a best model,

in order to report the expected error rate, a separate test set that has never been included in the training process shall be used. Additionally, such a set should be large enough for the estimate to be meaningful. Typically, one third of the available data is left as a test set, with the remaining two thirds being used for cross-validation, which is explained further down. In short, the training set optimizes the parameters, given a particular method or algorithm, while the validation set optimizes the hyperparameters of the method or algorithm. Finally, the test set is then used when both parameters have been optimized [Alp10].

2.5.1 Cross-Validation

In order to obtain the average of multiple validation runs, as mentioned in the previous section, one needs multiple training and validation sets. Such sets will need to be extracted from a dataset X, leaving a part out for the test set.

Because datasets are limited and not infinite, there is not enough data to allow for the extraction of multiple training and validation datasets. As a solution, cross-validation is introduced, which consists of repeatedly using the same data but split differently over multiple sets. However, cross-validation makes error percentages dependent, as data is shared between the sets.

When performing cross-validation, additional aspects have to be considered. The training and validation sets should be as large as possible to allow for more robust error estimates, while the overlapping between them should as small as possible. Additionally, classes are to be represented in the same proportions within all the subsets so that the class probabilities remain unaltered; this is termed stratification [Alp10].

K-Fold Cross-Validation

One common method of performing cross-validation is the K-Fold technique, where the dataset X is randomly divided into K equal parts. To create each training-validation pair, one of the K parts is left out as a validation set and the remaining K-1 parts are then combined to form the training set. This is done K times, leaving another of the K parts as validation each time, so that each part is used as a validation set. There are however two problems here. First, the effort to keep a large training set ends up leaving small validation sets. Second, there is a considerable overlap among the training datasets, i.e. any two training datasets share K-2 elements.

By increasing K the training sets grow bigger and therefore a more robust estimator is built, but, on the other hand, the validation sets shrink. In addition, there is an increased cost of training the classifier K times [Alp10].

2.5.2 Measurement of classifier performance

Now that ways of testing the predictions have been introduced, the next step is to provide measurements, or metrics, to assess the performance and the particularities of the

predictors.

In a two-class classification problem, where there is either a positive or a negative class, e.g., to be classified as a protein-protein contact or not, there are four possibilities for a given prediction: A positive example was either correctly predicted as positive, in which case it is named true positive, or incorrectly predicted as negative, becoming a false negative instead. In the same way, a negative example successfully predicted as negative is a true negative, where its incorrect prediction as positive is labeled a false positive. Based on these four prediction cases, the following measurements are regularly applied on binary classification problems [SR15].

- True positive rate (tp-rate), Recall or Sensitivity: tp/p
- False positive rate (fp-rate): fp/n
- Precision: tp/p'
- Specificity: $tn/n = 1 - fp - rate$
- Error: $(fp + fn)/N$
- Accuracy: $(tp + tn)/N = 1 - error$

Where tp is the number of true positives, tn the number of true negatives, fp the number of false positives, fn the number of false negatives, p the number of existing positives, p' the number of instances predicted as positive, n the number of negatives and N the total number of instances.

In the context of this work, these common metrics can be misleading due to the highly imbalanced dataset. For example, the true positive rate (recall) is the percentage of existing true contacts that were correctly identified. However, having a recall of 1 does not prevent falsely identified contacts, since simply classifying all the samples as contacts would earn a recall of 1 while providing a meaningless classification. Conversely, the false positive rate represents the proportion of non-contacts that were incorrectly identified as contacts, but considering that the amount of non-contact samples greatly outweighs the number of contact samples, classifying everything as a non-contact would provide a great score despite the unhelpful classification. Similarly, accuracy and error are unsuitable, their consideration of either false negatives or true negatives causes them to reward a classification that correctly predicts the majority of the negatives (non-contacts), since they are in much greater quantity, therefore being insensitive to the quality of the positive predictions (contacts).

From these metrics, precision is the most sensible to the contact prediction problem, it provides the ratio between correctly predicted contacts to all predicted as contacts. However, this is still not enough to judge the quality of the predictions.

The contact prediction problem resembles the situation encountered in information retrieval where the number of relevant documents are far outnumbered by the number of

non-relevant documents and it is important that the relevant are among the top ranked documents. In the same way, there are far less actual contact residues than non-contacting residues, but it is extremely important that at least one contact is predicted correctly so that one correct conformation is filtered by the searching stage of docking. Consequently, it is far more valuable to have at least one correct prediction per complex among the first 100 predictions than an overall better precision score.

For this reason, the main metric to be used to evaluate the performance of the classifier will be the percentage of complexes that had at least one correct prediction in their top 100 contact predictions.

2.6 Data

2.6.1 Protein Data Bank

PDB, an acronym for Protein Data Bank, is an archive for biological macromolecular crystal structures that was established in 1971. In its birth, it stored merely seven structures. Now, as a result of a dramatic increase in structure determination and depositing throughout the years, it holds around 141.000 structures, where around 93% are proteins.

For each deposition, the PDB assigns a unique PDB_ID and stores the structural coordinate data as well as other general required information, i.e. amino acid sequence and species, that can be extracted by the means of a PDB file. This allows for other datasets that work with proteins, such as benchmarks and homolog databases, to simply identify them with their PDB_ID.

2.6.2 Data Extraction

In order to carry out the intended prediction of inter-residue contacts between two proteins forming a complex, the following data is required:

1. Accurate protein complex data for the training and testing of the algorithms.

2. Homolog protein data for both proteins in the complexes in (1).

Firstly (1), complex data will be fetched from the Protein-Protein Docking Benchmark Version 5.0. The protein-protein benchmarking datasets were designed with the aim of providing curated complex structures suitable for testing docking predictions [Jan+03], which in turn makes them also appropriate for inter-protein contact prediction and the posterior testing of the docking results [KB15].

From the fetched benchmarking data, antibody-antigen complexes will not be considered. The reason is that antibodies do not coevolve with antigens, they are synthesized by V(D)J recombination instead [KB15], rendering the coevolutionary feature useless.

As a second step (2), for each complex, homolog sequences for both of the protein partners are to be retrieved. To do so, the UniRef (UniProt Reference Clusters) [Suz+07]

database will be used to search for a wide range of homolog samples that share at least 50% of sequence identity (UniRef50) with the query protein. Because the data is organized in clusters in UniProt, the homolog samples are quickly retrieved and constitute a greater and unbiased data set, bringing advantages over other common approaches like the BLAST search, which may return few homologs due to server side computing limitations, or PSI-BLAST, which finds homologs based on conservation profiles, yielding conservation biased samples [KB15].

2.7 Comparable methods

Most research related to hotspot or contact prediction uses an already existent protein complex structure and attempts to uncover hotspots or interfacial patches on the contacting surface between the two target proteins. Otherwise, with no present complex, it is also common the prediction of an interface area, or its constituent residues. Since contacts are naturally present around interfaces and hotspots, such research was also considered, but cannot be directly compared to contact prediction.

The presented methods are comparable since they work on the field of contact prediction, but no previous work was found on comparing different homolog configurations.

2.7.1 Coevolutionary-derived contact predictors

CCMPred [See+14] is a tool used for measuring the coevolutionary strength between any two residues, originally meant for predicting contacts within the same protein, i.e. intra-contacts, to tackle the problem of protein fold prediction. When trying to solve the three dimensional structure (fold) of a protein sequence, co-evolving residues are a strong indicator of connection and closeness in space. This is similar to the problem of complex prediction in the sense that the three dimensional structure of a bigger assembly involving different proteins is trying to be resolved. Thus, if instead of using the homologs of a single protein and derive co-evolution between residues within that same protein sequence (as for intra-contact prediction), the concatenated homologs of both interacting proteins (as if the complex would be one long protein sequence) are used instead, co-evolution can then be derived between residues belonging to each of the constituent proteins [Zho+17].

CCMPred [See+14] was thus chosen as a feature for measuring the coevolutionary strength between any two residues, but can be used as a pure coevolutionary contact predictor in itself [Zho+17].

There are several other coevolutionary based methods researched by [AC16], among them the most accurate are GREMLIN[Kam+13] and plmDCA[Eke+13], which CCMPred implements.

EVcomplex [Hop+14] is another method that resembles CCMPred in terms of accuracy and methodology, albeit slower [Zho+17]. The authors have also attempted to improve

docking by applying restraints on the HADDOCK docking framework using multiple predicted contacts at a time.

2.7.2 Machine learning-based contact predictors

Intra-contact predictors

Several state-of-the-art machine learning-based contact predictors were examined by [AC16], DNcon [EC12] was the most accurate.

These methods are intended for intra-contact prediction using only sequential information [AC16], and have shown success in protein structure prediction tasks, e.g. fold prediction. They have specialized approaches for short, medium, and long range contacts (how apart are the contacts in sequence), which are irrelevant for inter-contact prediction considering that contacts will be part of different chains.

Because this project is designed to aid the docking procedure, there is an implicit access to the structures of both protein partners, which are naturally omitted by DNcon and related contact predictors as fold prediction facilitators, who have to rely on predictions for some structural attributes [EC12].

Inter-contact predictors

Krippahl, L., & Barahona, P. (2015) [KB15] also aimed at the prediction of inter-contacts in order to assist docking. Likewise, the goal was to aid the searching stage of the docking procedure (further information at 1.4) by incorporating contact information into the BiGGER docking framework.

Despite using a different set of features, they are still similarly based on physicochemical, structural and evolutionary attributes of the proteins, with predictions being carried out using a Naïve Bayes Classifier.

This work will address the same problem but focus on probing for differences in the homolog data configuration.

Zhou, T., Wang, S., & Xu, J. (2017) [Zho+17] have predicted inter-residue contacts by using a deep learning approach initially designed for intra-contact prediction. However, despite good results (improvement over pure coevolutionary inter-contact prediction), their work has a different focus, as it still uses individual protein sequences, having to predict some structural features and possibly missing structural insights that may help the specialized prediction of inter-contacts (i.e. properties of the neighbor residues).

Chapter 3

Development

This chapter will focus on the fundamental details that concern the development of the algorithms and processes needed in order to set up the proposed environment for the testing and extraction of conclusions.

3.1 Tools

Python is one of the most common programming languages used for the implementation of machine learning solutions [KDn13]. This is due to its easy syntactical character and the presence of an extensive set of easily available and usable machine learning libraries with efficient state-of-the-art implementations. For these reasons and a present familiarity with the language, Python was the first choice. The following libraries were considered:

NumPy [Van+11] is an important Python package that introduces multidimensional arrays and a set of accompanying mathematical functions which provide a high-level abstraction for numerical computation without affecting performance. It is part of the SciPy [Oli07] fundamental library for scientific computing.

Scikit-Learn [Ped+11] is a Python library that provides efficient state-of-art implementations for most of the well known machine learning algorithms. Also supports validation and feature manipulation.

BioPython [Coc+09; HM03] is a collection of libraries for Python designed to cover a wide range of bioinformatics problems, such as the parsing of protein information, the conversion between different formats, and the representation of protein sequences and structure as data objects.

Matplotlib [Hun07] is a graphical package for Python that provides 2D graphing capabilities. It offers high quality graphical output for major 2D plot types, including *xy* plots, scatter plots, bar charts, etc.

Pandas [McK11] is a Python library that provides integrated routines for performing common data operations and analysis over structured datasets, such as indexing, labeling, handling missing values, performing group operations and more.

3.2 Data Preparation

This section will explain how to obtain and prepare the data in order for it to be ready for feature extraction. Essentially expanding on 2.6.2.

3.2.1 Homolog Extraction

To obtain suitable homolog sequences, the following steps were taken:

1. Download benchmark 5.0 data [Vre+15].
2. Download sequence data from PDB [Ber+00].
3. Query UniRef50 [Suz+07] for protein clusters.
4. Download the UniRef50 [Suz+07] protein clusters.
5. Download sequences from UniProt [Bat19].

(1) Benchmark 5.0 Data
As mentioned in 2.6.2, the first step is to fetch the complexes from DBMK 5.0, rejecting those which are the antibody-antigen type.

A benchmark 5.0 data table (excel file) along with the correspondent protein structure files is easily accessible. The data from the benchmark table contains the following information per entry:

1. A complex identifier of the form *CCCC_A:B*.
2. A first/left (receptor) protein identifier of the form *PPPP_A*.
3. A second/right (ligand) partner protein identifier with the same form *PPPP_A*.
4. Other fields with additional information (e.g. difficulty, complex type (category), etc. . .).

In (1), *CCCC* is a 4 alphanumerical character code identifying the protein complex (e.g. 1AKJ), *A* and *B* are chain letters, each identifying a protein chain in the provided complex structure file. The fact that they are separated by a colon represents that chains on the

left side are part of one partner protein while chains on the right are part of the other, that is, *A* belongs to the left/receptor protein while *B* belongs to the right/ligand protein. Furthermore, multiple chains can be specified on either side (e.g. *CCCC_AD:BE*), meaning that there is more than one chain for the respective partner protein(s). The difference is that instead of simply having to account for the interaction between *A* and *B*, now *A* can interact additionally with *E*. The same applies for the other chains, *D* can also interact with *B* and/or *E*.

Likewise, in (2) and (3), *PPPP* is also a 4 character code identifying the protein molecule, with *A* being a chain letter identifying the chain in the protein structure file.

Below is a simplified example of three entries in the benchmark data table. Regarding

Table 3.1: Benchmark 5.0 entries

Complex Protein	**Category**	**Partner Protein 1**	**Partner Protein 2**
1AKJ_AB:DE	OX	2CLR_DE	1CD8_AB
1BVN_P:T	EI	1PIG_	1HOE_
1E6E_A:B	ES	1EIN_A	1CJE_D

the protein structural information, the obtained files correspond to bound structure files and unbound structure files. The bound files contain the structures of the complexes, while the unbound contain the structures of the partner proteins. Hence the chain identifiers frequently not matching between complex identifiers and partner identifiers, such as in the case of *1AKJ_AB:DE* (see 3.1), where the *1AKJ* complex chains *AB* correspond to the *2CLR* (partner 1) chains *DE*, and *DE* on the complex to *AB* chains on *1CD8_AB* (partner 2). The position relative to the colon character determines the chains that map to either partner protein 1 or partner 2, left and right, respectively.

In the same way, for the more common single-chain cases like *1E6E_A:B*, the chain *A* is the also the chain *A* from partner 1 (*1E1N*) and the chain *B* is the chain *D* on partner 2 (*1CJE*). However, in some of these cases, like *1BVN_P:T*, the partner proteins may not specify any chain (*1PIG_* and *1HOE_*), that represents the fact that there is only one chain in the protein (generally chain *A*).

(2) PDB Sequence Data

After having the benchmark data defined above, the actual amino-acid sequences are obtained from PDB [Ber+00].

To do so, for every complex, a request is sent to the PDB database using its REST API for the chain sequences of each of the partner proteins. For each partner protein identifier, all its chain sequences are collected.

Using the previous example complex *1AKJ_AB:DE* (see 3.1), the sequences for the chains that belong to *2CLR* and *1CD8* were fetched.

(3) Queries UniRef50 for Clusters
Once all the chain sequences are downloaded, their respective homologs are to be obtained.

For every unique protein chain sequence across all partner proteins in the complexes, a BLAST query is submitted against the UniRef50 database through the EMBL-EBI Web Services using the default parameters of UniProt, spawning a job. After the job is completed, the result is a file that contains the query results: an ordered list of homolog clusters, from those with a smaller e-value (or greater score), to those with a greater e-value (or lesser score).

Each cluster is represented by a main homolog sequence: the representative. This sequence is named representative because it was the one which was matched and compared with the submitted query sequence (the partner protein chain) and all the other sequences in the cluster have a percent identity of 50% or more with the representative.

(4) Clusters of UniRef50
The previous step provides a list of homolog clusters for each query sequence, but only the cluster identifiers are present, not the clusters themselves, therefore, for each protein partner across every complex, their respective cluster lists are parsed for the identifiers of all the homolog clusters. Each cluster is then downloaded from the UniRef platform by querying the identifiers.

(5) Sequences of UniProt
The clusters, as retrieved in the previous step, do not have the actual homolog sequences besides the representative sequence. They have identifiers for the sequences, along with information such as the species they originated from. As a result, the sequences are to be retrieved from UniProtKB or UniParc. However, not all should be retrieved depending on the species they originated from, as explained below.

At this point, a list full of homologs of a given chain is ready to be extracted from the clusters. But if we are to assume that they are homologs of the complex as well, and not only from the concerned chain, then an homolog originating from the same species must exist for every other chain is the complex. In other words, if an homolog of one chain exists in a given organism from which there are no recorded homologs for the other chains, then it is unlikely that this homolog holds the same function or is part of an actual homolog complex protein, not to mention that there would be no same-species homologs on the connecting chains as to excerpt co-evolutive information. Therefore, only homologs under such conditions should be retrieved.

Resorting to the example of complex *1AKJ_AB:DE*, there are a total of three unique chains involved in this interaction:

- *A* from *1CD8_AB* (*B* is identical to *A*).
- *D* from *2CLR_DE*.
- *E* from *2CLR_DE*.

The outcome results in two sequence pairs that should theoretically co-evolve on inter-contacting residues:

- *1CD8_A - 1CLR_D* (*A* from *1CD8_AB* with *D* from *2CLR_DE*).
- *1CD8_A - 1CLR_E* (*A* from *1CD8_AB* with *E* from *2CLR_DE*).

Where each chain has homologs originating from the same organism.

3.2.2 Preparing the homologs

In the previous subsection all the suitable homologs for the benchmark complex data were gathered. Now, in order to excerpt any sort of evolutionary information from them, they need to be aligned, sorted and paired. This subsection will explain that in detail.

Recapitulating the previous section (3.2.1), when a search (query) for homologs is done for a given chain sequence, the immediate result is a list of clusters, which in turn contain multiple protein sequences that are 50% or more similar to the representative sequence of the cluster. These clusters are ordered by a score representing how well their representative sequence aligned with our query sequence.

Let us consider two sequences from two chains that interact with each other: *1EIN_A* and *1CJE_D* from the complex *1E6E_A:B* (see 3.1).

Two queries, one for each chain sequence, are submitted to UniRef50 and its clusters obtained from the reply. After, the homolog sequences are extracted from the clusters, such that we obtain two lists of homologs: one for chain *1EIN_A* and another for chain *1CJE_D*.

To have results for co-evolutionary analysis, it has to be correctly assumed which homologs from *1EIN_A* supposedly pair to those from *1CJE_D*, so that one can assume to have an actual homolog complex.

For this reason, the base strategy is that only homolog sequences that belong to a species that is found among both partners' homolog lists are kept, subsequently being ordered and matched by species on both ends. This would be enough should there only be one homolog per species, e.g. if there is only one homolog of chain *1EIN_A* and one homolog of the chain *1CJE_D* that originate from the cow (Bos Taurus), one can only assume that both of these homolog sequences interact with each other within the cow, since they both come from the same organism and are both homologs of the known interacting chains. However, the reality is generally different, as it was found that there

are several homologs per species on both ends and frequently more on one side than the other. This spawned an interesting problem because whenever there are multiple homologs per species for either chain, one cannot be sure which pairs of homologs from that species actually interact. It is in light of this fact that an ordering strategy should be defined.

An additional important factor is the similarity of the homolog sequences in relation to the original chain. Just as homolog sequences with little difference will make it difficult to understand where have residues been mutating over time throughout evolution due to little observed change, when homolog sequences are too dissimilar from the actual chain and are included on the dataset, the likelihood that they align correctly with the other sequences (1.5.1.1), or that they are even functional homologs, drops significantly and can jeopardize the quality of the predictions.

For these reasons the homolog data needs to be ordered and constrained strategically, so that the impact of the supplied homolog data and its arrangement can be measured for its predictive impact, in addition to attaining feasible computational costs.

Therefore, the data must be set to respect three key restrictions:

I A limit to the number of homologs allowed to be aligned.

II A limit to homolog sequences below a certain similarity.

III A strategy or logic by which to order the homologs.

Restriction I: To test whether the number of homologs on the alignment impact the feature calculation and in turn, the target prediction, different predictions can be made considering different sizes. **Restriction II**: Sequences of too poor similarity will likely degrade the quality of the predictions. The effect can be measured by setting this limit at different levels. **Restriction III**: The size restriction I is directly tied to the strategy of ordering the homologs. If the size is to be reduced, then it is important to define which homologs get left behind. However, more importantly, when multiple homologs originating from the same species exist for one or several interacting chains, the order defines which homolog sequences of one chain are assumed to interact with the homolog sequences of the other chains for that species, and which ones are to be discarded (e.g. when 3 homologs originating from the cow exist for chain *1EIN_A* but 5 exist for chain *1CJE_D*, two have to be discarded from *1CJE_D* and the other 3 assumed to pair in the right combination out of 6 possibilities (3×2)).

3.2.2.1 Homolog Similarity

Prior to the ordering/pairing of the homologs, it is required to know before-hand how similar are the homologs to their respective query sequences. This step is required for rejecting homologs below a certain limit, as iterated in point (2), and to allow any ordering that includes similarity, which affects the other points.

As described in 2.1.1.1, any similarity measure, be it percent identity or percent similarity, requires the compared sequences to be aligned. Thus, a pairwise alignment is performed between a chain sequence and each of its homologs using the following configuration:

BLOSUM62 matrix: This substitution matrix was built on clusters of sequences that were at least 62% identical on their aligned positions. It is also the matrix used by BLAST to search for homologs [Pea13]. For every pair of amino-acids, it contains a score representing how likely would one amino-acid replace the other in a mutation. These values were computed by observing mutation rates between amino-acids on sequences at least 62% identical.
Gap-open penalty of -11: This is the penalty often used in conjunction with BLOSUM62 that appears to provide the best results [Pea13]. This is the scoring cost suffered from choosing to assume that a gap has opened for a particular position, rather than align with the next amino-acid.
Gap-extend penalty of -1: The value is used for the same reason as stated above [Pea13]. It is the penalty suffered for assuming that the amino-acid next to a gap is also a gap, meaning that the assumed missing part of the sequence extends up to this next amino-acid.
Penalize-end set to True: This prevents the sequence alignment from being compelled to end sooner and lead to sequence fragments having a better rating, rather than full homolog sequences.

Altering these options for similarity score calculation may impact the final predictions as well, since the homologs are ordered and considered/discarded based on their similarity, therefore it is something that can be considered for future work, but is currently out of scope for this thesis.

3.2.2.2 Homolog Ordering

To test the impact of ordering, different ordering strategies can be employed and will be clarified further ahead:

1. Ordering solely by homolog similarity.
2. No ordering, random sorting.
3. Ordering by cluster position or value.
    a) Sub-ordering by homolog similarity.
    b) No sub-ordering, random sorting.

Ordering by similarity (1) In this strategy, multiple homologs originating from the same species will be ordered and paired solely according to their similarity to the original chain sequence.

Species	Homologs of partner 1
human	Sequence-1455 (70% similar)
human	Sequence-6153 (59% similar)
human	Sequence-9200 (95% similar)
rat	Sequence-1109 (82% similar)
rat	Sequence-1455 (70% similar)

Species	Homologs of partner 2
human	Sequence-8084 (70% similar)
human	Sequence-2001 (66% similar)
human	Sequence-0055 (66% similar)
human	Sequence-5892 (80% similar)
human	Sequence-5150 (90% similar)
rat	Sequence-1200 (90% similar)

Figure 3.1: Unordered and unpaired homologs.

As it can be observed on 3.1, the homologs do not follow any order nor are they in conditions of being paired with one another due to the unneven number on both sides. By simply ordering them by the similarity, the excess (unpaired) gets rejected and the rest gets paired from most similar to less similar.

The final result would translate to 3.2.

Species	Homologs of chain 1	Homologs of chain 2
human	Sequence-9200 (95% similar)	Sequence-5150 (90% similar)
human	Sequence-1455 (70% similar)	Sequence-5892 (80% similar)
human	Sequence-6153 (59% similar)	Sequence-8084 (70% similar)
rat	Sequence-1109 (82% similar)	Sequence-1200 (90% similar)

Figure 3.2: Ordered and paired homologs.

Random ordering (2) This strategy is analogous to similarity ordering 1 except that rather than being ordered, the homologs are randomly shuffled. Therefore, the excess homologs that cannot be paired get rejected randomly, and the pairs are established randomly.

Ordering by cluster (3) As explained in homolog extraction section (3.2.1), homolog sequences are extracted from clusters returned by the query to UniRef50. To reiterate, these clusters contain homologs that are 50% or more similar to the representative sequence in the cluster.

The representative sequence itself represents the cluster, and was the sequence that got aligned when the cluster was matched against the chain of the query. For this reason, the returned clusters for a particular query have associated rating values depending on how well the cluster (representative) aligned with the query sequence, such as a score, e-value (expectation value, number describing how expected is a random unrelated sequence to be as significant), percent similarity and a few others.

For a given query the clusters are ordered by score by default, the cluster with the highest score will appear in first position, and so on.

Oftentimes, different clusters can contain homologs from the same species, as a result, one could generally assume within the same species, that homologs originating from better rated clusters to be evolutionarily closer to the query sequence (since better rated clusters are indicative of more similarity and function). Two approaches can then arise:

- Keep homologs for a given species that come only from the first, best rated, cluster containing that specific species. Thus, for both homolog lists, the closest homologs per species would be saved and subsequently paired.
- Alternatively, instead of just the first, keep the N best rated clusters containing a given species on both homolog lists and order them by position (where N is the minimum number of clusters that contain this particular species on either list). This can be thought of as pairing the clusters, pairing those better positioned on both lists first, and then the second better positioned, et cetera, until N.

For both of the choices of either pairing the first cluster or the N first clusters, the number of sequences per cluster on both homolog lists must be the same and have a particular order, since clusters can have multiple members from a given species. Therefore, an extra tie breaker property is needed to determine which members on one cluster are assumed to pair to which on the opposing cluster. To solve this, one can either order by how similar a homolog sequence is to the original query sequence (likewise 1) or shuffle them randomly (likewise 2).

	Unordered homologs of partner 1	
Species	Origin cluster	Homolog sequence
human	Cluster nº 23	S-0440 (70% similar)
human	Cluster nº 23	S-0303 (90% similar)
human	Cluster nº 23	S-0800 (82% similar)
human	Cluster nº 55	S-6153 (59% similar)
human	Cluster nº 55	S-6240 (55% similar)
human	Cluster nº 104	S-4775 (70% similar)
human	Cluster nº 104	S-9200 (95% similar)
human	Cluster nº 150	S-9199 (60% similar)
human	Cluster nº 150	S-9199 (55% similar)

	Unordered homologs of partner 2	
Species	Origin cluster	Homolog sequence
human	Cluster nº 04	S-2001 (66% similar)
human	Cluster nº 04	S-8084 (70% similar)
human	Cluster nº 04	S-0055 (66% similar)
human	Cluster nº 04	S-8020 (60% similar)
human	Cluster nº 04	S-7755 (60% similar)
human	Cluster nº 40	S-5892 (80% similar)
human	Cluster nº 89	S-6665 (84% similar)
human	Cluster nº 89	S-0450 (80% similar)
human	Cluster nº 89	S-5150 (90% similar)

Figure 3.3: Unordered and unpaired homologs.

To demonstrate, the example tables (3.3) represent two homolog lists that need to be paired, one for each partner protein. To facilitate readability, the tables only present one species (mouse) and are already ordered by cluster. The 4 colors from green to dark red represent the first, second, third and fourth positions where one or several homologs from the mouse were found on that cluster.

For the homologs of partner 1, cluster number 23 was the first one containing homologs belonging to the mouse, whereas cluster number 55 was where mouse homologs were seen for a second time, and so on.

All the other clusters in between, before and after; contain sequences from different species and thus are irrelevant for the mouse case. Each of these clusters can also contain

several other sequences from different species. Such will be considered when iterating over their respective species.

In this particular case, the defined ordering strategy is to order by cluster position, keeping more than 1 cluster, and sub-ordering by similarity.

After pairing the homologs in the example, we should arrive to the result presented in 3.4.

	Homologs of partner 1		Homologs of partner 2	
Species	Origin cluster	Homolog sequence	Origin cluster	Homolog sequence
human	Cluster nº 23	S-0303 (90% similar)	Cluster nº 04	S-8084 (70% similar)
human	Cluster nº 23	S-1109 (82% similar)	Cluster nº 04	S-2001 (66% similar)
human	Cluster nº 23	S-1455 (70% similar)	Cluster nº 04	S-0055 (66% similar)
human	Cluster nº 55	S-6153 (59% similar)	Cluster nº 40	S-5892 (80% similar)
human	Cluster nº 104	S-9200 (95% similar)	Cluster nº 89	S-5150 (90% similar)
human	Cluster nº 104	S-4775 (70% similar)	Cluster nº 89	S-6665 (84% similar)

Figure 3.4: Ordered and paired homologs.

Since co-evolutionary information arises from having sequence pairings from different species, when downsizing the number of pairings (restriction I), keeping different species is the priority, thus, pairs will be iteratively removed from the species that contain the greatest number of homologs first.

The last pairing on the list is removed whenever a reduction demands, since they belong to the least rated cluster and have the lowest combined similarity within that cluster. In the example, it would translate to removing the bottom row (S-4775 paired with S-6665).

3.3 Feature Preparation

After obtaining and preparing the data (3.2), the following will be available:

- Structure pdb files for the complex in its bound conformation.
- Structure pdb files for the complex in its unbound conformation.
- Paired homolog sequences for the chains in the complex.

Before continuing on to the machine learning, the data needs to be converted into a usable feature vector. This section will describe how to extract the features from the data and will provide a feature overview at the end.

3.3.1 Preparatory Steps

This subsection describes the beginning of the feature extraction pipeline. The structural elements are constant and do not change with different homolog ordering or configuration strategies. Naturally, the structures used to extract the features or feature related information will come solely from the unbound files, the bound structures are the solution and thus unbeknownst to the predictor, they are only used to determine contacting residues.

3.3.1.1 Structure Based

Generally, the structure files do not perfectly match their respective sequences from PDB, some residues might be missing or swapped, hence, as a preparatory step, a pairwise alignment must be done between the chain sequences extracted from PDB and their structural file counterparts in order to create a map.

Afterwards, less than ideal complexes should be filtered out. One case being complexes with chains that are less than 50 residues in length, since short chains can act as a fragment and match with a great range of sequences on UniRef which are not necessarily homologous. Another case being complexes whose chains, as fetched from PDB, do not match well with the sequences in the structure file. Lastly, those who had less than 50 homologs were also removed. This resulted in 151 retained complexes.

From these remaining complexes, 98 were randomly split for training and 53 for testing. Care was taken so that the same proportion of difficulty was present in both sets (complexes have a difficulty value that ranges from 1 to 3).

The following step is then to identify the contacting residues (target values). Any pair of residues between the receptor protein and the ligand protein whose non-H atoms are at a distance no greater than 5Å, are considered contacts. This criterion has been used in the CAPRI programme [Jan+03].

Once past the previous steps, the next phase is to calculate the ASA of each residue on the surface of the proteins. The algorithm (2.2.1) works by processing a list of residue objects which describe the type and position in space of the residue's atoms. The tool will only take into account the residues present on the list. Therefore, for each partner protein, all the residues from all its chains must be included at once in order for the ASA values to reflect the interference of any chains within the same partner.

One immediate application for these ASA values is to filter out residues that most likely will not participate in a contact due to having little exposure (being buried inside the protein). For this task, a cut-off value can be defined, below which no residue will be considered. If the value is too low, there will be an increased number of false contacts to classify, if too high, many true contacts will be lost. Since the focus is to have at least one correct contact prediction, it must be ensured that no complex loses all its true contacts in the process. For that reason, the two following indicators were considered: One is the ratio between the number of potential contacts averaged over all complexes to the

minimum number of true contacts left on any complex; The other the percentage of lost true contacts on the complex left with the least amount of true contacts [KB15].

From this evaluation, the cutoff value of 32Å^2 was found to minimize the product of the two ratios and was thus selected. The number of potential contacts across the training data was reduced from 7.677.596 to 2.258.607 (70,6% reduction) while the amount of true contacts decreased from 7.568 to 5.277 (30,3% reduction). In respect to the ratio of the average number of potential contacts to the least amount of true contacts, it substantially increased from 2736:1 to 1029:1.

3.3.1.2 Homology Based

Before any features can be extracted from homolog information, a specific data preparation strategy needs to be defined, as explained in 3.2.2. Once a particular setup is chosen and the homolog data is pruned and ordered, a set of computations is then carried out over the data. More information can be found on 2.1.

Recapitulating, for a given complex, the setup essentially determines which homolog sequences of one partner pair to those from the opposing partner, but to obtain evolutive information one needs to align these sequence pairings with all other pairings in order to study the changes experienced by the residues of the protein throughout time. For this, a Multiple Sequence Alignment (MSA) is performed over the paired homolog lists of each partner (2.1.1.2).

With the aligned homologs, co-evolutive information can now be inferred from the MSA's. As explained in 2.1.3, both MSA's, one from each of the contacting partner chains, need to be concatenated as if their paired homologs became one single sequence, essentially representing an alignment of homolog complexes. Additionally, any alignment columns that have gaps in the main sequence need to be discarded as they are not relevant and contribute negatively for the RAM consumption of the tool, which grows exponentially with the length of the alignment (number of columns). The values are then calculated into a matrix that describes the co-evolutive strength between any two residues (columns) in the alignment, and stored for later use.

The previously computed alignments are also used to infer evolutionary conservation information. Each MSA is used an an input for the conservation tool (2.1.2) and a table is produced and stored, mapping each residue of the main sequence to a normalized conservation value. The more negative the value is, the more conserved the residue is on the alignment. However, a certain obstacle occurs when using this tool: If the alignment has more than 200 sequences, an arithmetic underflow might ensue. For this reason, even if more sequences are available, a maximum of 200 have to be picked for use in the conservation calculus. Whenever this happens, the first 200 sequences in the alignment that belong to different species are kept and used.

Once having performed this batch of computations for each complex in the data, this data setup is ready to have its features drawn out.

3.3.2 Feature Extraction

Now that the data is prepared, features can start to be collected. Each sample to be classified consists of a potential contact: For a given complex, a potential contact comes from two potentially contacting chains in the complex, one from each partner, and two potentially contacting residues, each belonging to each chain.

The collected feature information that follows is respective to these potentially interacting residues, obtained either from the structure or from the homolog alignments, as described in the previous section (3.3.1).

First, the structural based features are extracted. For the two residues in the sample: Both ASA values are collected, setting a minimum and a maximum ASA value for the interaction; In the same way, a minimum and a maximum hydrophobicity value is collected according to the type of the residue, as determined by the hydrophobicity scale (2.3.1).

Next, the following homology based features are obtained: co-evolutive value between the two residues, as stored in the previously calculated matrix; minimum and maximum MSA conservation values, as previously determined; minimum and maximum averaged hydrophobicity values for the corresponding positions in the MSA (average hydrophobicity for the same position in the MSA across all homologs); minimum and maximum fraction of gaps for the corresponding MSA positions and minimum and maximum fraction of gaps relative to the total amount across the whole MSA.

Following, additional features are created by aggregating these initial descriptors from the neighboring surface residues. The neighborhood consists of the residue itself plus all candidate residues within contact distance. From here, different aggregation strategies can be used: By averaging the attributes of all the neighbor residues on one partner to the average of the neighbors on the other partner (all-to-all) or by averaging the neighborhood on one partner to the opposing residue on the other partner (all-to-one).

All initial descriptors then get all-to-all features computed over their neighborhoods. Considering the two sets of neighbors from both residues involved in the interaction, average the scores of each set into a minimum and a maximum neighborhood value for the descriptor being aggregated. For the co-evolutive descriptor, which is not a partner independent attribute, since two opposing residues need to be specified for a co-evolutive score, the all-to-one strategy is applied by averaging all the co-evolutive scores between neighborhood residues and the opposing residue. Additionally, spawn other features: one containing the best co-evolving score found among any pair across neighborhoods; another containing the average of the best scores found for each of the N residues that compose the smallest of the two sets; and finally, the best score found between the interaction residues and any opposing neighbor.

This lead to a total of 31 features (listed in tables 3.2 and 3.3).

Table 3.2: Overview of structural features

Structure Features	Description
asa_max	The greatest of the two ASA values for either residue
asa_min	The lesser of the two ASA values for either residue
asa_neighs_max	The greatest of the two average ASA values for either neighborhood
asa_neighs_min	The lesser of the two average ASA values for either neighborhood
hyd_max	The highest of the two hydrophobicity scores for either residue
hyd_min	The lowest of the two hydrophobicity scores for either residue
hyd_neighs_max	The highest of the two neighborhoods averaged hydrophobicity scores
hyd_neighs_max	The lowest of the two neighborhoods averaged hydrophobicity scores

3.3.3 Post Processing

The feature vector, as prepared up to this point, is still not ready to be directly admitted into machine learning models. These necessary corrections can only be addressed once all the features have been prepared. Namely, the scale of the features and the existence of homomers. These issues are covered in the following sections.

3.3.3.1 Normalization

Regarding scaling, certain machine learning techniques, such as Logistic Regression or Support Vector Machines, are sensitive to big differences in the magnitude of the feature values (e.g. for a given residue in a potential contact, ASA values might range between 32 and 277 while gap ratio only ranges between 0 and 1). Features with larger number ranges may desensitize the algorithm towards those with smaller ranges, leading to a longer and harder search for the global minimum [Ras14]. For this reason, it is in general good practice to normalize the feature set.

Two common approaches used for feature normalization are the standardization (sometimes called Z-score normalization) and range normalization.

Z-score normalization rescales the features so that they will have the properties of a normal distribution while having a mean of 0 ($\mu = 0$) and standard deviation of 1 ($\sigma = 1$). Considering a given feature with μ as its mean (average) and σ as its standard deviation, then the normalized value (z) of any given feature value x is calculated according to the formula 3.1.

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

In the range normalization, commonly known as Min-Max scaling, the feature data is simply scaled to a specified range, usually from 0 to 1. With x_{min} as the minimum value found in the feature data and x_{max} as the maximum value, then the normalized value

Table 3.3: Overview of homology features

Homology Features	
r4s_min	The lowest conservation score for either residue
r4s_max	The highest conservation score for either residue
r4s_neighs_min	The lowest average conservation score for either neighborhood
r4s_neighs_max	The highest average conservation score for either neighborhood
hyd_avg_min	The smallest average hydrophobicity score for the MSA position of either residue
hyd_avg_max	The greatest average hydrophobicity score for the MSA position of either residue
gaps_min	The smallest gap percentage for the MSA position of either residue
gaps_max	The greatest gap percentage for the MSA position of either residue
msa_gaps_min	The smallest gap percentage for either position relative to the whole MSA
msa_gaps_max	The greatest gap percentage for either position relative to the whole MSA
coevolution	The co-evolutive score between the two residues
coev_neighs_min	The lowest co-evolutive score of either residue and its opposite neighborhood
coev_neighs_max	The greatest co-evolutive score of either residue and its opposite neighborhood
coev_neighs_avg_min	The lowest co-evolutive score of either residue and its opposite neighborhood
coev_neighs_avg_max	The greatest co-evolutive score of either residue and its opposite neighborhood
coev_all	Averaged best possible co-evolutive scores (multiple) between neighborhoods
max_coev_neighborhood	The best possible co-evolutive (single) score between neighborhoods
avg_hyd_neighs_min	Smallest of the averaged MSA hydrophobicities of either neighborhood
avg_hyd_neighs_max	Greatest of the averaged MSA hydrophobicities of either neighborhood
gaps_neighs_min	Smallest averaged gap percentages (position restricted) for either neighborhood
gaps_neighs_max	Greatest averaged gap percentages (position restricted) for either neighborhood
msa_gaps_neighs_min	Smallest averaged gap percentages (whole MSA) for either neighborhood
msa_gaps_neighs_max	Greatest averaged gap percentages (whole MSA) for either neighborhood

(x_{norm}) of any given feature value x can be obtained through 3.2.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.2}$$

However, Min-Max scaling is sensible to the presence of outliers, all it takes is the presence of a single outlier to set either the minimum or maximum range limit to that respective outlier value, causing the bulk of values to be compressed on a much shorter scale than that which was originally intended. For that reason, when outliers are present, another form of normalization is Robust scaling. For this method, a lower and a upper quartile, Q1 and Q3, are set. Any values outside of the Q1-Q3 range do not influence the scaling. Any given feature value x is calculated according to the formula 3.3.

$$x_{norm} = \frac{x - Q1}{Q3 - Q1} \tag{3.3}$$

The feature normalization pipeline was then set to scale the features according to their properties: Those that do not resemble a normal distribution and do not have significant outliers are normalized to fit between -1 and 1 using the Min-Max scaler; Those remaining which are not based on co-evolutive information are normalized using a Robust scaler with Q1 set to 1% and Q3 set to 99% to reduce the interference caused by the small amount of present outliers.

Regarding co-evolutive features, there is a specially important reason for applying a rescale: The meaning of the computed co-evolutive values varies according to the size of the used MSA (the number of present homologs). These scores do not hold any probabilistic meaning, they act instead as a ranking scale, where those ranked higher are the likeliest to be evolutionarily correlated [See+14]. Since complexes have a different number of available homologs, their co-evolutionary scores are not comparable with one another. The figure 3.5 shows the score distribution for three different random complexes, where one contains two different chain pairings (only D is considered because D and E are homomers).

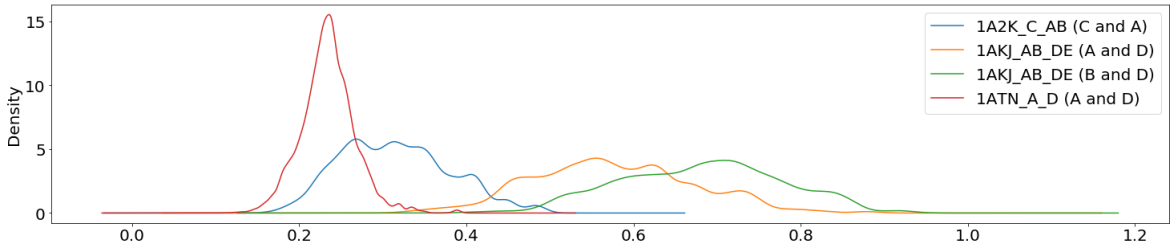

Figure 3.5: Distribution of co-evolutionary values across different complexes before scaler.

Consequently, co-evolution based features have to be normalized respective to the complex they originate from, rather than the whole of the training data.

To do so, a z-score normalization is applied independently over each set of feature values originating from each complex, resulting in the depicted distributions on picture 3.6.

Besides co-evolutionary based features, any data normalization will only be performed

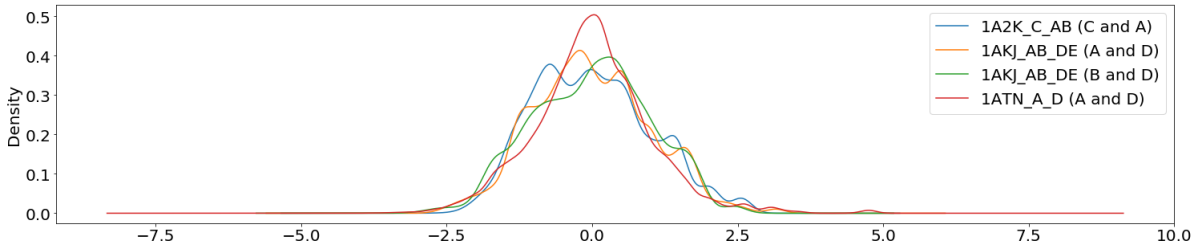

Figure 3.6: Distribution of co-evolutionary values across different complexes after scaler.

later when training data subsets are generated by cross-validation during model building. This is necessary to avoid normalization bias.

3.3.3.2 Homomers

Another issue with the data is the existence of homomers, that is, chains with identical sequences in the same partner protein. This is a problem because from the standpoint of the learner, two or more virtually identical chains, which have nearly the same structural properties and the exact same homolog information, will have contradictive information regarding which residues participate in a contact, since the chains may not use the exact same residues to connect to the opposing partner chain.

One such example can be found on figure 3.7. The chains A, B, C and D all belong to one homomer partner, and have the same sequence. They form a complex with the opposite partner which is just chain E. The putative interactions are $A \leftrightarrow E$ (0 contacts), $B \leftrightarrow E$ (9 contacts), $C \leftrightarrow E$ (38 contacts) and $D \leftrightarrow E$ (2 contacts). Given that A and C have the same sequence, same homology information and identical structure, the same 38 residue interactions that are classed as contacts when the chain is C have the same features as those that get classed as non-contacts when the chain is A, which is conflictive.

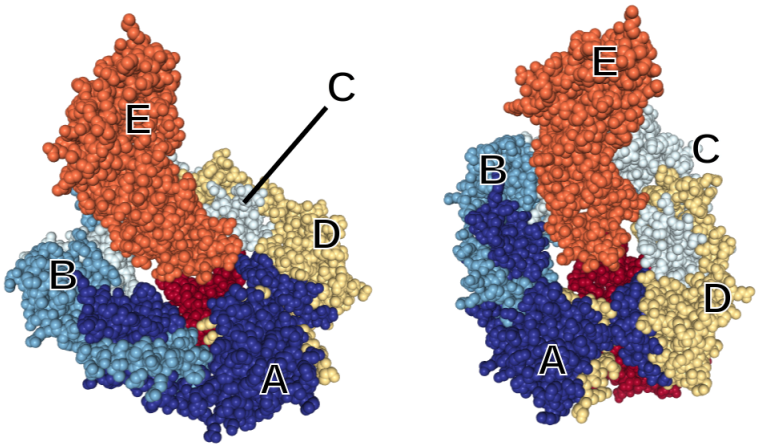

Figure 3.7: Complex 4JCV, how the ABCD chains from one partner connect to chain E from the opposing partner.

To solve this, groups are made for identical chains on either partner. Then, chains that belong to the same groups have their features averaged and their target variables (contact information) compiled. A more detailed explanation follows:

Let $A = a_0,\ldots,a_n$ be the set of chains on the left partner and $B = b_0,\ldots,b_m$ the set of chains on the right partner. Initially, every combination of chain pairs is a potential connecting pair, so, features are computed for $A \times B$ pairs.

Now let $G = g_0,\ldots,g_i$ be a set of groups, where each group g contains chain pairs from $A \times B$ that have identical sequence pairs. e.g. if a_0 and a_1 have the same sequence, then the chain pairs $a_0 \leftrightarrow b_0$ and $a_1 \leftrightarrow b_0$ would belong to the same group.

For every group in G, all its pairs are compiled into one representative pair. This means that every sample (residue-residue interaction) in the representative pair will have the average of the feature values from the compiled pairs and will be considered a contact if at least one of the pairs had it marked as a contact.

Referring back to the earlier example of complex 4JCV (3.7), the four potential chain pairings ($A \leftrightarrow E$, $B \leftrightarrow E$, $C \leftrightarrow E$ and $D \leftrightarrow E$) get collapsed into one with a total of 49 contacts.

3.4 Model Building

In the previous sections it was shown how the homolog and structural data is assembled into a usable feature vector, regardless of the chosen homolog configuration. The next step is to define how to learn from this data and yield comparable results, such that the effects of different homolog configurations can be studied. Therefore, this section will describe how models are built and tested, and how results are collected.

Prior to starting using the data, it must first be split into a training and a test dataset, with the proportions being $\frac{2}{3}$ and $\frac{1}{3}$, respectively. The test data is meant to be used at the end, after all optimization is done, as a form of assessing the performance against truly unseen data. Thus, it is over the training data that the model optimizations are carried out.

3.4.1 Model & Hyperparameter optimization

One first decision is which ML algorithm to use. Different algorithms vary in the way they learn the data, the time and memory required, the recommended feature preparation and their configuration parameters (hyperparameters). Since one cannot be sure which would perform better, one strategy is to start with a baseline algorithm to determine whether predictions can be made by learning this data, and additionally provide insights on the importance of each feature.

3.4.1.1 Model selection

As the priority is testing the effects of different data configurations, and for each configuration the cycle of training and optimizing begins anew, it is important to choose a model that can train and optimize relatively fast.

Logistic Regression was chosen for the bulk of calculations, as it is a common technique used for binary classification which is fast to train, optimize, and generally provides good results, along with an estimate of the contribution that each feature provides towards the prediction of contacts. Other models are generally not as simple and can have many hyperparameters, thus making them slow to optimize and train.

Nonetheless, this model building pipeline is setup such that any technique/algorithm can be used for the predictions. Heavier algorithms are best suited for testing against a potential ideal configuration preliminarly picked up by Logistic Regression.

Before applying any algorithm, its hyperparameters need to be set. These parameters control how an algorithm will learn from the data, and thus, their choice affects the predictive performance.

For example, relevant parameters for Logistic Regression are the inverse regularization strength (C) and penalty. The C parameter is a positive float number that determines how well does the model fit the data, a smaller value will promote a stronger regularization, leading the model to compute a more generalized and simpler decision function that captures the general trend separating the classes, essentially being less sensitive about training samples ending up on the wrong side of the decision boundary. On the other hand, a bigger C value encourages a more complex decision function in order to avoid misclassifying the training samples. Smaller values may risk underfitting, where the model has not learned the data well enough, whereas bigger values may risk overfitting, where the model has adapted to the training data so much that it fails to generalize to unseen data.

The penalty parameter can typically be either L1 or L2, these are two different strategies used to penalize the misclassification done by the model. The former can fully ignore features that do not seem to help the prediction (feature selection) and is robust to outliers, the latter still considers the input of all features and is able to learn a more complex decision function.

It is hard to know beforehand which set of hyperparameters performs better. Thus, a common approach is to search for an ideal configuration by testing different parameterizations and evaluating the outcomes, that is, training a new model for each hyperparameter configuration and keeping the one that yields best performance. This technique is called hyperparameter optimization, and it can be done in different ways.

3.4.1.2 Hyperparameter Optimizers

The three main techniques often used for searching an optimal set of hyperparameters are Grid Search, Random Search and Bayesian Optimization.

Grid search is the most intuitive, it consists of simply trying all parameter combinations. However, when a considerable number of combinations is available, it quickly becomes impractical as it cannot complete within a reasonable amount of time. Furthermore, parameters cannot be chosen from a continuous range of values (e.g. any number between 0 and 1) since that would imply an infinite number of possibilities. Thus, a discrete set of possible values from where to choose is typically defined.

Usually, to tackle either an infinite or a large number of parameter configurations, Random Search is often used. After having defined a maximum number of N iterations, N randomly sampled parameter configurations are tested. This has been shown to produce equal or better results comparatively to grid search given a similar computational budget due to having access to a larger configuration search space [BB12], rather than being bound by defined sets of parameter values to be experimented with.

Lastly, Bayesian Optimization can be seen as an improvement on random searching in the sense that it also does not explore the whole hyperparameter search space, but unlike Random Search, it does not make random choices about which set of parameters to try. Instead, it considers the results of its iterations in order to build a probabilistic model of the function that maps the hyperparameters to the resulting predictive performance. Thus, each tested configuration contributes to a better understanding of the function and further testing is aimed at configurations that are expected to produce better results.

Since Logistic Regression is relatively fast to train and can be tested with a small amount of relevant parameter configurations, Grid Search can be employed. This way, for each homolog arrangement, the explored hyperparameter search space is identical.

3.4.1.3 Data configuration

As explained, during hyperparameter optimization, each iteration trains and tests a new model with its assigned parameters. In order to train and evaluate any model, there must be a training and a validation/test set (more information on 2.5), so that the model can learn from the training set and make predictions against the validation set, generating estimates of its performance over unseen data. In this way, models can then be compared based on their performance against the validation data. However, if fixed portions of the data are attributed to training and validation, where validation is typically 20% of the available data, then predictions can only be made against those 20%, leading to an increased chance factor on the perceived performance. To tackle this, 5-fold cross-validation (more on 2.5.1) was employed.

Cross-validation is a frequently used technique that enables the entirety of the data to be used for validation. Essentially, it works by splitting the data into k folds, and then training the model k times, each time using one of the folds as a validation set and

the remaining as a training set. It should also be noted that the feature data is to be normalized according to training folds (more on 3.3.3.1), as the validation fold acts as unseen data.

The results against each validation fold are then averaged to get a better performance estimate.

There were, however, additional concerns when splitting the data. In the way the data is configured (3.3.2), each sample represents a potential residue-residue contact between two proteins forming a complex. Therefore, it is unwarrantable that samples originating from the same complex end up simultaneously on the training, validation or test datasets. To solve this, a grouping strategy was applied on the data splitting mechanism, where the division occurs at the level of groups of samples rather than the samples themselves, such that all samples pertaining to a given complex are found on a single set.

Another characteristic of the data that influenced the splitting strategy is the existence of a difficulty value attributed to each complex, as specified in the DBMK 5.0, where three levels of difficulty can be found as 1, 2 and 3. Though normally a random split should have a somewhat fair distribution of different difficulties, stratification was used to ensure that similar difficulty proportions end up on both train and test datasets.

3.4.1.4 Optimization Pipeline

The model selection pipeline goes as follows: The training data is split into 5 parts (folds), where no fold contains samples originating from complexes already present in any other fold. Afterwards, a grid of hyperparameter options is defined for the model. It is from this grid that the search mechanism, or optimizer, will pick the parameters to try with each iteration. For example, the hyperparameter grid assembled for Logistic Regression consists of C values: 0.001, 0.01, 0.1, 1, 10, 100 and 1000, and penalty values: L1 and L2. Trying all combinations simply requires 14 iterations, which is low enough to warrant the use of Grid search in this case.

Due to the 5-fold cross-validation, for every iteration, the optimizer will train the model 5 times, once per fold, and average the results.

The results output by the model have to be based on a metric that returns a single value. The chosen metric is the percentage of complexes for which at least one contact was correctly predicted among their top 100 predictions, as the default metrics are not informative on the quality of the predictions (discussed in 2.5.2). This custom metric has to be implemented into a scoring function that will override the default scoring functions of the optimizer.

After the execution of the pipeline, a best set of hyperparameters will emerge. The complex predictions generated for each validation fold by this hyperparameter configuration are kept. This way, other metrics such as mean, median, percentiles and different limits for the first correct contact prediction can be derived from the model and be used to compare against other models emerged from different data configurations.

3.4.2 Final Model

After having built and optimized Logistic Regression models on different data configurations, one or more configurations can be chosen to make the final predictions against the test data. These final predictions should not be used to influence the creation of more data arrangements, they are merely an estimate of the expected error for that configuration performing over unseen data.

That being said, once a particular configuration is selected, the whole training data is now used to train a final model, using the optimized parameters. If the model has not been hyperparameter optimized for this configuration yet, it will first go through the optimization procedure. Lastly, the final model makes predictions against the test data.

Chapter 4

Results and Discussion

In this chapter, the predictive results will be shown and the performance impact for different homolog data configurations will be assessed.

The decision on which configurations to test was based on certain data characteristics found during data preparation that raised a question on whether homolog quantitative, qualitative or ordering restrictions would play an additional factor on predictive capabilities (3.2.2).

Before continuing on to the tests, a summary should be made regarding the interpretation of the resulting predictions. As previously mentioned in the evaluation section at 2.5.2, when contact predictions are made over a given complex, the most relevant aspect is the position (or rank) of the first correctly predicted contact. Thus, a suitable metric to assess contact prediction over multiple complexes is the percentage of such complexes for which a correct prediction was captured within the first N predictions for each respective complex. The chosen N limit for comparing model performances during model selection was 100, a reasonable compromise considering that, on average, train complexes have 18951 potential contacts. However, a wider limit of 200 can additionally be used to help visualize result differences.

The mean, median and percentiles regarding the position of first true contacts across complexes are also informative metrics to use in this scenario.

It is also worth reiterating that for each homolog data configuration, the model building pipeline runs from the beginning using the recomputed training data pertaining to the new configuration. Predictions are then generated for each complex in the training data using cross-validation.

percent identity restriction	# training complexes
0%	98
10%	92
20%	83
30%	70
40%	53
50%	42

Table 4.1: Number of available complexes for several identity thresholds.

4.1 Homolog Similarity Comparison

The most relevant question to be answered by this research is whether restricting the homologs based on how similar they are to their respective proteins, has any impact on the prediction.

In this section, the effects of using different homolog similarity thresholds are tested. First, similarity thresholds based on percent identity, after, based on percent similarity.

4.1.1 Homolog percent identity

To recapitulate 2.1.1.1, percent identity is a measure of how similar a given homolog is to the original sequence. It represents the percentage of amino-acids that are identical between two aligned sequences over their entire lengths.

Having established that a pair of sequences is homologous, percent identity is a reasonable measure of evolutionary distance [Pea13]. Thus, to answer the question of whether it is favorable to restrict the data to sequences most likely to be homologous or closely related, the following datasets were computed: 0%, 10%, 20%, 30%, 40%, 50%.

For these configurations, homologs originating from the same species were simply ordered by the identity, as explained in 3.2.2.2.

An important observation is that as the homolog restrictions increase, less complexes are available, as some cease to have enough homologs above a certain similarity threshold to compute features from (see table 4.1).

For any given configuration, the predictions over its respective available complexes were then evaluated according to the specified metrics. They were then plotted alongside each other, resulting in the following graph 4.1.

Analyzing the results, the mean/average metric was not reliable, as it is greatly affected by outlier complexes, such as $1JTD$, whose first correct prediction is often around rank 5000 across configurations. As complexes end up being rejected, the mean is inevitably susceptible to great variations. For this reason, it was left out. The median and the quantiles are more informative to look at.

The median starts at 34, meaning that without restrictions, half of the complexes have the first correct contact prediction under position 34. It slightly increases as the

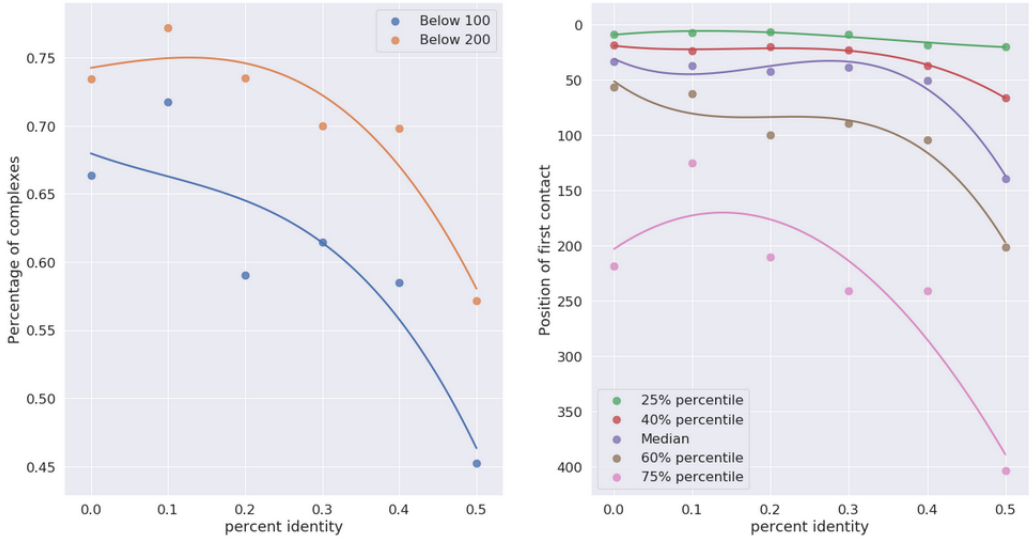

Figure 4.1: Comparison of different percent identity configurations.

restrictions move forward, indicating a general small loss in predictive capability, increasing significantly beyond 40% identity. The situation is similar for the first quarter of complexes (25% quantile). However, there is an interesting observation around the 10% restriction, the first three quarters of complexes have now the first correct prediction under 125 rather than 219. This improvement can also be observed with an increase on the percentage of complexes with correct predictions under the first 100 and 200 ranked contacts.

From the results, it appears that increasing the similarity restriction past 10%, in general, does not improve the predictive capabilities. If anything, it seems to impair it as more complexes and homologs are lost to the stricter constraints.

Although a common rule-of-thumb is that two sequences should be homologous if they share an identity of 30% or more, it has been shown that statistically significant homologs can share less than 20% identity, and that identity by itself is not a good measure of homology, but rather a proxy for evolutionary distance on already assumed homologous sequences [Pea13]. The BLAST search procedure coupled with the UniProt database, which was used to obtain the homologs, is quite reliable at inferring homology [Pea13]. So this emerges as a test of how compensating is the trade-off between having less distant homologs and less sequences overall.

The results support the idea that, in fact, relevant information is lost by rejecting sequences under 30% identity, where 28 complexes cease to have sufficient homologs, with around 61% of the homolog pairs being rejected across the data, when compared to the unrestricted configuration.

It could be that percent identity is too insensitive, as it only considers strictly identical residues. The next subsection contains tests with percent similarity, which considers chemically similar amino-acids as matches, rather than strictly identical elements.

Similarity restriction	# training complexes
0%	98
10%	96
20%	91
30%	84
40%	70
50%	57
60%	40
70%	29

Table 4.2: Number of available complexes for several percent similarity thresholds.

4.1.2 Homolog percent similarity

Percent similarity is an alternative way to judge how similar two given sequences are. As explained in 2.1.1.1, it represents the percentage of amino-acids that are chemically similar between two aligned sequences. It is more sensible than percent identity, e.g., if two sequences in a 100-residue alignment share 10 identical residues, 20 different but chemically similar residues and 70 remaining dissimilar residues, then such sequences would be 10% percent identity similar and 30% percent similarity similar. Thus, it might not be so eager to discard homologs that could otherwise be pertinent.

The following similarities were then used to compute new datasets: 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%.

Likewise in the identity test, with these configurations, homologs originating from the same species were simply ordered by percent similarity (3.2.2.2).

The loss of complexes due to these restrictions is presented in table 4.2. It can be observed in table 4.2 that percent similarity causes less complexes to be rejected with its less restrictive constraints when compared to identity in table 4.1. For the same reason as identity based rejection, the mean/average metric is not reliable due to the loss of complexes.

In this case, unlike identity based rejection, prediction quality seems to increase along with the similarity threshold up to around 40%-50%. It also starts to decay later, past 60%, likely due to the increased loss of complexes. The averaging metrics display a similar pattern, all percentiles follow the trend of a decreasing first contact position that starts to worsen only past 50%.

In general, percent similarity based restriction sees improvement with bigger thresholds due to its increased sensitivity when compared to identity. Nonetheless, the top result in identity (around 10%) is comparable to the top results in similarity (40%-50%), where all manage to capture a first contact within 100 predictions around 70% of the cases. However, the similarity based restrictions do show additional improvements when observing the percentage of true contacts within the first 200 predictions, they capture 80% and 87% of the complexes, where the identity configuration reaches 77%. The

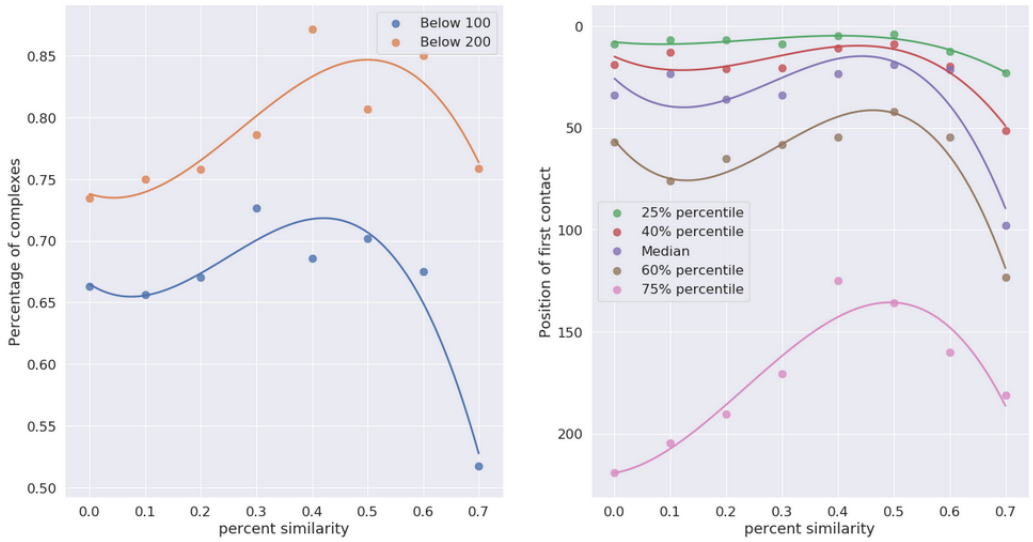

Figure 4.2: Comparison of different percent similarity configurations.

averaging metrics also indicate an improvement on the side of similarity.

4.2 Homolog Order Comparison

Here, the effect of having an homolog pairing strategy is measured. As explained in the development section under 3.2.2.2, it is the conflict that needs to be resolved when multiple homologs originating from the same species exist for both partner proteins, leaving an assumption to be made regarding which homologs to pair.

4.2.1 Ordering by a similarity measure

In order to test the impact of a similarity based ordering strategy, two datasets were issued: An ordered one, where any instances of multiple homologs were resolved by being ordered from the most similar to the least similar, and an unordered counterpart, equally configured but set to randomly shuffle and pair homologs from the same species.

Both datasets were configured to use all the homologs, that is, no similarity restriction was applied.

The metrics emerging from predictions done on either dataset configuration were compared side by side in a bar chart, resulting in graph 4.3.

From the comparison, ordering the homologs by identity appears to be more advantageous than pairing them randomly. The averaging metrics (percentiles and mean) appear to corroborate that idea, however, the most significant metric is the percentage of complexes with a correct prediction within 100 predictions, which also improved. This metric is used to choose the optimal model for predicting within each configuration, so models are generally adjusted to maximize it.

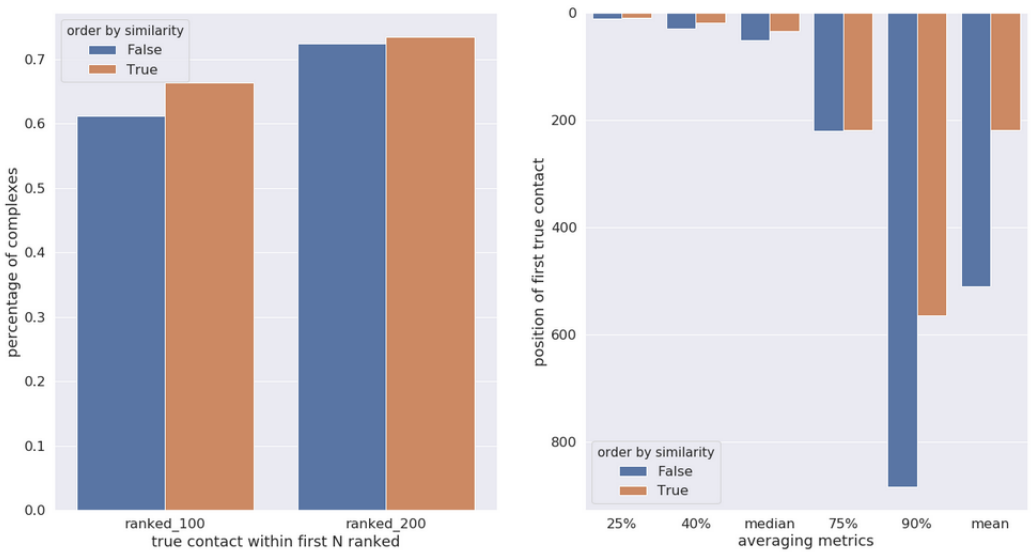

Figure 4.3: Performance comparison between randomly ordered homologs and identity ordered homologs.

4.2.2 Ordering by cluster

As explained in 3.2.2.2, another property that can help guide the pairing of multiple homologs is the cluster from which they have originated from when extracted from UniProt.

One way of measuring its effect is to restrict homologs to the first cluster (highest ranked) they were obtained from, as homologs from the same species can come from different clusters with different rankings. To test this, the dataset configuration that pairs all homologs based solely on identity was compared against a dataset that restricts the homologs to their first cluster, and only after proceeds to order them by identity as well.

The comparison graph can be seen at 4.4. In this case, there appears to be slight to no improvement when enforcing a first cluster restriction. Two out of the original 98 complexes present in the identity-only configuration are lost due to enforcing the first cluster only restriction. In the unrestricted case, 65 out of 98 complexes captured a true contact within the first 100 predictions (66.3%), in the cluster restricted case, with two rejected complexes, 64 out of 96 captured a true contact (66,6%), negligible change. A slight improvement was seen regarding the 200 prediction limit.

The averaging metrics seem to be somewhat stable, there is an improvement over the 75% percentile but a decrease in the mean, which could be motivated by the loss of the complexes.

4.3 Homolog Quantitative Comparison

Another factor to consider is the maximum number of allowed homolog sequences for any given complex. An upper limit of 2000 has been used as a default for performance considerations, however, lower limits of 1000, 500 and 125 were tested as well to measure

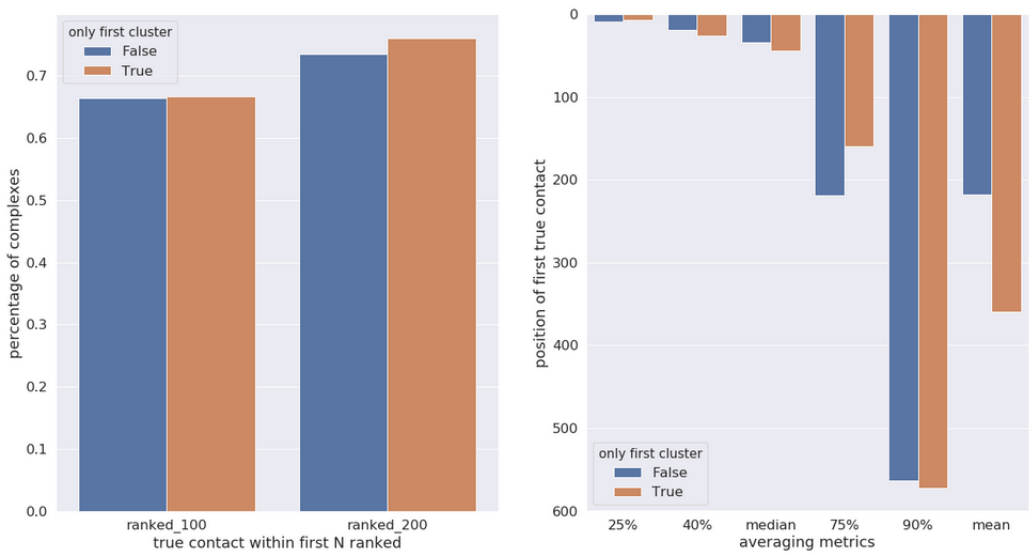

Figure 4.4: Performance comparison between identity ordered homologs and first cluster constrained ordered homologs.

its effect. The result can be seen in figure 4.5. Reducing the number of homologs does not

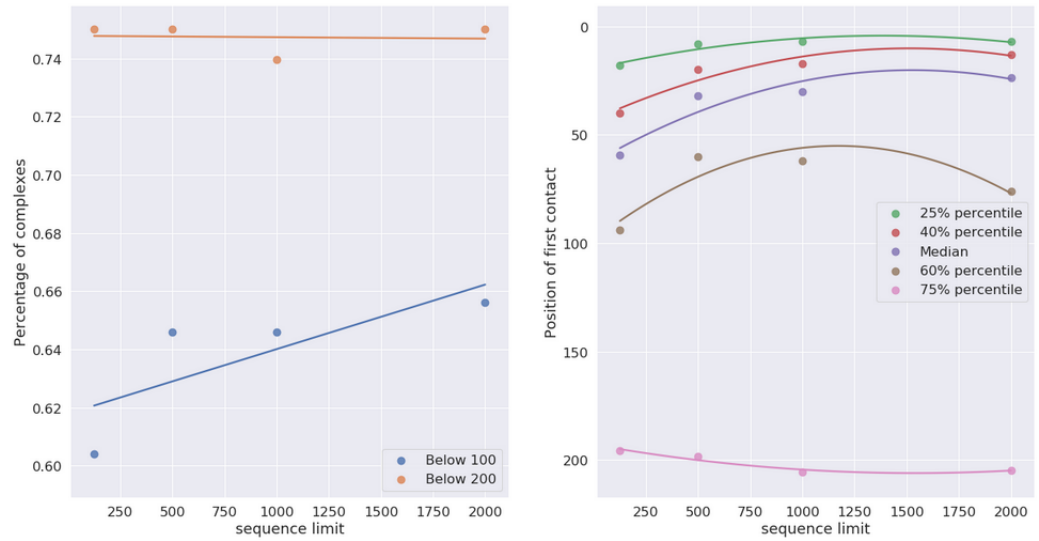

Figure 4.5: Performance comparison by using different limits.

have the expected negative effect, predictive capabilities do not change significantly, a slight loss can be seen in the percentage of complexes with a correct prediction among the first 100 and also on the averaging metrics. Nevertheless, not all complexes have a large number of homologs, and when similarity/identity restrictions are applied, the numbers get naturally reduced by being rejected.

4.4 Discussion and Test Results

The possible configurations with the data leads to a combinatorial explosion. It adds to the difficulty that each configuration may require up to two days of computation on the available equipment. Thus, tests were chosen in order to spot differences in particular ways of configuring the data. However, more tests can be done, especially by combining different configuration techniques, in favor of consolidating the effects of each technique and additionally seek for an optimal data setup.

The chosen configuration, perceived to be among the best, was the 40% percent similarity rejection strategy, explored in 4.1.2. The choice was due to being around a peak of improved predictive performances (graph 4.2), having a particularly good coverage of complexes with correct predictions below 200 (87%), a peak value for the 75% percentile and a reasonably good amount of complexes.

4.4.1 Test on the improved configuration

The data was then configured to ignore homologs below 40% similarity and pair those within the same species according to their similarity. This resulted in a set of 70 (out of 98) training complexes and 39 (out of 53) test complexes. From here, the training set was normalized and the same scaling factor was applied to the test set.

A Logistic Regression model was set to use the optimal hyperparameters found for this configuration during model building, for which there was a training score of 71% and a validation score of 69%, indicating neither overfitting nor underfitting.

The final Logistic Regression model produced good results, especially when considering the only comparable research, Krippahl, L., & Barahona, P. [KB15]. The classifier ranked a median of 19.0 for the position of the first correct prediction, an improvement over the article's 26.5. Interestingly, it also attained a remarkable mean of 77.0 for the first correct prediction across the test complexes. For 32 out of 39 test complexes (82%), there was a true contact among the first 100 predictions (top 100), and for 37 out of 39 (94.9%), within the first 200 (top 200). This last metric was also an improvement over the article's 89.2%.

However, part of the good results were owed to the use of Logistic Regression, compared to the Naïve Bayes classifier used in the article. To conclude this, a Naïve Bayes classifier was set up to run over this data and, in fact, did not outperform the classifier in the article, as it had a median of 34.0, captured 69.2% of the complexes with top 100 predictions and 84.6% with top 200 predictions.

A Random Forest classifier was also used on this set. To pick the best hyperparameters, the same cross-validation procedure over the training data was applied. Due to time constraints, a small grid of hyperparameters was defined: The number of trees was set to 500, the maximum tree depth could be either 1,3,5 or 8, and the minimum required

samples to create a split or to become a leaf node were both set to 10. After optimizing in cross-validation, a tree depth of 3 was chosen. The optimized model produced a training score of 71% and a validation score of 69%, indicating neither overfitting nor underfitting.

The final Random Forest model also had good results. It ranked a median of 16.0, better than the previous results, and correctly identified a true contact in 33 out of the 39 test complexes for the first 100 predictions (85%), which was also an improvement. The score for the first 200 predictions lowered to 34 out of 39 complexes (87%). The last 3 complexes (worst rating) had their first true contact ranked 492, 1097 and 4005, while Logistic Regression had 178, 750 and 844, and Naïve Bayes 270, 738 and 3781.

In a way, the results were surprising because the used models are relatively simple in comparison with the article, with approximately half the features being used. Also, no feature selection was employed.

4.4.2 Test on the baseline

From the baseline unrestricted configuration, the same tests were also performed. Naturally, for this test, all 98 train complexes and 53 test complexes are available. The data was set to this configuration and the pipeline followed the same procedure as described above. Both Regression and Forest models performed identically in the train and the validation sets after optimization, 66% and 68%, respectively.

For the Logistic Regression, the median of the first correct prediction was 29.0. For 38 out of 53 (71.7%) complexes, a true contact was captured within the first 100 predictions, and 41 out of 53 (77.4%) for the first 200 predictions. The restricted configuration appears to aid the performance on the test data. Improvements can also be seen on the averaging metrics, for example, the first 25% of the complexes had a true contact predicted under position 8.0, while the restricted had 4.5, for the first 75% it had 162.0, and the restricted had 45.0. Both the Random Forest and the Naïve Bayes classifiers performed worse on this baseline data configuration as well.

The comparison of the final models can be seen in the following table, 4.3. In fact, all

classifier	restriction	25%	median	75%	90%	mean	top 100	top 200
regression	yes	5.0	19.0	52.0	160.4	77.15	0.820	0.948
forest	yes	7.0	16.0	40.5	212.2	173.6	0.846	0.872
bayes	yes	7.0	34.0	145.5	219.8	178.6	0.692	0.846
regression	no	8.0	29.0	162.0	752.6	250.24	0.716	0.773
forest	no	12.0	38.0	125.0	614.6	282.64	0.717	0.830
bayes	no	16.0	65.0	204.0	449.2	214.8	0.566	0.736

Table 4.3: Comparison between testing with restricted data and unrestricted data, using quantiles, mean and percentage of complexes having at least one correct contact in the given top positions.

classifiers perform worse in the baseline configuration.

4.4.3 Feature importance

The following bar plots describe how important the features were for the classifiers in the improved data configuration, 4.6.

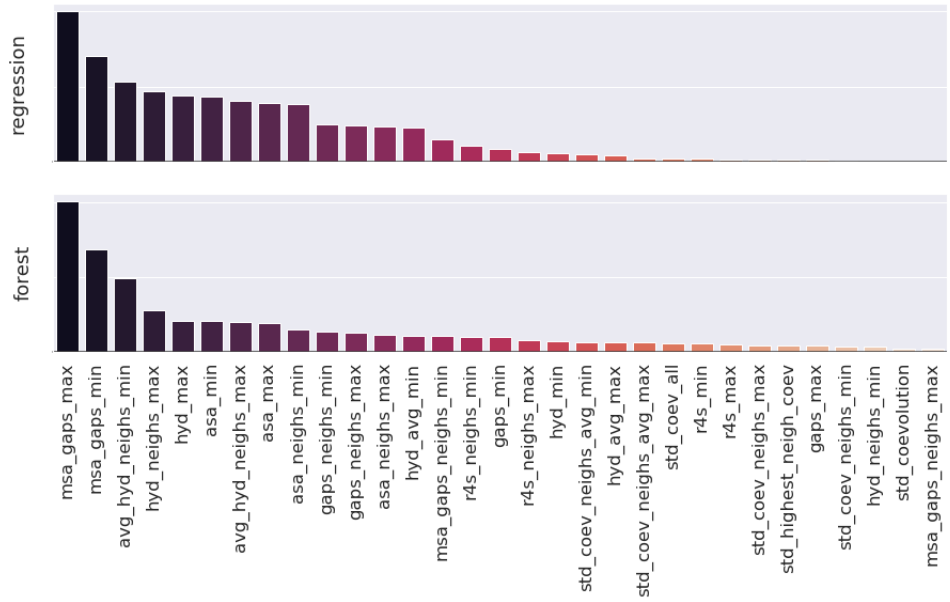

Figure 4.6: Feature importance for Logistic Regression and Random Forest for classifying in the tuned data.

Interestingly, the co-evolutive features did not have the expected importance. Nevertheless, it does not rule out their contribution. Measures based on gaps, hydrophobicity and conservation, which are still dependent on homology, ranked higher. It can be observed that gaps in the alignment have a big impact, most likely determining when it is not a contact. Important residues such as contacts should rarely be missing in any homologous protein.

The hydrophobicity value is also significant, as it gives a measure of the affinity that proximally close residues should have. In particular, the averaged hydrophobicity for the neighborhood throughout the homolog alignment. A contact vicinity should keep its hydrophobic properties even in different homologs.

Chapter 5

Conclusion and Future Work

The goal of predicting contacts between two protein molecules is an important one. Its resolution mainly depends on two sources of information: the structure of the interacting proteins, and the evolution of sequences homologous to the interacting proteins.

This thesis focused on optimizing the selection of homolog sequences, from which evolutionary information is dependent. From here, adding structure derived features can possibly improve the predictions, such as: contact propensities [Jha+10; KB15]; desolvation and electrostatic properties [AA+15]; secondary structure information [AA+15]; and residue depth and mobility [FZ10].

During project implementation, several challenges were met, especially during the data acquisition and preparation, which was not a trivial task (described in 3.2.1). The result was a total of 2.140.796 protein homolog sequences.

To extract knowledge from these sequences, they need to be arranged. For a given protein, its associated homologs must be aligned in a way that equivalent residues in terms of structure and functionality end up aligned in a one-to-one correspondence, this provides an outlook on the evolution of that particular protein. For a hint on co-evolving residues between two pairing proteins, which form a protein complex, homologs of both these proteins must be paired with each other whenever they originate from the same organism, such that together they represent an homologous protein complex sequence.

During inspection of the homolog data, it was noticed that several non-identical homologs existed for same organisms, and additionally, significantly dissimilar homolog sequences were being admitted into the data. As a result, questions were raised regarding the impact that different homolog filtering and ordering strategies would have in the predictive results.

Thus, various ways of filtering and organizing the homolog data were devised, along

with a mechanism to keep track of each different configuration and test them by applying machine learning independently.

Though further testing can be done, improvements were observed when restricting the quality of the homolog sequences using a particular measure of homolog similarity. Leading to the confirmation that optimizing the homolog data is important.

Future Work

It was observed that the configured co-evolutive features in particular, scored low on feature importance. While that does not rule out their contribution, there is the possibility that the used configuration is non-ideal and thus, could be dampening the effects of homolog ordering strategies, which mainly affect coevolutionary measures. As such, additional co-evolutive features based on different methods could be tested in the future.

Some homolog sequences have enzymatic information, this could be analyzed to check its usefulness in potentially guiding the ordering mechanism responsible for pairing multiple same organism homologs on either of the pairing proteins.

Regarding the conservation features, the used method had to be limited to compute a maximum of 200 homolog sequences due to arithmetic underflow, thus, an alternative solution can be found to take advantage of the full set of available homologs.

Finally, the next logical step would be to test additional machine learning classifiers on optimized data configurations, such as the Neural Networks and SVMs that were initially considered, with a new goal of searching for ideal classifiers for this problem.

Bibliography

[AC16] B. Adhikari and J. Cheng. "Protein residue contacts and prediction methods." In: *Methods in Molecular Biology*. 2016. ISBN: 1940-6029 (Electronic)\r1064-3745 (Linking). DOI: 10.1007/978-1-4939-3572-7_24.

[Alp10] E Alpaydin. *Introduction to Machine Learning, 2nd edn. Adaptive Computation and Machine Learning*. 2010.

[AA+15] T. T. Aumentado-Armstrong, B. Istrate, and R. A. Murgita. "Algorithmic approaches to protein-protein interaction site prediction." In: *Algorithms for Molecular Biology* 10.1 (2015), pp. 1–21. ISSN: 17487188. DOI: 10.1186/s13015-015-0033-9.

[Bat19] A. Bateman. "UniProt: A worldwide hub of protein knowledge." In: *Nucleic Acids Research* (2019). ISSN: 13624962. DOI: 10.1093/nar/gky1049.

[BB12] J. Bergstra and Y. Bengio. "Random search for hyper-parameter optimization." In: *Journal of Machine Learning Research* (2012). ISSN: 15324435.

[Ber+00] H. M. Berman, J Westbrook, Z Feng, G Gilliland, T. N. Bhat, H Weissig, I. N. Shindyalov, and P. E. Bourne. "The Protein Databank." In: *Nucleic Acids Research* (2000). DOI: 10.1002/0470020571.ch10.

[BR03] M. J. Betts and R. B. Russell. "Amino acid properties and consequences of substitutions." In: *Bioinformatics for geneticists* 317 (2003), p. 289.

[Bis+03] K. M. Biswas, D. R. DeVido, and J. G. Dorsey. *Evaluation of methods for measuring amino acid hydrophobicities and interactions*. 2003. DOI: 10.1016/S0021-9673(03)00182-1.

[Bra+99] C. I. Branden et al. *Introduction to protein structure*. Garland Science, 1999.

[Bur08] F. J. Burkowski. *Structural bioinformatics: an algorithmic approach*. CRC Press, 2008.

[Cav+03] L. Cavallo, J. Kleinjung, and F. Fraternali. "POPS: A fast algorithm for solvent accessible surface areas at atomic and residue level." In: *Nucleic Acids Research* (2003). ISSN: 03051048. DOI: 10.1093/nar/gkg601.

[CP09] S. Chakrabarti and A. R. Panchenko. "Coevolution in defining the functional specificity." In: *Proteins: Structure, Function and Bioinformatics* (2009). ISSN: 08873585. DOI: 10.1002/prot.22239.

[Coc+09] P. J. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, and M. J. De Hoon. "Biopython: Freely available Python tools for computational molecular biology and bioinformatics." In: *Bioinformatics* (2009). ISSN: 13674803. DOI: 10.1093/bioinformatics/btp163. arXiv: arXiv:1011.1669v3.

[Cub+05] M. V. Cubellis, F. Cailliez, and S. C. Lovell. "Secondary structure assignment that accurately reflects physical and evolutionary characteristics." In: *BMC Bioinformatics* (2005). ISSN: 14712105. DOI: 10.1186/1471-2105-6-S4-S8.

[Day+78] M. Dayhoff, R. Schwartz, and B. Orcutt. *A model of evolutionary change in proteins*. 1978. DOI: 10.1.1.145.4315.

[EC12] J. Eickholt and J. Cheng. "Predicting protein residue-residue contacts using deep networks and boosting." In: *Bioinformatics* (2012). ISSN: 13674803. DOI: 10.1093/bioinformatics/bts598.

[Eke+13] M. Ekeberg, C. L??vkvist, Y. Lan, M. Weigt, and E. Aurell. "Improved contact prediction in proteins: Using pseudolikelihoods to infer Potts models." In: *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* 87.1 (2013), pp. 1–19. ISSN: 15393755. DOI: 10.1103/PhysRevE.87.012707. arXiv: 1211.1281.

[Ezk+09] I. Ezkurdia, L. Bartoli, P. Fariselli, R. Casadio, A. Valencia, and M. L. Tress. "Progress and challenges in predicting protein-protein interaction sites." In: *Briefings in Bioinformatics* (2009). ISSN: 14675463. DOI: 10.1093/bib/bbp021.

[Fet95] J. S. Fetrow. "Omega loops: nonregular secondary structures significant in protein function and stability." In: *The FASEB Journal* 9.9 (1995), pp. 708–717.

[FZ10] S. Fiorucci and M. Zacharias. "Prediction of protein-protein interaction sites using electrostatic desolvation profiles." In: *Biophysical journal* 98.9 (2010), pp. 1921–1930. ISSN: 0006-3495.

[Gla+01] F. Glaser, D. M. Steinberg, I. A. Vakser, and N. Ben-Tal. "Residue frequencies and pairing preferences at protein–protein interfaces." In: *Proteins: Structure, Function, and Bioinformatics* 43.2 (2001), pp. 89–102. ISSN: 1097-0134.

[Gon+13] A. J. González, L. Liao, and C. H. Wu. "Prediction of contact matrix for protein-protein interaction." In: *Bioinformatics* 29.8 (2013), pp. 1018–1025. ISSN: 13674803. DOI: 10.1093/bioinformatics/btt076.

[Gro10] M. M. Gromiha. *Protein bioinformatics: from sequence to function*. Academic Press, 2010.

[Hal+02] I. Halperin, B. Ma, H. Wolfson, and R. Nussinov. "Principles of docking: An overview of search algorithms and a guide to scoring functions." In: *Proteins: Structure, Function, and Bioinformatics* 47.4 (2002), pp. 409–443.

[HM03] T. Hamelryck and B. Manderick. "PDB file parser and structure class implemented in Python." In: *Bioinformatics* 19.17 (2003), pp. 2308–2310. ISSN: 13674803. DOI: 10.1093/bioinformatics/btg299.

[HF04] M. Heinig and D. Frishman. "STRIDE: a web server for secondary structure assignment from known atomic coordinates of proteins." In: *Nucleic Acids Research* 32.suppl_2 (2004), W500–W502. DOI: 10.1093/nar/gkh429. eprint: /oup/backfile/content{_}public/journal/nar/32/suppl{_}2/10.1093/nar/gkh429/3/gkh429.pdf. URL: http://dx.doi.org/10.1093/nar/gkh429.

[HH92] S. Henikoff and J. G. Henikoff. "Amino acid substitution matrices from protein blocks." In: *Proceedings of the National Academy of Sciences* 89.22 (1992), pp. 10915–10919.

[Ho18] B. Ho. *pdbremix: Library to analyze protein structures and protein simulations*. https://github.com/boscoh/pdbremix. 2018.

[Hop+14] T. A. Hopf, C. P. Schärfe, J. P. Rodrigues, A. G. Green, O. Kohlbacher, C. Sander, A. M. Bonvin, and D. S. Marks. "Sequence co-evolution gives 3D contacts and structures of protein complexes." In: *eLife* (2014). ISSN: 2050084X. DOI: 10.7554/eLife.03430. arXiv: 1405.0929.

[HN93] S. Hubbard and T. J. NACCESS. "Computer program." In: *Department of Biochemistry and Molecular Biology, University College, London* (1993).

[Hun07] J. D. Hunter. "Matplotlib: A 2D graphics environment." In: *Computing in Science and Engineering* (2007). ISSN: 15219615. DOI: 10.1109/MCSE.2007.55. arXiv: 0402594v3 [arXiv:cond-mat].

[Jan+03] J. Janin, K. Henrick, J. Moult, L. T. Eyck, M. J. Sternberg, S. Vajda, I. Vakser, and S. J. Wodak. "CAPRI: A critical assessment of PRedicted interactions." In: *Proteins: Structure, Function and Genetics* (2003). ISSN: 08873585. DOI: 10.1002/prot.10381.

[Jha+10] A. N. Jha, S. Vishveshwara, and J. R. Banavar. "Amino acid interaction preferences in proteins." In: *Protein Science* (2010). ISSN: 09618368. DOI: 10.1002/pro.339.

[JT10] F. Johansson and H. Toh. "A comparative study of conservation and variation scores." In: *BMC Bioinformatics* (2010). ISSN: 14712105. DOI: 10.1186/1471-2105-11-388.

[Jon+12] D. T. Jones, D. W. Buchan, D. Cozzetto, and M. Pontil. "PSICOV: Precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments." In: *Bioinformatics* (2012). ISSN: 13674803. DOI: 10.1093/bioinformatics/btr638. arXiv: /bioinformatics.oxfordjournals.org/content/suppl/2011/11/29/btr638.DC1/target.txt [http:].

[KS83] W. Kabsch and C. Sander. "Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features." In: *Biopolymers* (1983). ISSN: 10970282. DOI: 10.1002/bip.360221211. arXiv: 83/122577-6 [0006-3525].

[Kam+13] H. Kamisetty, S. Ovchinnikov, and D. Baker. "Assessing the utility of coevolution-based residue-residue contact predictions in a sequence- and structure-rich era." In: *Proceedings of the National Academy of Sciences* (2013). ISSN: 0027-8424. DOI: 10.1073/pnas.1314045110.

[KB15] L. Krippahl and P. Barahona. "Protein docking with predicted constraints." In: *Algorithms for Molecular Biology* 10.1 (2015), p. 9.

[KD82] J. Kyte and R. F. Doolittle. "A simple method for displaying the hydropathic character of a protein." In: *Journal of Molecular Biology* (1982). ISSN: 00222836. DOI: 10.1016/0022-2836(82)90515-0. arXiv: arXiv:1407.5140v1.

[LR71] B. Lee and F. M. Richards. "The interpretation of protein structures: Estimation of static accessibility." In: *Journal of Molecular Biology* (1971). ISSN: 00222836. DOI: 10.1016/0022-2836(71)90324-X.

[Lew98] D. D. Lewis. "Naive (Bayes) at forty: The independence assumption in information retrieval." In: *European conference on machine learning*. Springer, 1998, pp. 4–15.

[LL09] J Li and Q Liu. "'Double water exclusion': a hypothesis refining the O-ring theory for the hot spots at protein interfaces." In: *Bioinformatics* 25 (2009). DOI: 10.1093/bioinformatics/btp058. URL: https://doi.org/10.1093/bioinformatics/btp058.

[LW02] a Liaw and M Wiener. "Classification and Regression by randomForest." In: *R news* 2.December (2002), pp. 18–22. ISSN: 16093631. DOI: 10.1177/154405910408300516. arXiv: 1609-3631.

[MG08] H. Madaoui and R. Guerois. "Coevolution at protein complex interfaces can be detected by the complementarity trace with important impact for predictive docking." In: *Proceedings of the National Academy of Sciences* 105.22 (2008), pp. 7708–7713. ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0707032105. URL: http://www.pnas.org.ezproxy.lib.monash.edu.au/content/105/22/7708{\%}5Cnhttp://www.ncbi.nlm.nih.gov/pubmed/

18511568{\%}5Cnhttp://www.pnas.org.ezproxy.lib.monash.edu.au/content/105/22/7708.full.pdf{\%}5Cnhttp://www.pnas.org.ezproxy.lib.monash.edu.au/content/105/22/.

[Mar09] S. Marsland. "Machine Learning: An Algorithmic Perspective." In: (2009).

[May+04] I Mayrose, D Graur, N Ben-Tal, and T Pupko. "Comparison of site-specific rate-inference methods for protein sequences: ..." In: *Molecular biology and evolution* (2004).

[MN98] A. McCallum and K. Nigam. "A comparison of event models for naive bayes text classification." In: *AAAI-98 workshop on learning for text categorization*. Vol. 752. 1. Citeseer, 1998, pp. 41–48.

[McK11] W. McKinney. "pandas: a Foundational Python Library for Data Analysis and Statistics." In: *Python for High Performance and Scientific Computing* (2011).

[Mih+08] J. Mihel, M. Šikić, S. Tomić, B. Jeren, and K. Vlahoviček. *PSAIA - Protein structure and interaction analyzer*. 2008. DOI: 10.1186/1472-6807-8-21.

[Mit16] S. Mitternacht. "FreeSASA: An open source C library for solvent accessible surface area calculations." In: *F1000Research* (2016). ISSN: 2046-1402. DOI: 10.12688/f1000research.7931.1. arXiv: 1601.06764.

[Mor+11] F. Morcos, A. Pagnani, B. Lunt, A. Bertolino, D. S. Marks, C. Sander, R. Zecchina, J. N. Onuchic, T. Hwa, and M. Weigt. "Direct-coupling analysis of residue coevolution captures native contacts across many protein families." In: *Proceedings of the National Academy of Sciences* 108.49 (2011), E1293–E1301. ISSN: 0027-8424. DOI: 10.1073/pnas.1111471108. arXiv: 1110.5223. URL: http://www.pnas.org/cgi/doi/10.1073/pnas.1111471108.

[Mor+07] I Moreira, P Fernandes, and M Ramos. "Hot spots–A review of the protein-protein interface determinant amino-acid residues." In: *Proteins* 68 (2007). DOI: 10.1002/prot.21396. URL: https://doi.org/10.1002/prot.21396.

[NW70] S. B. Needleman and C. D. Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins." In: *Journal of Molecular Biology* (1970). ISSN: 00222836. DOI: 10.1016/0022-2836(70)90057-4.

[OR07] Y. Ofran and B. Rost. "Protein-protein interaction hotspots carved into sequences." In: *PLoS computational biology* 3.7 (2007), e119.

[Oli07] T. E. Oliphant. "SciPy: Open source scientific tools for Python." In: *Computing in Science and Engineering* (2007). ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.58.

[Ovc+14] S. Ovchinnikov, H. Kamisetty, and D. Baker. "Robust and accurate prediction of residue–residue interactions across protein interfaces using evolutionary information." In: *eLife* 3 (2014). Ed. by B. Roux, e02030. ISSN: 2050-084X. DOI: 10.7554/eLife.02030. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4034769/.

[Pea13] W. R. Pearson. "Selecting the right similarity-scoring matrix." In: *Current Protocols in Bioinformatics* (2013). ISSN: 1934340X. DOI: 10.1002/0471250953.bi0305s43. arXiv: NIHMS150003.

[Ped+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* (2011). ISSN: 15324435. DOI: 10.1007/s13398-014-0173-7.2. arXiv: 1201.0490.

[PG01] J. Pei and N. V. Grishin. "AL2CO: Calculation of positional conservation in a protein sequence alignment." In: *Bioinformatics* (2001). ISSN: 13674803. DOI: 10.1093/bioinformatics/17.8.700.

[Pup+02] T Pupko, R Bell, I Mayrose, F Glaser, and N Ben-Tal. "Rate4Site: an algorithmic tool for the identification of functional regions in proteins by surface mapping of evolutionary determinants within their homologues." In: *Bioinformatics* 18 (2002). DOI: 10.1093/bioinformatics/18.suppl{_}1.S71. URL: https://doi.org/10.1093/bioinformatics/18.suppl{_}1.S71.

[Ras14] S. Raschka. *About Feature Scaling and Normalization*. 2014.

[RM17] S. Raschka and V. Mirjalili. *Python Machine Learning - Second Edition*. 2017. ISBN: 9781787125933. DOI: 10.1016/0025-5408(96)80018-3.

[Rus14] D. J. Russell. *Multiple Sequence Alignment Methods*. Vol. 1079. 2014, p. 287. ISBN: 9781627036450. DOI: 10.1007/978-1-62703-646-7. URL: http://link.springer.com/10.1007/978-1-62703-646-7{\%}5Cnhttp://www.ncbi.nlm.nih.gov/pubmed/24170407.

[SR15] T. Saito and M. Rehmsmeier. "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets." In: *PLoS ONE* (2015). ISSN: 19326203. DOI: 10.1371/journal.pone.0118432.

[See+14] S. Seemayer, M. Gruber, and J. Söding. "CCMpred—fast and precise prediction of protein residue–residue contacts from correlated mutations." In: *Bioinformatics* 30.21 (2014), pp. 3128–3130. DOI: 10.1093/bioinformatics/btu500. eprint: /oup/backfile/content_public/journal/bioinformatics/30/21/10.1093_bioinformatics_btu500/2/btu500.pdf. URL: http://dx.doi.org/10.1093/bioinformatics/btu500.

[SR73] A. Shrake and J. A. Rupley. "Environment and exposure to solvent of protein atoms. Lysozyme and insulin." In: *Journal of Molecular Biology* (1973). ISSN: 00222836. DOI: 10.1016/0022-2836(73)90011-9.

[SH14] F. Sievers and D. G. Higgins. "Clustal Omega, Accurate Alignment of Very Large Numbers of Sequences." In: *Multiple Sequence Alignment Methods*. Ed. by D. J. Russell. Totowa, NJ: Humana Press, 2014, pp. 105–116. ISBN: 978-1-62703-646-7. DOI: 10.1007/978-1-62703-646-7{_}6. URL: https://doi.org/10.1007/978-1-62703-646-7{_}6.

[Sie+11] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding, J. D. Thompson, and D. G. Higgins. "Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega." In: *Molecular Systems Biology* 7.1 (2011). URL: http://msb.embopress.org/content/7/1/539.abstract.

[Suz+07] B. E. Suzek, H. Huang, P. McGarvey, R. Mazumder, and C. H. Wu. "UniRef: comprehensive and non-redundant UniProt reference clusters." In: *Bioinformatics* 23.10 (2007), pp. 1282–1288. DOI: 10.1093/bioinformatics/btm098. eprint: /oup/backfile/content{_}public/journal/bioinformatics/23/10/10.1093/bioinformatics/btm098/2/btm098.pdf. URL: http://dx.doi.org/10.1093/bioinformatics/btm098.

[Tou+15] W. G. Touw, C. Baakman, J. Black, T. A. H. te Beek, E Krieger, R. P. Joosten, and G. Vriend. "A series of PDB-related databanks for everyday needs." In: *Nucleic Acids Research* 43.D1 (2015), pp. D364–D368. ISSN: 0305-1048. URL: http://dx.doi.org/10.1093/nar/gku1028.

[TS98] G Trinquier and Y. H. Sanejouand. "Which effective property of amino acids is best preserved by the genetic code?" In: *Protein Engineering Design and Selection* (1998). ISSN: 1741-0126. DOI: 10.1093/protein/11.3.153.

[Tun+09] N. Tuncbag, A. Gursoy, and O. Keskin. "Identification of computational hot spots in protein interfaces: combining solvent accessibility and inter-residue potentials improves the accuracy." In: *Bioinformatics* 25.12 (2009), pp. 1513–1520. DOI: 10.1093/bioinformatics/btp240.

[Van+11] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. "The NumPy array: A structure for efficient numerical computation." In: *Computing in Science and Engineering* (2011). ISSN: 15219615. DOI: 10.1109/MCSE.2011.37. arXiv: 1102.1523.

[Vre+15] T. Vreven, I. H. Moal, A. Vangone, B. G. Pierce, P. L. Kastritis, M. Torchala, R. Chaleil, B. Jiménez-García, P. A. Bates, J. Fernandez-Recio, A. M. Bonvin, and Z. Weng. "Updates to the Integrated Protein-Protein Interaction Benchmarks: Docking Benchmark Version 5 and Affinity Benchmark Version 2." In: *Journal*

of Molecular Biology (2015). ISSN: 10898638. DOI: 10.1016/j.jmb.2015.07.016. arXiv: 15334406.

[Wan+12] L. Wang, Z. P. Liu, X. S. Zhang, and L. Chen. "Prediction of hot spots in protein interfaces using a random forest model with hybrid features." In: *Protein Engineering, Design and Selection* 25.3 (2012), pp. 119–126. ISSN: 17410126. DOI: 10.1093/protein/gzr066.

[Zho+17] T. Zhou, S. Wang, and J. Xu. "Deep learning reveals many more inter-protein residue-residue contacts than direct coupling analysis." In: *bioRxiv* (2017). URL: http://biorxiv.org/content/early/2017/12/29/240754.abstract.

Webography

[KDn13] KDnuggets. *Languages for Analytics / Data Mining / Data Science*. Sept. 2013. URL: https://www.kdnuggets.com/polls/2013/languages-analytics-data-mining-data-science.html (visited on 07/05/2018).

[Pro] *What are proteins and what do they do?* URL: https://ghr.nlm.nih.gov/primer/howgeneswork/protein (visited on 05/16/2017).