UNIVERSITY OF CALIFORNIA
SANTA CRUZ

RESIDUE-RESIDUE CONTACT PREDICTIONS USING NEURAL NETWORKS AND SELECTED STATISTICS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

BIOINFORMATICS

by

George Shackelford

December 2010

The Dissertation of George Shackelford
is approved:

Professor Kevin Karplus, Chair

Professor Richard Hughey

Professor Herbert Lee

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Residue-Residue Contact Predictions Using Neural Networks and Selected Statistics

by

George Shackelford

Residue-residue contact prediction involves predicting that two residues in a protein sequence, separated by six or more residues along the sequence, will be close to each other when the protein folds into its structure. Interest in contact prediction is prompted by the challenge of protein structure prediction which has been a long-standing problem in molecular biology. With the sequencing of multiple genomes, the number of protein sequences in the non-redundant sequence database has grown to over six million. While the number of known structures has also grown, there are only about forty thousand distinct structures known. As experimental methods are expensive and time-consuming and as the gap between known sequences and known structures grows, the need for accurate structure prediction has increased.

The Critical Assessment of Techniques for Protein Structure Prediction (CASP), a bi-annual experiment in structure prediction, currently has two difficulty levels for tertiary structure prediction. The easier, template-based modeling, uses known structures for sequences similar to the target that can be used to build the prediction models. The second, free modeling, deals with making predictions where there are no templates

known. This second category is much more difficult and needs ways to reduce the combinatorial explosion of the tertiary search space.

With free-modeling predictions in mind, CASP6 introduced residue-residue contact prediction as a new category for predictions. I developed a neural network which takes 449 inputs and predicts contacts, and my predictions for CASP7 were assessed as the best. This thesis will cover the process of selecting the inputs from types of local structure predictions, and paired statistics, including one novel statistic.

I will address issues with the evaluation of contact predictions and propose an alternative accuracy measure. A comparison to contact predictions from the best group's free-modeling tertiary predictions suggests that contact predictions could be useful in improving free-modeling tertiary predictions.

For CASP8, the assessors found that no group's predictions were considered outstanding, including my own. I will show evidence that a lack of an adequate number of sequences in the multiple sequence alignments is a significant limiting factor in my effort to generate good predictions. I conclude with a discussion of possible research to improve future contact predictions.

To my mother,

Lib Shackelford,

and my late father,

John Cooper Shackelford

and especially to my wife and coach,

Marsha Hudson,

Acknowledgments

I wish to thank my committee, and my adviser, Kevin Karplus.

Chapter 1

Introduction

This thesis presents my research in residue-residue contact prediction. The work is broken into four parts: one on basic information so the reader will understand the material in and motivation for this thesis, a second on neural networks and source of inputs, a third on evaluation methods, and the fourth on results and conclusions.

Every two years since 1994 a three month long international experiment is conducted called the Community Wide Experiment on the Critical Assessment of Techniques for Protein Structure Prediction, or CASP. This involves the release of sequence information for proteins whose structure is expected to be solved by the end of the experiment. Therefore, the sequences are representative, not of any particular class of proteins, but simply a sample of those proteins whose structures are currently being determined. The participants submit structure predictions in various categories shortly after receiving the targeted sequence. The category called residue-residue contact prediction was introduced less than a decade ago. The organizers are hoping for results

from these predictions that could help predict the full structure for the most difficult targets in the experiment.

The residue-residue contact predictions I submitted for CASP7 were assessed as the best. Typically a student is not expected to achieve this. How I achieved this result is at the heart of this thesis.

Chapter 1 explains what proteins are, the hierarchy of their structures, and what experiments determine protein structures and how those experiments have significant limitations. Chapter 2 covers predictions of protein structural elements including the entire structure, and Chapter 3 explains residue-residue contact predictions in particular.

Chapter 4 describes in detail neural networks, a machine learning technique for prediction. I discuss choices I made concerning the development of the software I used, challenges handling large amounts of data, as well as how formatting of the data could affect the accuracy. I start Chapter 5 by listing statistics and local structure predictions I test later as inputs for a neural network. I finish with a discussion of the ways the data can be formatted for input. The actual source of data I use is discussed in Chapter 6 including the size of *multiple sequence alignments* and *thinning* which turn out to be significant. How I evaluate the results of experiments is covered in Chapter 7.

Finally, I start to report results as I evaluate individual statistics in Chapter 8. Chapter 9 offers an interlude with preliminary predictions and results for CASP6 in 2004. The research discussion continues in Chapter 10 as I start matching statistics together. Chapter 11 covers the addition of local structure predictions as inputs leading

to the new neural network, predictions for CASP7, and the excellent results of that endeavor. Chapter 12 follows with post-CASP7 research leading to a two-stage predictor and CASP8 in 2008. The final chapter, Chapter 13, concludes with the disappointing aftermath of CASP8 and future research that may solve problems and improve future predictions.

The rest of this chapter will provide background material that will show primary, secondary, and tertiary aspects of the protein structure; a basic classification of the secondary structure; how protein structure can be determined by experimentation; and the need for protein structure prediction.

1.1 Proteins and their Structure

Proteins are the workhorses of our bodies. For example, the interactions of proteins build the cells of our bodies, regulate our metabolism, carry oxygen to our muscles, and drive the cognitive functions of our brains. In the earlier part of the previous century, we developed drugs to fight invasive organisms that caused disease. But many diseases are caused by malfunctions in the functions of proteins, and these have proven harder to treat because of the complexity of the proteins' functions. Knowing the structures of the proteins aids in the discovery of protein functions and interactions. As we better understand the functions and interactions, we are able to develop new drugs and treatments. Consequentially, there is great interest in determining the structure of proteins.

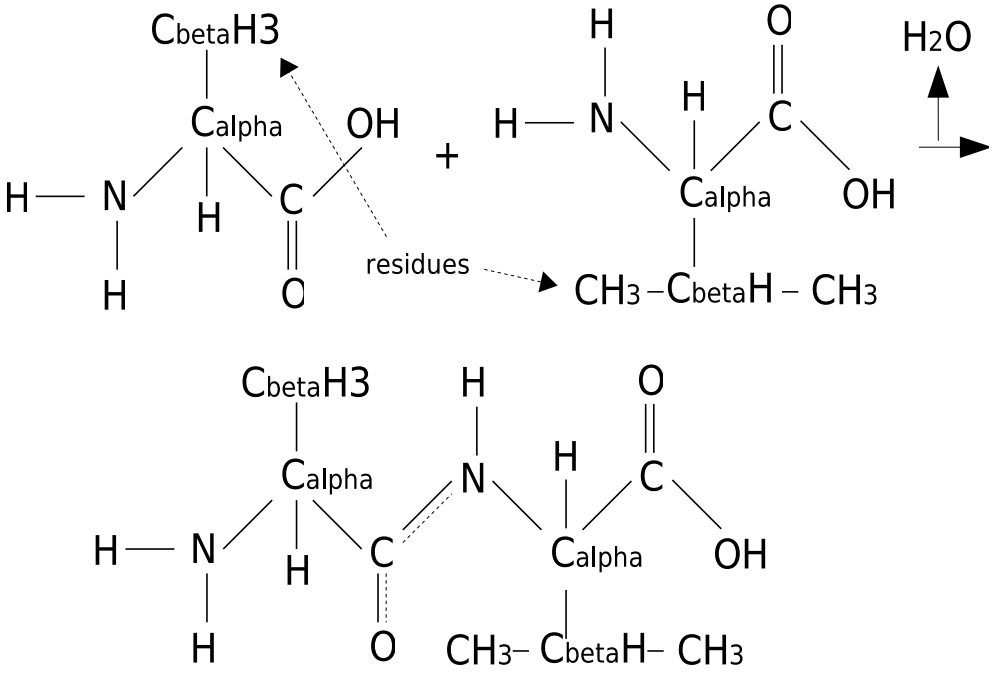


Figure 1.1: This shows two amino acids, alanine and valine, joining to form a simple peptide backbone and releasing a single water molecule. Note the sharing of bonds in the backbone resulting in the restriction of the omega angle between the second carbon and the nitrogen to either the *cis* or *trans* configuration.

A protein is primarily composed of a sequence of amino acids. Amino acids are distinguished by a central carbon atom, the alpha carbon, which has a carboxyl group on one bond, a hydrogen atom on another, an amino group on the third, and a group known as the side chain on the fourth bond (See Figure 1.1). The genes in our genome contain the coding of these sequences and the coding only specifies twenty different amino acids. The side chains are what distinguish between the different amino acids.

The amino acids are joined together through chemical bonds called covalent bonds to form the peptide backbone. The sequence of the amino acids and the backbone

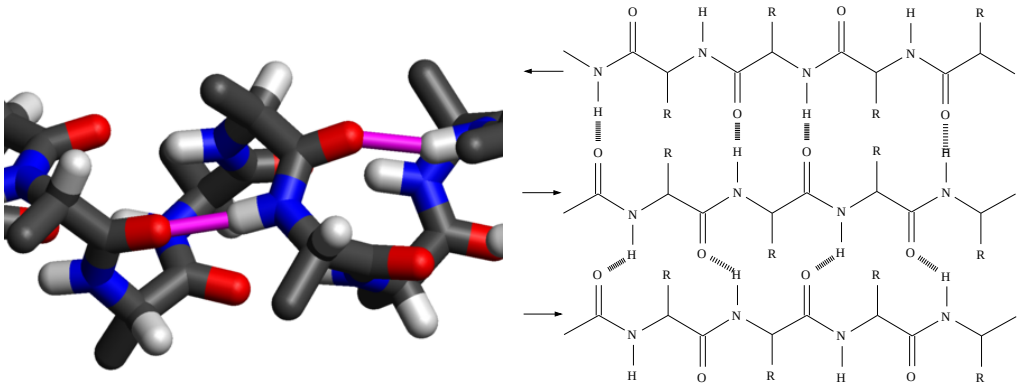
they form are called the *primary structure*. We currently know more than six million protein sequences across a wide variety of organisms. Covalent bonds are strong, with a binding energy of 30 kilocalories per mol of bonds [1].

This primary structure (after possible adjustments by other proteins or non-coding RNA) contains the information for the final structure. If there were no flexibility in the covalent bonds, then the final structure would be simple to determine but would not support the functions needed for life.

But there is flexibility in the covalent bonds of the backbone, described by three torsion angles per residue, the phi, psi, and omega angles. The bond between two atoms can act like an axle with one of the atoms rotating like a wheel; the amount of rotation is called the torsion angle. The first, between the N and the C_α is called the ϕ angle. The second, ψ, is between C_α and the C (carboxyl) atom. Finally, the third, ω, is between the carbon atom and the nitrogen of the next amino acid. The ω angle is energetically restricted to near 0° (*cis*) or near 180° (*trans*). The ω angle is almost always *trans*; I mention ω only in passing and it will not be discussed again. The other two are more flexible but have some steric restrictions [74]. Because of the flexibility of these angles, the chain can fold up into a final three-dimensional structure.

The torsion angles are considered *local structure properties* and define the *secondary structure*, since the structure involves only one amino acid and one or both of its neighbors. One local property is the widely used classification for the secondary structure: the α helix, the β strand, and the loop [1].

The final class of structure we will consider is the *tertiary structure*. The two



(a) Alpha helix (b) Beta strands

Figure 1.2: This figure shows the structure for an alpha helix and a beta strand, and how they typically form hydrogen bonds that are the basis for the tertiary structure. The alpha helix is shown as a 3-dimensional structure to illustrate the hydrogen bonds (in smaller-diameter pink cylinders) that hold it together as a helical structure. The beta-strands with their hydrogen bonds can be shown as a flat structure. Beta strands typically bond together to form *beta sheets*. The arrows show the direction along the backbone; pairings between strands are either in the same direction (*parallel*) or opposite directions (*anti-parallel*).

main forces that form the tertiary structure are hydrogen bonds and hydrophobicity.

Hydrogen bonds occur when a hydrogen with a covalent bond to an electronegative atom exhibits a partial positive charge that in turn attracts another electronegative atom. The common electronegative atoms in organic molecules are the nitrogen and oxygen atoms. Figure 1.2 shows the most common hydrogen bonds: a backbone oxygen is bonded to the hydrogen which is covalently bonded to a backbone nitrogen. Hydrogen bonds are chemically weak, with a binding energy of about 1.5 kilocalories per mol [1]. But the bonds make up for the weakness in number; nearly all backbone $N-H$'s are in hydrogen bonds.

Patterns of backbone hydrogen bonds define the α helix and β strands. The α helix occurs when the hydrogen of the nitrogen of one residue is bonded to the oxygen of

the fourth residue along the chain as shown in Figure 1.2. A series of such bonds forms a helical structure. The α helices involve about 38% of the residues in proteins [21]. The β strand is defined by bonds between it and another β strand (see Figure 1.2). Any residues not involved in either a helix or a strand are considered part of a loop.

Hydrophobicity as an effect is more properly a hydrophilic force. Water molecules can be characterized as having a partial positive charge on hydrogen side and a partial negative on the oxygen side. They form a natural affinity for each other and for polar and charged residues. The result is that the exterior of a globular protein is typically composed of hydrophilic residues while the interior is hydrophobic. (For the transmembrane part of membrane proteins, the reverse is true: the outer surface is exposed to the hydrophobic interior of the cell membrane, and therefore, the outer surface tends to be hydrophobic.)

1.2 Protein Structure Determination

Linus Pauling and his colleagues solved the first significant protein structure, hemoglobin [70]. They achieved this by trial-and-error analysis of the chemical bonds and properties, a process which is very slow and laborious. Fortunately, two widely used experimental techniques have since been developed to solve protein structures: x-ray crystallography and nuclear magnetic resonance (NMR).

The results of published protein structure determinations are publicly available in the Protein Database [6] or PDB. This provides the source of structures used for

statistical data used in training and testing (see Chapter 6).

Before discussing the two methods, we must acknowledge that there is a huge gap between known or putative protein sequences, and experimentally determined structures. The PDB contains fewer than sixty thousand structures, and due to sequencing of numerous genomes, the number of known or putative sequences is greater than six million. And the gap is only getting worse as sequencing genomes gets less expensive.

1.2.1 X-ray Crystallography

Based on inclusion in the PDB, crystallography is the most widely used method for determining protein structure. For x-ray crystallography, a concentrated and purified solution of the target protein is prepared. The solution is chemically treated to form crystals of the protein, a process that is still more art than science. The crystal is hit with an x-ray beam, and a diffraction pattern is recorded on photographic film or with electronic sensors, and the crystal is rotated for more patterns. The appearance is a set of bright dots; the intensity of each dot represents the amplitude of the wave front. To reconstruct the structure of the protein requires the phase of the wave as well as the amplitude; this leads to the *phase problem*. One approach is to substitute heavy atoms in place of lighter ones such as using synthetic amino acids that contain selenium in place of sulfur.

Once the phase problem is solved, a map of the electron density is created from the combined data using the Fourier transform. The next problem for the crystallographer is to fit the sequence within the density map, a task which can lead to errors.

The resulting structure is stored in the PDB along with parameters that indicate the accuracy of the electron map.

X-ray crystallography provides two measures for the accuracy of their models: resolution provided by the diffraction pattern measured in Ångstroms, and the *R-factor* which measures how well the final model matches the actual data. Lower values for both should mean a better model. It is not unusual that the same protein has been analyzed repeatedly to provide more accurate models.

Each stage of crystallography is open to challenges; not all proteins are amenable to purification and concentration, or crystallizing, or correcting for the phase problem. In particular, transmembrane proteins play an important role in a number of diseases but the very fact that much of the structure is hydrophobic makes the formation of crystals extremely difficult. While the number of globular proteins that have been solved number in the tens of thousands, the transmembrane proteins that have been solved number only in the hundreds [92]. My research requires a large number of solved structures for training and testing, and so is currently limited to globular proteins.

Another major drawback is that the work is labor intensive. Solving one structure can take from months to years. To deal with the labor intensiveness, there is research into using high-throughput experimental methods [8]. For CASP8 [17], almost all of the protein sequences submitted for predictions were being solved by high-throughput methods. There is a possibility that the techniques used will bias the set of solved proteins to those that are most amenable to forming crystals by the most common techniques. In contrast, the more labor-intensive experimental structure determinations

tend to focus on proteins whose structures are most needed in ongoing biomolecular research.

Nevertheless, because of the accuracy of the crystal structures, x-ray crystallography is the gold standard for experimental protein structure solution.

1.2.2 NMR Spectroscopy

Nuclear magnetic resonance spectroscopy (NMR) is the second most common technique for determining protein structures. Nuclei with an odd number of protons and neutrons have an intrinsic magnetic moment which can be polarized in an intense magnetic field. This polarization can be perturbed by a radio frequency (RF) pulse. The choice of frequency and the intensity of the magnetic field determine which nuclei are affected. The absorption, or resonance, frequencies of the nuclei can be measured, resulting in a one-dimensional spectrum. Furthermore, the frequency of a nucleus can be affected by the surrounding electron density and by a spin-spin coupling between neighboring nuclei. Two-dimensional spectra are possible by using combinations of different RF pulses. These chemical shifts in the frequency provide distance and angular constraints that can be used to determine the structure.

NMR spectroscopy requires a purified and concentrated sample of the protein, but does not require a crystal which makes it useful in cases where a crystal does not form. However, the data can become too convoluted when the protein is large; NMR is limited to small proteins.

1.3 Summary

In Section 1.1 I covered the basics of the protein structure including the three types: primary, secondary, and tertiary. In Section 1.2 I described two experimental methods for experimental structure determination. I also argued that the current experimental techniques of crystallography and NMR cannot fulfill the need for structures following the huge growth in known and putative protein sequences.

Chapter 2

Protein Structure Prediction

This chapter will cover prediction of secondary and tertiary structures of protein chains given their sequence, and includes the following:

- a fundamental method for comparing and matching similar sequences;
- statistical information based on the matching of similar sequences;
- secondary structure statistics from known structures;
- basic prediction methods for secondary structure including neural networks;
- CASP, an international experiment for protein structure prediction;
- the two categories for CASP8 tertiary predictions: template-based modeling and template-free; and
- the need for effective constraints for model predictions.

2.1 Multiple Sequence Alignments

The fundamental tool for structure prediction is the *multiple sequence alignment*. Organisms with common ancestors have protein homologs, i.e., proteins which are descendants of the same protein (possibly duplicated) in a common ancestor. These proteins tend to have similar sequences and similar protein structures. Bioinformatics turns this around to infer that similar sequences are often homologs. Data on substitutions of residues in similar structures [36] lead to substitution matrices such as BLOSUM [37]. From this information and other analysis, Altschul, *et al.* developed BLAST [2], a tool that searches the non-redundant (*nr*) database of protein sequences, selects protein sequences similar to the target (called subject sequences), provides an E-value for the statistical significance that each subject sequence is not similar to the target simply by chance, and aligns the subject sequence to the target sequence by showing which pairs of residues represent likely substitutions. Gaps are added to represent deletions or insertions. An example of a top-scoring (i.e. low E-value) subject sequence and its alignment to a target is shown in Figure 2.1. The pair-wise alignments can be combined to form a multiple sequence alignment, or MSA.

Multiple sequence alignments provide an immediate application; similar sequences tend to provide the same function. This allows for hypothetical assignment of function to proteins. If there are experimental results that indicate which residues are involved in the function, then further analysis of the sequences may do more to help determine the likelihood of similar function.

```
Length=124

 Score = 64.7 bits (156),  Expect = 2e-09, Method: Composition-based stats.
 Identities = 44/132 (33%), Positives = 71/132 (53%), Gaps = 15/132 (11%)

Query  1    MKFLTTNFLKCSVKACDTSNDNFPLQYDGSKCQLVQDESIEFNPEFLLNIVD--RVDWPA  58
            MK  T N L C+ + C    ++FPL+    K   V+  +   + +FL+++V+  R++W
Sbjct  1    MKLFTHNLLICTKRQCGI--NSFPLKITAEK---VEKVTTPLDADFLISLVESERLNWNG  55

Query  59   VLTVAAELGNNALPPTKPSFPSSIQELTDDDMAILNDLHTLLLQTSIAEGEMKCRNCGHI  118
            ++T AA +G  A+PPT P    + Q+        L  L  +++   + EGE+ C  CG
Sbjct  56   LVTSAANIGL-AVPPTLPEDWKTNQQF-------LQALWDVVMDCQVIEGELICPVCGRH  107

Query  119  YYIKNGIPNLLL  130
            Y I NGIPN+LL
Sbjct  108  YPIHNGIPNMLL  119
```

Figure 2.1: This is an example of an alignment from BLAST using the sequence for CASP7 target T0319. Here is a match with a conserved hypothetical protein with an E-value of 2.0e-9. Notice the gaps shown as dashes represent deletions and insertions. The middle row between the target sequence (here referred to as the *query*) and the subject sequence shows where there are identical matches. The "+" denotes matches between amino acids with similar chemical characteristics such as polarity and acidity.

After BLAST, there came improvements such as PsiBlast [3]. Here at UCSC, researchers applied hidden Markov models to sequence alignment and developed SAM [57, 42, 51, 52, 53, 55] which provides us with quality alignments and E-values that I use in my research.

2.1.1 Amino Acid Distribution

Besides determining the possible function of a protein, multiple sequence alignments are a source for amino acid distribution per residue in the target sequence. Amino acid distributions for a sequence provide useful information. For example, if a glycine is *conserved*, or found in every sequence within the same column, then it is likely that it is a vital part of the structure, typically in a hairpin loop. Likewise, a conserved histine may be involved in binding a metal ion.

For any column in an MSA, the normalized count of the amino acids in that column provides a naive amino acid distribution. However, the naive distribution is often inaccurate due to the paucity of data. One improvement to this is to assume a prior distribution based on what is seen in the column and applying Bayesian statistics to provide a more accurate distribution. At UCSC we invented the technique for using a Dirichlet mixture, a distribution over a set of distributions, to estimate this distribution and use this estimate as the prior [89, 50]. I use the UCSC `estimate-dist` program to calculate the distributions that I use for my predictions.

2.2 Local Structure Alphabets

In Section 1.1 I covered two basic secondary structures: alpha helices and the beta strands that form beta sheets. A third category, called loops, can be used to specify residues that do not fit into either of these categories. These three categories form a widely used local structure classification. There are classifications that make a finer distinction about the secondary structure. One is the DSSP [45] classification which, for example, separates 3-10 helices and the pi helices from the more common alpha helices. The authors not only specified eight different categories in their paper but also provided the DSSP program which used the coordinates of the backbone atoms in the structure to determine the classification.

Each of these two classifications assigns single letters to each class making it easy to display the sequence of a protein next to its secondary classification, such as

showing a part of the 2baaA sequence from the PDB with the corresponding classification from DSSP:

```
...FAWGYCFKQERGASSDYCTPSAQWPCAPGKRYYGRGPIQLSHNYN...
...GGCTTCCSBCCSCCCCCCCCCSSSCCCTTCCCCCBTTTTBCSHHH...
```

When I mention local structure *alphabets* in the following chapters, I mean local structure classifications that can be represented by single letters. At UCSC we have developed many local structure alphabets including one that classifies residues based on the number of C_β atoms within a 14 Ångstrom radius of the residue's C_β. This *burial* classification provides an indication of the residue's exposure to solvency.

A recently published paper [56] provides an excellent overview of the PREDICT-2ND program and some of the alphabets developed by the Karplus lab. I chose BURIAL-14-7, STR2, and NEAR-BACKBONE-11 as alphabets to consider for inclusion as inputs for my neural network. For more examples and analysis, see Karchin [46] and Karchin [47].

These classifications are not only used to analyze structures but are also used for predictions. Accurate predictions of local structure alphabets are an important part of current protein structure predictions.

2.3 Secondary Structure Prediction

In the earlier years, predictions of extended strands, alpha helices, and loops which comprise the EHL alphabet, only used the target sequence [19]. Each residue had a propensity to be part of one of these classes; later research made improvements by considering segments of up to 51 residues long [77]. This topped out at about 60%

accuracy and considering that the background accuracy using no sequence information is about 38%, this is not a strong result. The use of multiple sequence alignments provided the basis for the next big improvement, but to make use of it, researchers frequently turned to machine learning methods [76].

A machine learning program is first trained by processing a set of examples, pairings of inputs with desired outputs. The results of the processing may be a set of adjusted weights, or decision trees, depending on the method. Then a prediction is made by entering a set of inputs only; the program returns a set of outputs that represent the prediction. Testing for accuracy of predictions must be done with a separate set of examples, otherwise *overlearning* may occur, i.e. the program has learned the answer for all the known examples but has not generalized to make good predictions for unseen examples.

A common choice of the machine learning methods in bioinformatics is the neural network [76, 44]. I chose neural networks as the method for my contact predictions, and I discuss neural networks in detail in Chapter 4.

2.4 Tertiary Structure Prediction

The main goal of CASP is predictions of tertiary structures for target sequences. There are currently two different approaches depending on the results of the multiple sequence alignment for the target sequence: *template-based modeling* (TBM) and *free-modeling* (FM).

2.4.1 Template-based Modeling

Since we know that the primary sequence contains all the information necessary for the final tertiary structure, we may assume that if we have a sequence that is similar to our target sequence and we know the structure of that sequence, then we can propose that the structure for our target is similar. This is the basis for template-based modeling. However, there needs to be a method to determine how similar the template sequence is as well as a method for finding such sequences. For both methods we use the tools for multiple sequence alignments (Section 2.1).

If a sequence with a known structure is found in the multiple sequence alignment, it can be used as a basis, or *template*, for our prediction. If a sequence is found that has an E-value < 0.001 and is also found in the PDB, it is considered a good candidate for template. (*E-values* reflect the likelihood that the sequence is incorrectly selected. Low E-values imply unlikely, high ones imply that it is likely.) Even if there are no global matches, local matches may be found that provide templates for parts of the structure.

Once there is a template for the target, we need to create a model. Simply copying the coordinates of the corresponding peptide backbone C_α atoms in the alignment can generate a reasonable model with respect to the actual structure when measured by the criteria used at CASP. Various programs such as `MODELLER` [80] and `nest` [97] are used in an effort to improve the model. For example, `MODELLER` uses a stochastic process and a cost function to generate its models. Despite mixed results [95],

the predictions for template-based models can be quite good.

Contact predictions can be created from template-based models. Since the models are reasonably good, the contact predictions from them are also usually good. CASP assessors note this and exclude contact predictions for template-based models from assessment. This may be unfortunate as models based on local templates could still benefit from good contact predictions that provide information about areas of the structure not covered by local templates.

2.4.2 Free Modeling

There are targets for which there are no available templates. When no templates are found for a protein sequence, the task of protein structure prediction becomes much harder. This is represented by the second tertiary category at CASP called free-modeling. Predictions in this category are of significantly poorer quality than the template-base category.

The naive solution would be to examine all possible ways to fold the protein; however, the Levinthal paradox [58] argues that examining all possible configurations a protein can assume, even at billions of times a second, would require far more time than the expected life of the universe. The paradox is how can proteins find their proper configuration and fold so quickly (usually in a few milliseconds)? Of course the resolution to the paradox is that the proteins can't go through every configuration, but the relevant point is that researchers can't consider every configuration either. Even if we could, we would need an evaluation function to tell us which is the right configuration. This

complexity requires constraints to avoid an exhaustive search. The problem becomes one of determining constraints to limit the search and developing effective evaluation or cost functions.

Developing a good cost function is important. This involves taking a model and generating a score based on the current configuration. For example, we may consider how compact the structure is. By assigning a score on compactness we can start to judge how good the model is. We can also score a model on how well the hydrophobic residues are buried within the structure. By putting a number of measures together we form a cost function. Typically the function is designed so the lower the cost, the better we rate the model. Often new models for consideration are simply changes in an initial model. If the cost goes up, we discard those changes. If the cost goes down, we keep the new model or models for further adjustments.

Although there may be no templates, we can search for matches in known structures for small sections about 3 to 9 residues long [87, 9, 10, 11]. By gathering data on the structures of fragment matches and very close matches, we hypothesize that we can predict the structure of fragments in the target sequence. This allows the model builder to consider selecting fragment structures and limiting the search space. Although this method of model building reduces the search space considerably, the combinatorial growth soon takes over with larger proteins. The use of fragments res presents the current best efforts in free-modeling. Nevertheless, more constraints are needed. This is where residue-residue contact prediction comes in; the potential usefulness of contact predictions has been noted for some time [78].

2.5 CASP

To encourage development of effective methods to predict protein structures, an international experiment was started in 1994 and repeated every two years since [64, 65, 88, 67, 66]. The Community Wide Experiment on the Critical Assessment of Techniques for Protein Structure Prediction (CASP) provides the community of researchers with a list of protein sequences over a three-month period for predictions in various categories. The results and selected talks are presented at a conference near the end of November. This experiment has helped to encourage development of the prediction methods.

The predictions come from two approaches: biophysics and bioinformatics. Biophysics uses first principles of physics to make predictions based on protein folding. However, the computational complexity of the molecular and quantum mechanics has severely limited the results. Bioinformatics does not attempt to emulate the folding process, and, while it does use some physical properties such as hydrophobicity, it primarily uses databases such as known protein structures and rotamer limitations, and often includes machine learning techniques and simplified physical models to make structure predictions. I use the bioinformatics approach in making my predictions.

At CASP, there are other categories of predictions besides the tertiary structure predictions. In particular I will be discussing *residue-residue contact* predictions, and unless otherwise mentioned, I will be using the current CASP definition of residue-residue contacts. Two residues of a protein sequence are defined as *in contact* when

the respective C_β's of the two residues are within 8 Ångstroms of each other (C_α if the residue is glycine) [66]. There are other definitions for contact such as when two residues share a common Voronoi boundary [24]. The distance used for CASP is arbitrary but useful because of the consistent use. We also did some brief experiments involving alternative contact definitions such as using side-chain centroids rather than C_β, but found that the CASP definition actually provided the best results when predicting contacts and adjusted for background probability (data not shown).

2.6 Review

In Section 2.1 I covered multiple sequence alignments and their importance to both bioinformatics and molecular biology including the derivation of the amino acid distribution for a sequence. I introduced the use of alphabets for local structure classification in Section 2.2, which was followed by a brief history of predicting the secondary structure in Section 2.3. I covered the more difficult challenge of tertiary structure prediction in Section 2.4. Finally, in Section 2.5, I discussed an important effort in protein structure prediction: CASP.

Chapter 3

Contact Predictions

After introducing some conventions I will be using now and in later chapters, this chapter will cover

- correlated mutations,
- correlation statistics for contact prediction as well as those contact predictions that use them, and
- contact predictions that use neural networks.

3.1 Conventions

Separation: Given two residues with sequence indices, i and j, separation is defined as $|i - j|$. I shall show how separation plays an important part in predictions. CASP evaluates predictions for separations of ≥ 6, ≥ 12, and ≥ 24. My predictions are for separation ≥ 9.

Paired statistics: I define *paired statistic* as a statistic using both the i and j columns of a multiple sequence alignment as parameters. Correlation statistics are one class of paired statistics, but not all the statistics I use are correlation statistics. Also I will use the following conventions when discussing paired statistics used in prior research as well as my own. For each pair of columns in the MSA, I build a contingency table indexed by the twenty residues (corresponding to the twenty essential amino acids). Each entry in the table represents the count of the pairs of residues in the two columns (pairs do not include gaps). I use the counts in the contingency table and the marginal counts in computing the statistical values. Aspects of the table are as follows:

- $C_{x,y}$ is the count of pairs of residues, x and y, in the respective columns;
- C_x and C_y represent the marginal counts of the residues for the two columns, $C_x = \sum_y C_{x,y}$, $C_y = \sum_x C_{x,y}$; and
- R is the set of twenty residues and T is $\sum_{x,y} C_{xy}$.

3.2 Early Predictions and Correlation Statistics

With the advent of multiple sequence alignments, researchers had an effective evolutionary tool for predicting contacts. The approach used the notion of co-evolving or *correlated mutations*. When one residue mutates in a protein structure, there is a likelihood there will be a compensating mutation nearby [78]. The compensation may be for structural or functional reasons. If functional, the two residues may be close but not close enough to fit the CASP definition of contact.

The following research uses correlation statistics to score pairs of columns; the better the score the more likely there is a correlated mutation due to contact. As I have noted, there can be other reasons for correlated mutations other than contact. Later research introduced in Section 3.3 may be getting better results due to detection of the differences.

By 1994 multiple sequence alignments were established as a source of evolutionary changes. Rost and Sander [78] acknowledged the use of correlated mutation for identifying likely contact pairs. By analyzing the pairs of residues in the two columns of i and j using a correlation statistic, such pairs could be detected.

Taylor and Hatrick [91] compared a form of correlation scoring to scoring based on conservation of residues. Given sequences, x, y from the MSA and indices i, j, they compared x_i, x_j to y_i, y_j. They assumed compensation is physicochemically based, but they focused only on the two residues in one sequence at a time to determine a value that they added to the score. They also excluded pairs where $x_i = y_i$ or $x_j = y_j$ or $x_i = x_j$ or $y_i = y_j$, which would reduce the number of pairs considerably. When they compared their scoring values to pairs of columns that were highly conserved, they found that the highly conserved signal was stronger for indicating contact than their correlation scoring method. But their negative results for correlation is likely due to the pairs they excluded as well as the fact that they did not consider using standard correlation statistics.

On the other hand, Göbel, *et al.* [31] approached the problem by using the MacLachlan substitution matrix [63] to provide the relative likelihood of one type of

residue mutating to a specific other type. The matrix, a 20×20 matrix indexed by residue type, provided them with a null model consisting of random but biased mutations against which they would detect correlations.

Weighted correlation coefficient: Göbel, *et al.* used a version of correlation coefficients that sought to model the process of mutation and the difference between background mutations and correlated mutations. They used the Pearson correlation coefficient. Given two random variables X, Y with mean values $\bar{X}, \bar{Y}$ and variances σ_X, σ_Y, the correlation coefficient is:

$$P = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sigma_X \sigma_Y}.$$

They use the MacLachlan substitution matrix as the basis for their random variables. Let S_{ikl} be the value in the MacLachlan matrix for the substitution of the residue in column i and sequence l for the residue in column i and sequence k. Let $\bar{s_i}$ be the mean and σ_i be the standard deviation for all such substitutions in column i.

Furthermore, they were concerned about the effect of too many similar sequences in the multiple sequence alignment. They added a weighting value based on the dissimilarity between the two sequences by using the method defined in Sander and Schneider [81]. The more dissimilar the sequences, the higher the weight; weights were

normalized so the sum of all weights equaled one. The resulting formula is

$$\mathrm{WCC}(i,j) = \frac{1}{N^2} \sum_{k,l} \frac{w(k,l)(s_{ikl} - \bar{s_i})(s_{jkl} - \bar{s_j})}{\sigma_i \sigma_j}$$

Unweighted Correlation Coefficient: They also evaluated the unweighted case, where $w(k,l)$ is always one. Their results suggested that the weighted version may be slightly more accurate.

Because the weighting has been discarded, I can compute the statistic using the contingency table. I can consider all pairs transitioning from one entry to another. Let C be the contingency table for columns i, j and M is the MacLachlan matrix Then:

$$\sum_{C_{x,y} \in C} \sum_{C_{x'y'} \in C} \frac{C_{x,y} C_{x'y'} (M(x,x') - \bar{M_i})(M(y,y') - \bar{M_j})}{\sigma_i \sigma_j}$$

where $\bar{M_i}$ represents the mean and σ_i is the standard deviation for all MacLachlan matrix transitions in column i (likewise for column j). The mean and standard deviation for each column can be pre-computed for efficiency.

Mutual Information: Mutual information is a measure of how much information is gained about the residue in column i by knowing what the corresponding residue is in column j. I use the following formula to get mutual information:

$$\mathrm{MI}_{i,j} = \sum_{x \in R} \sum_{y \in R} \frac{C_{x,y}}{T} \log \frac{C_{x,y} T}{C_x C_y}$$

Highly conserved positions have low mutual information, but positions with correlated mutations should have high mutual information.

Joint Entropy: This follows from the standard definition of joint entropy:

$$\mathrm{Ent}_{i,j} = \sum_{x \in R} \sum_{y \in R} \frac{C_{x,y}}{T} \log \frac{C_{x,y}}{T}$$

Pairs of highly conserved positions, which have low joint entropy, are more likely to be in contact than highly variable positions. The joint entropy also provides an upper bound on the mutual information for a pair of columns.

Mutual Information Over Entropy: This combination of statistics attempts to normalize the mutual information measure based on conservation.

$$\mathrm{MIE}_{i,j} = \frac{\mathrm{MI}_{i,j}}{\mathrm{Ent}_{i,j}}$$

Martin, *et al.* [62] argued that mutual information was an effective statistic. They also tested using joint entropy to correct mutual information at highly conserved positions. Their results show that mutual information over joint entropy was a better predictor. However, I shall show in Section 8.1 that mutual information by itself is a poor statistic, and that the improvement they got from joint entropy is likely due to the strength of joint entropy alone.

Fodor and Aldrich [29] examined the effect of conservation on several of these

statistics resulting in an unbiased comparison of them. They found that Pearson's correlation coefficient (which they call McBASC) was the best of the group. Next best was a statistic they introduce called OMES (Observed Minus Expected Squared). They found that mutual information was the least effective which I confirm later in my own research.

OMES: Observed Minus Expected Squared is calculated as

$$\text{OMES}_{i,j} = \frac{1}{T} \sum_{x \in R} \sum_{y \in R} \left(C_{x,y} - \frac{C_x C_y}{T} \right)^2$$

.

Fodor and Aldrich tested a fourth statistic, Statistical Coupling Analysis, or SCA [59], and it fell between OMES and mutual information as an effective statistic. However, retrieving the software necessary for this statistic proved problematic, and I did not include this statistic in my evaluations or as an input for neural networks.

Their paper also suggested that the statistics were measuring different signals implying that they could make complementary inputs for a neural network. The paper is especially appealing because Fodor made the Java software he used available to others. I found this an excellent way to test the accuracy of my own software.

3.3 Predictions using Multiple Sources

Olmea and Valencia [69] were among the first to combine multiple sources to improve predictions including the unweighted correlation coefficient and, from the MSA, sequence variability, secondary structure via the HSSP database, and accessibility. They used a simple linear combination of the inputs which they optimized by a naive regression technique. They showed results improving on the initial correlation coefficient statistic by a factor of two.

The next major improvement after the introduction of correlated mutations and early correlation statistics, occurred when Fariselli and Casadio [26] trained a neural network for predicting contacts. Interestingly, their method did not use any correlation statistics as input but rather the method had 210 inputs for all the possible pairs of residues $(20 \times (20+1)/2)$ with each input containing the normalized count of that pair of residues at i and j. The method also included one input for sequence length and one for the separation. Fariselli and Casadio had several networks where they included inputs for hydrophobicity at both i and j, and the biggest included pair inputs for pairs $i-1, j+1$, $i-1, j-1$,$i+1, j-1$, and $i+1, j+1$. This network contained a total of 1054 inputs. They trained the network on 200 proteins, and reported their results by using the C predictions where C was the actual number of contacts for the sequence. They showed that predictions were more difficult for longer proteins. They also tried a filter on the predictions limiting the number of predictions for each residue based on the expected number of possible contacts for that residue. This filter improved accuracy to

about 0.18 which maps to approximately $4L$ on my accuracy graphs. While this appears to be a very impressive result (compared to my CASP7 predictor which gets 0.22 at $1L$), Fariselli used a minimum separation of four which includes the very common contacts in alpha helices.

In 2001, Fariselli and colleagues added inputs to this basic neural network that included the Pearson correlation statistic, predicted secondary structure, and sequence conservation [27, 28]. The final network improved the accuracy to 0.25 from 0.18. Again, the accuracy of the predictions was based on the actual number of contacts rather than a percentage of the sequence length and includes contacts from alpha helices.

In 2004 Hamilton *et al.* [34] described a neural network that expanded on the use of *windows*, that is, not only taking inputs based on columns i and j, but including columns around each of them. This paper used windows of size 5, and looked at all columns in $i \pm 2, j \pm 2$. Using a separation greater than four, the paper calculated accuracy based on fractions of the sequence length, L/10, L/5, and L/2, rather than the actual number of contacts. They used the propensity of contact between the types of residues in pairs as an input. They also encoded secondary structure prediction as three binary inputs representing the EHL classification using windows of size 3. They included sequence length, separation, and classification of the basic chemical properties of the residues as the other inputs. The result was a modest sized, but well performing, network.

Punta and Rost [73] developed a neural network predictor, PROFcon, that was assessed as one of the best contact predictors at CASP6 [66]. They used inputs similar

to those above, but also included a window of residues around $(i + j)/2$. They argued that the central connecting segment between the two residues contained important information and their results support that argument. Importantly, they acknowledged that sparse alignments resulted in poor predictions; the contact prediction results of CASP8 dramatically underscored this fact (see Section 12.4).

3.3.1 Alternative Methods

There are different approaches than just using a standard neural network. For example, Baldi, Pollastri, and their colleagues have been using recursive and recurrent neural networks to predict contacts [71, 5, 72]. Their results are problematic because it is not clear how the accuracy predictions relate to the length of the protein. Also they seem to include pairs with separation less than 6. As a result, they include the large number of contacts involved in helices. It becomes very difficult, if not impossible, to compare their accuracy results to those at CASP and those reported by other researchers. Given the higher background probability of contacts, the accuracy results in their articles are significantly higher than those of other researchers, and I must note that they have not had any notable success at CASP where all predictions are assessed equally.

Another alternative method did compete at CASP6; MacCallum [60] used self organizing maps and genetic programming for contact predictions. At CASP6, his method was assessed as one of three groups that distinguished themselves from the other groups in contact prediction. Self organizing maps are a form of clustering. MacCallum used them to classify each residue of a multiple sequence alignment and these

classifications, along with examples from contacts in β sheets, were used as examples for genetic programming [32], a machine learning technique that I will not be discussing in this thesis. The results of the genetic programming were two functions that were used later to generate the predictions. These contact predictions were specifically developed for β sheets. The prediction of β sheets in free modeling targets is difficult[79], and such contact predictions for β sheets may help.

Bystroff [86, 98] explored a different source of data for contact prediction by finding I-sites [15] via a hidden Markov model. I-sites are sequence motifs related to structure motifs. His first method used ROSETTA [9, 15] to make I-site predictions; however, this approach essentially was equivalent to using ROSETTA to generate a 3D model which could be used to generate contact predictions by itself. Since my objective in making contact predictions is to help programs like ROSETTA build models, I did not find this method interesting.

The second method used Bystroff's own HMMSTR-CM program, which was built on top of HMMSTR [16]. This simply uses the original sequence (and multiple sequence alignment) to help identify the structure motifs and, in turn, predict likely contacts. Again, the output of HMMSTR-CM could be used directly by a model builder, bypassing the model builder's need for its contact predictions.

3.4 Review

This chapter introduced my area of research, residue-residue contact predictions. I provide a set of notational conventions in Section 3.1 used in the equations for statistics. In Section 3.2, I covered the development of contact predictions over the past two decades, including the idea of correlated mutations and related statistics. In Section 3.3, the history moved on to the combining of multiple sources of data to improve the predictions and also covered some alternative approaches.

Chapter 4

Neural Network Design

This chapter will discuss the implementation of feedforward neural networks in detail, providing an understanding of the choices I made in selecting and setting up the neural network for contact predictions. It includes general issues concerning over-training and choice of back-propagation, and issues specific to contact prediction such as dealing with a large set of examples and the very large ratio of negative to positive examples used in training.

This chapter is not a formal description of a neural network. Those interested in a formal description should consult Herbert Lee's booklet [39].

4.1 How Neural Networks Work

Neural networks have become the default technique for bioinformatics simply because they have provided the best results [77], e.g. PSIPRED [44]. While there are different versions of neural networks, the one I will discuss is a multilayer feedforward

neural network. An example of the structure is shown in Figure 4.1.

The goal of the training phase is to evaluate the predictions for a test set calculating the total output error, to use the error to readjust the weights of the network to decrease the error, and to repeat this process. A separate *cross-training* set is used to avoid *overfitting*, learning all the specific answers for the training set and failing to generalize. By using a separate cross-training set and stopping the process when the error of the cross-training set stops decreasing, we can properly monitor our progress during the training phase.

The following implementation is based on the neural networks I used for research. While the basic three layers I use are common, they are certainly not universal. Specifically the PREDICT-2ND program[56], the neural network program which we use for local structure predictions, uses a much more complex series of hidden layers. Also, while the forward propagation is quite standard, the method for back propagation, RProp [75], is relatively new and is not used in PREDICT-2ND. Nevertheless, the principle of calculating an error and back-propagating to adjust weights is fundamental.

To calculate each prediction we start at the layer above the inputs for each node n_1. We calculate the output of each node as follows: let $\overline{y}_i$ be the set of inputs to node, n_i, $\overline{W}_i$ be a set of weights associated with each input, and g be a sigmoid function that maps $(-\inf, \inf)$ to $(0, 1)$; we use the logistic function, $\frac{1}{1+e^{-x}}$. Then the output for n_i is $g(\overline{W}_i \cdot \overline{y}_i)$. Note the extra nodes with no inputs in Figure 4.1. These nodes simply output 1.0. This is called the *bias*, which allows the node's output to be translated as well as scaled.

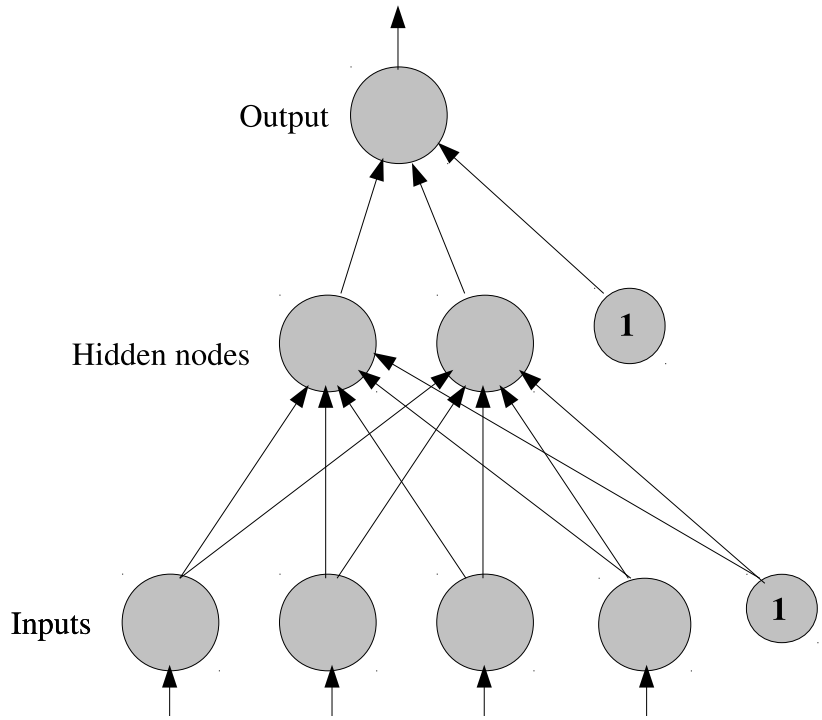


Figure 4.1: Example of the feedforward back-propagation neural network used in my research. Each row is a layer: the bottom is a set of four input nodes, the middle is the set known as the hidden layer, and the top is the single output node. Feedforward is represented by the arrows that direct floating point values from layer to layer starting at the input nodes. Each of these arrows has a weight associated with it. During back-propagation, the weights are adjusted with the intent to reduce the error in the output. The nodes to the side without inputs and mark with a one are *bias* nodes; they output a constant 1.0 and their arrowed feedforward also have weights associated with them.

To minimize the error, we need to specify what we mean by output error and the choice of an error function is significant. When the outputs can range over $(0.0, 1.0)$, the sum-of-square errors function, which is also used in linear regression, is a common choice:

$$E = \sum_{i=1}^{N} \sum_{j=1}^{O} (t_{ij} - p_{ij})^2$$

where N is the number of examples, O is the number of outputs, t is the target output, and p is the predicted output. However, in the research I discuss later, the output is a single classification; 1.0 means the example is a contact, 0.0 means it is not. The desired output is binary and now we have a situation where the noise in the data is a Bernoulli distribution rather than a Gaussian or other distribution. Because it is a Bernoulli distribution, the cross-entropy error function is a better choice:

$$E = -\sum_{i=1}^{N} \left(t_i \ln(p_i) + (1 - t_i) \ln(1 - p_i) \right).$$

If there are several outputs used to designate different classes, where the desired output is 1 for target class and 0 for the other outputs, then a method called *softmax* is used [14]. This is what is used in PREDICT-2ND.

Finally, I explain how we use *back-propagation* to take this error value and adjust the weights. The back-propagation method I will show is called *resilient back-propagation* or RProp. I chose this method over others because results indicate RProp is effective for classification prediction [75, 43, 90]. Again, this method is not used in PREDICT-2ND.

There are two other fields associated with each weight in the network: the *delta weight*, which is initially some small value, and a sign, which is initially either $-$ or $+$. The procedure starts at the output level and works backwards to the source level. For each node on the current level, we adjust each weight by taking the partial derivative (chosen as our sigmoidal function) with respect to the weight of the logistic function. While most back-propagation methods use the value of the partial derivative, RProp is only interested in the sign. We first ask is it negative or positive? If negative, add delta to the weight, else subtract it. Second, we ask is the sign different from the stored sign? If so, delta is multiplied by 0.5 and the old sign is replaced by the new sign (because we passed a minimum) else it is multiplied by 1.2. The process is designed to be a gradient descent to a minimum error.

This process continues by moving back through the network, adjusting each weight as necessary. All the examples are processed, a new error is accumulated, and the process repeats until the cross-training fails to show improvement in predictions.

While neural networks provide a good method for classification by training, they tend to be "black boxes" that provide little, if any, information about which inputs are the most relevant, or how the inputs are related. It is up to the designer to consider carefully which inputs to use.

4.2 Building the Input On-the-Fly

The training of a neural network requires examples, i.e. a vector containing the values for the inputs and a vector of values representing the desired output. For contact predictions, output vector has only one desired value, 1.0 if there is a contact, and 0.0 if there is not. On the other hand, the input vector can become quite large; the number of inputs for the CASP6 predictor is 280, and later predictors have around twice that number. Each element of the vector is a floating-point number which may take 4 or 8 bytes depending on the desired precision. The number of sequences in the training sets is approximately 500, and the number of examples from each is approximately 5000. When you put it all together, assuming 300 inputs and precision that uses 4 bytes, you get three billion bytes or three gigabytes of memory. This does not include some of the overhead for maintaining the structure of data.

This exceeds the available memory for most computer workstations. I solved this problem by building the examples on-the-fly. There is considerable redundancy in each example; when building examples for i and j where i is being changed each time, the local prediction values around j don't change. As a result, a large training set of examples can be built rather than stored, resulting in reduced RAM required. By using profiling, I found that the extra time to build each example is small compared to the processing time for each example, so little time is lost for considerable space saving.

4.3 Selection of Neural Network and Back-propagation Method

Rather than write my own neural network code, I chose an off-the-shelf source of code. I tried two different packages: Lightweight Neural Network [93] and Fast Artificial Neural Network or `fann` [68]. I eventually settled on `fann` for developing the networks used in CASP7 and CASP8 because `fann` had simpler code, making for easy modification to handle building inputs on-the-fly, and it had RProp [75] as a back-propagation method.

RProp is a simple method that depends only on the sign of the derivative with respect to each weight and not the actual value of the derivative. Research has shown that RProp is effective for real-world classification problems [35, 90]. Furthermore, RProp has only two parameters to adjust and their values can be left to the default values with no impact on the effectiveness of the back-propagation. Other approaches such as Quickprop [25] or the use of a value called momentum are much more sensitive to the values of their parameters.

4.4 Training and Cross-training

Neural networks, like many machine learning techniques, can over-fit the examples. In effect, they can learn all the individual examples while failing to generalize, so they do poorly on new examples. There are two methods to use to avoid over-fitting: keeping the parameter space small compared to number of examples, and monitoring

the learning with a *cross-training* set.

The solution I used was to include a separate set of examples for cross-training. After several cycles (or *epochs*) of training, I test the network using the cross-training to see how well the neural network is doing. The code is written to record the last best score on the cross-training set and store a copy of the current state of the neural network. If a predetermined number of epochs has passed without a new best score, then the training stops. The drop-off in improvement marks the point where over-fitting is occurring.

It is important that the cross-training set has no examples in common with the training set. Furthermore, a third set, used for final evaluation of how well the neural network performs, must also be disjoint from both the first two sets. Failure to fulfill this requirement can result in over-stating the effectiveness of the neural network.

Both the number of epochs per cross-training and the number of epochs used to determine the stopping point are parameters for the training program. In general I use four epochs per cross-training and from 100 to 250 epochs without improvement as the stopping point for training.

4.5 Downsampling

Another problem for residue-residue contact prediction is the disparity between positive and negative examples. The number of pairs of residues is $O(L^2)$ while the actual number of contacts is $O(L)$ where L is the length of the sequence. The overall

background probability of a contact is about 3% (see Figure 7.1). Given this, a neural network can quickly learn to predict that there is not a contact given a pair of residues; it would be correct 97% of the time!

This can be countered by *downsampling*. I reduce the number of negative examples to bring them closer in number to the positive examples. The standard way of doing this is to build a set of examples with fewer negative examples. The approach I take is different. Instead, when the program has a negative example to present to the network for training, it then randomly decides whether or not to present it.

The likelihood of discarding a negative example is another parameter available for the training program. The advantage is that, in the long run, the network will see a larger set of negative examples. This is not unlike the technique of bagging used with other machine learning methods [12] and may provide the same advantages. Bagging, or bootstrap aggregating, is a technique of taking subsets of the examples when training a collection of machine learning predictors. Such subsets tend to simulate disjoint sets of examples and can help provide better learning and prevent overlearning.

4.6 Size of Hidden Layer

The generalization of a neural network is sensitive to the number of nodes in the hidden layer. If there are too few, then the network over-generalizes and predictability goes down. If there are too many, learning is slower and there is risk of over-learning. In general, I tended to include more nodes when setting up the hidden layer. To prevent

the problem of over-learning, I used the method of cross-training to detect and avoid over-learning.

4.7 Review

This chapter provided a detailed description of the implementation of the basic neural network I currently use, one that used RProp back-propagation in Section 4.1. An issue concerning the size of the data set was discussed in Section 4.2. This led to my choice of neural network software in Section 4.3. The next two sections, Section 4.4 and Section 4.5 dealt with issues on ensuring that the network is properly trained. Finally, in Section 4.6, I discussed how I determined the size of the hidden layer in my neural networks.

Chapter 5

Selecting Neural Network Inputs

Which inputs are chosen and how they are scaled is essential to making good contact predictions. While neural networks are resistant to redundant or ineffective inputs, such inputs are likely to reduce predictive power. Such extra inputs also add to the training time, making it difficult to fine-tune the network.

In this chapter I cover the paired statistics and include two new ones: propensity and mutual information E-value. I will discuss the local structure alphabets for possible inclusion, and show an evaluation of them.

5.1 The `correlated-columns` Program

Kevin Karplus had developed a program that computed mutual information as a statistic, given a multiple sequence alignment. I made major modifications to the program, `correlated-columns`, so new paired statistics could be easily added and specified by the program's built-in scripting language. The final version of the program

generates columns of both the statistical value and its ranking for the selected statistics.

The scripting language provides a way to specify the top number of each statistic as a multiple of the length of the sequence. If a pair of residues is in the top number for one of the statistics but not for others, the value and rank of those missing statistics is computed and the results included. Therefore, there is no missing data for those statistics chosen for training purposes. The top number of statistics specified ranged from $4L$ to as much as $12L$ for the most recent research, where L is the length of the sequence.

The output provides the paired statistical data used for training the neural networks.

5.2 Paired Statistics

From Chapter 3, I have the following statistics:

- mutual information,
- joint entropy,
- mutual information over joint entropy,
- Pearson's correlation coefficient (or McBASC),
- weighted correlation coefficient,
- and OMES.

My research includes two new statistics besides the ones just mentioned: a statistical significance for the mutual information value and a measure of the likelihood, or propensity, of two residue types being near each other.

5.2.1 Mutual Information E-Value

Much of the discussion below about mutual information E-value comes from the paper I wrote for CASP7 [85]. I have added some changes to expand the explanation.

My research in Chapter 8 shows that mutual information is a poor contact predictor, despite the theoretical attractiveness of the statistic. A new statistic was suggested by my advisor, Kevin Karplus, who hypothesized that the problem arises because of the sparseness of the contingency tables, and that the observed mutual information was more a function of small-sample effects than of the correlated mutation signal we were interested in.

We note that marginal counts are statistically more dependable. What we want to determine is how significant the observed mutual information is, given the marginal counts. To do this we need to estimate a distribution of mutual information values for random contingency tables that have identical marginal counts, then use that distribution to estimate the probability of seeing the observed mutual information by chance (the p-value). We multiply the p-value by the number of pairs of columns tested to get the E-value, which is the number of residue pairs expected to score this well on the whole protein.

We construct random contingency tables with the same marginal counts, which

can be achieved by randomly permuting one of the two columns of the multiple sequence alignment, and measuring the mutual information of the newly generated pairs. When he repeated the computations for large numbers of such pairs, he observed that the resulting distribution was extremely well fit by a gamma distribution (data not shown), though we have no theoretical justification for the gamma distribution. For densely filled tables, he found that the mutual information asymptotically approaches a χ^2 distribution [83], but neither of us could find theoretical analysis of mutual information for sparse contingency tables.

Because we have many pairs of columns to evaluate, I made modifications to the initial code in the program `correlated-columns` so the prediction method only generates 50 random contingency tables per pair, and uses moment matching to fit a gamma distribution [54]. Moment matching is a practical way to calculate the parameters for the gamma distribution given a set of data points. Then, given the top number of pairs I wished to have for training (which ranged from $4L$ to as much as $12L$), I recompute the E-values using 500 random tables.

5.2.2 Propensity

The second new statistic is a simple measure of the likelihood, or propensity, that two residue types are in contact. Kevin Karplus processed a set of 2191 proteins by counting the number of possible contacts per pair of residue types in each sequence, using that count as the denominator, then counting the number of actual contacts per pair of residue types and using that as the numerator. Finally, he used the results to

build a distribution table over all possible residue pairs:

$$\text{Prop}(x,y) = \text{Prop}(y,x) = \sum_{s \in S} \frac{\sum_{(i,j) \in P(s,x,y), |i-j| \geq 6} \text{Contact}(i,j)}{|(i,j) \colon (i,j) \in P(s,x,y), |i-j| \geq 6|}$$

where S is the set of sequences, $P(s,x,y)$ is the set of residue pairs (i,j) for sequence s where the residue types of i and j are x and y respectively, and $\text{Contact}(i,j)$ is 1 if residues i and j are in contact, otherwise 0.

There are two other variations on the propensity statistic. Both use the function $\text{ProbContact}(i,j) = \text{probability}(\text{Contact}(i,j)||i-j|)$. The first variation replaces the denominator by $\sum_{(i,j) \in P(s,x,y), |i-j| \geq 6} \text{ProbContact}(p)$. The second includes that adjustment and includes $\text{ProbContact}(p)$ in the numerator's summation. Experiments show that the difference in these three as predictors is negligible (data not shown).

Melissa Cline has shown that propensity is not a strong signal [20] when considered over a single sequence. Almost all of the signal is shown to be hydrophobicity. If both of the considered residues are hydrophobic, and thus typically found in the core of the protein, they are more likely to be in contact than otherwise. Nevertheless, I shall show that propensity is a good signal for contact compared to other paired statistics when multiple aligned sequences are used.

5.3 Local Structure Alphabets

I have discussed the concept of local structure alphabets in Section 2.2. I now address the specific ones I consider for inputs. Here and in the chapters on research,

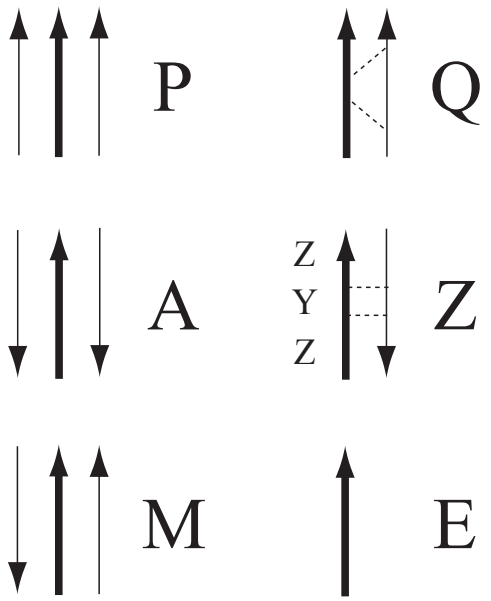


Figure 5.1: The STR2 alphabet expands on the classification of a simple strand. Here each letter represents a different juxtaposition of the strand to other strands. P is a residue on a strand between two strands of a parallel sheet, Q is a residue on an edge strand of a parallel sheet. A is a residue on a strand between two strands of an anti-parallel sheet. Y and Z are residues on an edge strand of an anti-parallel sheet. Y has h-bonds to the sheet, Z has none. M is on a strand which is parallel on one side, and anti-parallel on the other. Finally, the E class is for sections of strands with no h-bonds (usually beta-bulges).

the alphabets are identified by short notations.

The alphabets can be divided into three groups: backbone geometry alphabets [47], residue burial alphabets [46], and h-bond alphabets (currently the only published information on the h-bond alphabets is available in the supplement for a paper [4]). The backbone alphabets I used are derived from DSSP, defined by Kabsch and Sander [45] or from the more recent STRIDE, defined by Frishman and Argos [30]. The burial and h-bond alphabets were developed at the Karplus lab.

DSSP is an alphabet of eight letters: E for beta strands, B for short beta

bridges, G for the 3_{10} helix, H for the α helix, I for the rare π helix, T for turns, S for bends, and C for all others. The turns are characterized by a hydrogen bond similar to a helix, and bends are characterized by a backbone geometry with a tight curvature.

STRIDE is an alphabet of seven letters: E for beta strands, B for short beta bridges, G for the 3_{10} helix, H for the α helix, I for the π helix, T for turns, and C for all others. STRIDE improved on the recognition of helices by considering the dihedral angles.

str2: This alphabet expands on DSSP. Most of the classification is unchanged: `I` is mapped into `H`, and `E`, the strand from DSSP, is subclassed into `AEMPQZY` depending upon aspects such as anti-parallel, parallel, and presence of h-bonds. The different categories for strands are explained in Figure 5.1.

ehl and ehl2: These two alphabets reduce DSSP to the traditional EHL classification. EHL maps strands onto E; loops, turns, beta-bridges, and π helices onto L; and α and 3_{10} helices onto H. EHL2 maps all helices to H, strands and beta-bridges to E, and all else to L.

near-backbone-11 and burial-14-7: These two alphabets indicate how much exposure the residue has to solvent (water). The main difference is the method used for measuring burial. Both define a point that represents the center of the residue as follows: form a plane using N-C_α-C, define $C_\alpha$as the origin and N-$C_\alpha$as the x-axis. The residue center point is defined as -2.66, -5.15, 3.48 Ångstroms. For BURIAL-14-7the burial clas-

sification of a residue uses the number of C_β's found within a 14 Ångstromradius of the the residue's center point. For NEAR-BACKBONE-11, the classification of a residue uses the count of near-backbone points defined as the point found at the coordinates 1.24, 0.64, 0.23, near the backbone nitrogen. A more thorough discussion of NEAR-BACKBONE-11is available in the online supplementary material for a paper by John Archie and Kevin Karplus [4].

h-bond alphabets: This is a set of alphabets based on the residue's hydrogen bond or h-bond. The criteria used for defining an h-bond is based on five different geometric features. The prefixes O and N distinguish between the residue's backbone oxygen or nitrogen being part of the h-bond. The "notor" and the "sep" alphabets are the two main groups. The "sep" alphabets use separation between the two residues of the h-bond to distinguish between classes. The "notor" uses a combination of secondary structure properties, such as parallel or anti-parallel, along with the torsion angle formed by the nitrogen and oxygen. All alphabets have classes to cover non-backbone h-bonds, multiple h-bonds, and no h-bonds. The h-bond alphabets were not available for CASP7.

str4: This alphabet uses part of the classification of STR2 and part of the classification for N_NOTOR2. Like the h-bond alphabets, it was not available for CASP7.

5.3.1 Evaluation of the Alphabets by Predictability

The alphabets I have discussed have been evaluated with respect to predictability. Rachel Karchin did considerable work on the protocol for evaluation [53, 46]. Pre-

alphabet	**network**	**bits-saved**	Q_n
STR4	IDGaaH13	1.148	0.442
STR2	IDGaaH13	1.122	0.558
STR2	IDGaaH13 (t04)	1.091	0.551
N_SEP	IDGaaH13	0.978	0.632
N_NOTOR2	IDGaaH13	0.912	0.592
N_NOTOR	IDGaaH13	0.862	0.591
O_SEP	IDGaaH13	0.855	0.610
O_NOTOR2	IDGaaH13	0.814	0.578
EHL2(dssp)	IDGaaH13	0.786	0.779
O_NOTOR	IDGaaH13	0.772	0.577
BURIAL-14-7	IDGaaH13	0.573	0.354
NEAR-BACKBONE-11	IDGaaH13	0.545	0.249

Table 5.1: This table shows various alphabets evaluated for predictability as measured in bits-saved and evaluated in Q_n. The higher the bits-saved the better the predictability. All alphabets were evaluated using alignments from SAM-T06 except for the line containing (t04), which shows the difference for STR2 when using alignments from SAM-T04. The identifier for the specific PREDICT-2ND neural network architecture is also indicated.

dictability is assessed as the average of the log likelihood ratios between the probability of an alphabet class, a, $P(a)$, and the background prior probability of a, $P_b(a)$. This ratio represents information gain, and is measured in bits-saved. We favor predictability over the traditional Q_n because we are evaluating alphabets of different sizes and Q_n is naturally biased in favor of small alphabets.

I used predictability when considering which alphabets to include in the neural network; top performers in this evaluation were favored for inclusion. In Table 5.1 I show the results obtained by Rachel Karchin and Grant Thiltgen (who helped develop the h-bond alphabets). Included in the table besides predictability is the standard Q_n measure and the identifier for the neural network architecture.

5.4 Input Formats

The sources of inputs is only part of the picture. I now discuss different ways the sources can be inputted to the neural network.

5.4.1 Windows

Often researchers have not only provided inputs for predictions and data at the two residues, i, j, but also for nearby residues. These extra inputs are referred to as *windows*. A window consists of including inputs for $i-n, ..., i-1, i+1, ..., i+n$ and $j-n, ..., j-1, j+1, ..., j+n$. The width of the window is the number of residues in the window; widths are always odd numbers. In my research I will be examining windows of widths from three to seven.

5.4.2 Rank or Value

The paired statistics can be input to the neural net as either the value or as the rank of the statistic's value compared to all other instances of that statistic's value for a given protein. I initially started out using value only, but later I tested which input type (if not both) may be most effective.

5.4.3 raw value, log value, or adjusted value (z-value)

There are several ways to enter a value. The simplest is the raw value. Another format is the log of the value. There are two reasons for using log. First, in neural networks each node takes the weighted sum of the inputs. If each value is a log or log

equivalent, then the sum represents the multiplication of values. Second, the values represent a log distribution. As seen in Figure 7.1, the distribution of separation vs. probability of contact is approximately logarithmic, so entering separation as log is reasonable.

There are other ways to adjust a value such as the length of the sequence simply by dividing the value by 1000. Another is to normalize the value to a range from 0 to 1. I later tested using z-values as inputs (see Section 10.1). The values are adjusted by using $\frac{value-\mu}{\sigma}$ where μ is the mean and σ is the standard deviation of the statistic's values with respect to the sequence.

5.4.4 binary or real value

Another way to enter length is to break it into different ranges and treat it as binary inputs: 1 for the actual range, 0 otherwise. Binary inputs that represent classes work well when the neural network needs distinctly different learning for each class.

Using separate binary inputs for each range of values allows the neural network to learn complicated non-linear functions at the expense of discarding the information of the actual value. This can also apply to any of the alphabet entries. For example, EHL can be entered as one of three binary inputs. However, the output of the local structure predictor I use provides a vector of probabilities. My decision is to enter these as the probability vector by assuming that there is more information and that the network can use the extra information.

5.4.5 summary statistic(s) or vector

If the classes represent divisions of a scalar value then the probability vector can be summarized by using the mean value, optionally with the variance. For example, burial can be summarized as a mean value representing the depth of burial while secondary structure cannot. There is no meaning to an average value for distinct structures, such as one-third helix and two-thirds strand. The advantage would be to reduce the number of inputs and possibly improve learning. I tested this in my research for burial and found that the summary statistic did not perform as well as the actual vector of probabilities (data not shown).

5.5 Review

After a brief discussion about the `correlated-columns` program in Section 5.1 I began to cover the sources of inputs for my neural network. I listed the previously discussed paired statistics and introduced two new ones, mutual information E-value and propensity, in Section 5.2. In Section 5.3 I covered the local structure alphabets. In Section 5.4 I discussed different input formats, including windows, raw values or modified values, binary inputs vs. real value inputs, summary statistics vs. vectors of values, and using the rank of a statistic as an input.

Chapter 6

Selecting Sources of Data for Inputs

Not only do I need to consider the selection of inputs, such as correlation statistics, but I need to consider the source of data used to calculate the statistics. Previous researchers have not always handled this issue well. Only one aspect of selecting sources of data is the actual selection of data sets, the set of sequences used in training, cross-training, and testing. Other aspects are the selection of the source of data are multiple sequence alignments, the predictors used for local structure prediction, and thinning of the MSA.

6.1 Selection of Sequence Datasets

Previous researchers have used SCOP [40] as a source, attempting to provide a selection of sequences that is representative of a wide variety of protein families as specified by SCOP. Some researchers would break the proteins into all-helical, all-beta, helical and beta strands combined (e. g. TIM barrels), and structures that are sepa-

rated into helical sections and beta sheets sections. These classes have rather arbitrary boundaries; a structure that is mostly helical except for a couple of strands could be included in the all-helical class or not.

I do not use either of these two methods of data selection. Instead, I emphasize using larger data sets and more accurate data sets. The use of large data sets on the order of 400 sequences can provide an effective coverage of different families of proteins.

The initial source of sequences comes from the Protein Data Bank and uses the culling of sequences provided by PISCES [96], a server that provides culled sets of sequences from the PDB by limiting the similarity between sequences and by using the accuracy of the experimental method of structure determination. I did not include NMR structures due to the low accuracy of their models. Inaccurate models are noisy and degrade the training of the neural networks. X-ray crystallography provides two measures for the accuracy: resolution measured in Ångstroms and the R-factor or free R-factor. Lower values for both should mean a better model. It is not unusual for the same protein to have been analyzed more that once with later models being more accurate. PISCES does an initial culling by eliminating duplication of structures using those two parameters. PISCES also allows a limit on sequence similarity; no two sequences can have more than a specified percentage of matching residues.

The PISCES server allows for specifying limits on resolution and R-factor. Kevin Karplus chose an initial set of 1335 proteins in March 2003 using resolution ≤ 1.8 Ångstroms, R-factor ≤ 0.25, and sequence similarity $\leq 30\%$. Several were removed due to excessive clashes in the structure, microheterogeneity, or shortness, leaving 1329

sequences.

I chose that set, and separated all sequences with unbroken backbones, leaving a set of 421 unbroken backbone sequences and another set of 908 with broken backbone sequences, some of which are used later for evaluation. I split the set of 421 arbitrarily into two sets: 350 for training and 50 for cross-training (the remaining 21 were simply set aside). A major limitation was the computer memory. Although there are not a lot of sequences involved, the data used represented millions of possible pairs as well as actual contact pairs. The input vectors to the networks are fairly long and the memory consumption was considerable. Even using the technique discussed in Section 4.2, the set of 350 training and 50 cross-training sequences used most of the memory available on a computer with four gigabytes.

6.2 Source of Multiple Sequence Alignments, Local Structure Predictions, and Amino Acid Distribution

In previous contact prediction research, the source of the multiple sequence alignment was only briefly discussed. However, I will argue that the poor result of contact prediction in CASP8 is the result of poor accuracy and coverage by multiple sequence alignments (see Section 12.4 and Chapter 13). By accuracy I mean the percentage of sequences in the alignment that are actual structural homologs, even if only locally, and by coverage I mean the percentage of the homologous sequences in the sequence database (typically nr) that are included in the multiple sequence alignment.

There are a several sources for multiple sequence alignments. The most commonly used by bioinformaticists is PSI-Blast. At UCSC we have an excellent program, SAM, for selecting and aligning sequences which has been under development for a decade [51, 41, 52, 55]. One of its strengths is the ability to find homologous sequences where the sequence identity is low. The version I am using for my predictions is SAM-T04 [55]

There are two companion programs to SAM. The first is PREDICT-2ND [49, 56] for predicting local structure alphabets (see Section 5.3). This program uses a sophisticated neural network which provides some of the most accurate local-structure predictions available as well as providing predictions for my choice of classification alphabets that I used as inputs. The second is ESTIMATE-DIST which regularizes the amino acid distribution for a column by using Dirichlet mixtures as a Bayesian prior [48, 89]. This will provide the amino acid distribution inputs for the contact prediction neural network as well as for PREDICT-2ND predictions. I note that a new method for amino acid distribution has been recently developed elsewhere [7] that may provide a better basis for amino acid distributions.

6.3 Limit on Minimum Number of Sequences in the Alignment

In CASP7 the assessors assumed that residue-residue contact predictions would be most important when the target sequences had no templates. Such target sequences

tended to have few sequences in their alignments.

While other researchers have set a minimum on the number of sequences when doing their contact predictions, I decided to have no minimum sequence count in either training or testing. This should reduce the accuracy but reflect real world conditions far better. The impact of having alignments with few sequences is discussed in Section 11.6.

6.4 Thinning of Alignments

When alignments are built, it is likely that some of the sequences may differ from each other by only a few residues. Pairwise sequence identity (percentage of residues identical between two proteins) provides a measure of similarity. When there are a lot of sequences in the multiple sequence alignment with identity greater than 90%, the conservation and/or correlation of two columns can be inaccurately represented. For example, if there are a lot of sequences with no changes of residue type in a column, statistical measures would interpret this as a highly conserved column. One way to avoid such false interpretations is to remove sequences until all pairs of sequences in the alignment are below an arbitrary identity similarity value. I define *thinning a multiple sequence alignment to N* to mean removing sequences until pairwise identity is less than N%.

In Section 8.2, I will examine the impact of different thinnings on paired statistics by considering thinnings of 35%, 40%, 62%, 70%, and 90% so as to determine an optimal thinning.

6.5 Review

In Section 6.1 I covered the selection of the protein sequences used for data input. In Section 6.2 I touched on the source of multiple sequence alignments, the amino acid distribution, and the program PREDICT-2ND which provides the local structure predictions. I specified that there would be no minimum number of sequences in the multiple sequence alignment in Section 6.3, and in Section 6.4 I discussed the thinning of alignments, the impact of which I show in Section 8.2.

Chapter 7

Evaluation Methods

In this chapter I discuss the various evaluations used for analysis of contact predictions. First, I cover the CASP assessors' evaluation and their rationale. Second, I cover the methods we use at UCSC for evaluation that includes a new method called *weighted accuracy* in addition to our own rationale.

7.1 CASP Evaluation Methods

The CASP assessment of contact predictions [33] differs significantly from the assessment used for secondary structure prediction. Secondary structure prediction is assessed for all residues; there are only L number of predictions where L is the length of the sequence, and the null model for EHL can provide 38% accuracy simply by classifying every residue as part of a helix. Furthermore, evaluation of local structure predictions deals with only one residue per prediction. Contact predictions deal with two residues, and the separation between them has a significant effect on both the predictive and the

background accuracy. As a result, CASP breaks the predictions down by using three different separations: ≥ 6, ≥ 12, and ≥ 24. CASP requires a probability associated with each prediction that suggests (perhaps roughly) how accurate the prediction is. CASP then breaks the predictions down further by considering the top fraction of predictions sorted by this value into sets of size L/10, L/5, and L/2 where L is the length of the sequence (Some CASPs also considered L, and 2L). Length of the sequence is used because the number of actual contacts is linear with respect to the length (approximately $6 \times L$). Again the smaller number of predictions is expected to have higher accuracy, although lower coverage. This results in the analysis of nine different sets of predictions.

The assessors use accuracy and coverage when evaluating each set. Accuracy is defined as T_p/T where T_p is the total number of correct predictions in a set and T is the number of predictions. Coverage is defined as T_p/T_a where T_a is the actual number of contacts in the structure.

CASP includes two more evaluations for each set: improvement over random, and a function called X_d. Improvement over random is the accuracy of the predictions over the accuracy of random predictions for the same criteria. Accuracy of random predictions is computed as the set of actual contacts with the required minimum separation over the number of possible contacts (residue pairs) with the required minimum separation.

The function X_d attempts to compensate for the assessments that evaluate contacts as binary. Instead, X_d evaluates predicted contacts using a distribution over the actual distance between residue pairs. To quote the assessors [33]:

"Xd represents an evaluation of the proximity of the predicted contacts rather than a direct evaluation of the physical contacts, that is, the difference between the distance distribution of the predicted contacts and the all-pairs distance distribution in the 3D target structure.

$$\mathrm{Xd} = \sum_{i=1}^{i=13} \frac{(\mathrm{P}ip - Pia)}{(di * 15)}$$

There are 15 distance bins covering the range from 0 to 60 . The sum runs for all the distance bins. di is the distance representing each bin, its upper limit (normalized to 60). Pip is the percentage of predicted pairs whose distance is included in the i bin. Pia is the same for all the pairs.

When the average distance between predicted residue pairs is less than the average distance between all residue pairs in the structure, Xd is > 0. Xd=0 shows no separations between the distance populations."

I will not be using Xd in my evaluations.

7.2 Our Evaluation Methods

I mainly use two basic methods for my evaluations, accuracy and weighted accuracy, both of which are represented as graphs rather than tables.

7.2.1 accuracy vs. predictions/residue

The accuracy method is a plot of the accuracy vs. predictions per residue. This results in more detail than simply providing results for only L/10, L/5, and L/2. The accuracy graph only provides results for separation ≥ 9. There are two reasons for this separation. Using 9 splits the difference between 6 and 12, and I found that restricting the training and testing of predictions to those with separation ≥ 9 actually provides overall a better predictor compared to one trained using separation ≥ 6 (results

not shown). I hypothesize this result is due to the significantly higher background probability in the range 6 to 9 (see Figure 7.1). The neural network ends up focusing on the predominance of good predictions in this range at the expense of predictions with greater separation.

Notice that the use of predictions per residue implies that for a fixed fraction of length there are more contact predictions for longer sequences included simply because of the length difference. For example, at 0.1 I will be plotting the accuracy that includes the first 40 predictions of a 400-residue-long sequence, but only 10 predictions for a sequence of 100 residues. I consider this approach acceptable since the longer sequences are the greater challenge for structure prediction.

One drawback of focusing on fraction of length for the graph is that coverage is not considered. As noted in CASP6, coverage can differ greatly from the fraction of length. For example, in CASP6, for the targets used for assessment, L/5 could represent coverages from 13.4% to 90.4% of actual contacts [33]. Figure 7.2 also shows the difference between coverage and length. Plots representing accuracy vs. percentage of coverage might yield useful insights.

Related to this issue of coverage vs. fraction of length is an ongoing problem for predictors in the difference between all-helical structure and those containing beta sheets. As shown in Figure 7.2, there is a significant difference in actual number of contacts with regard to the length between all-helical structures and mixed structures. This is likely the reason for the discrepancy in predictions for these two different types of structures. Anna Tramontano (personal communication) suggested that the fact

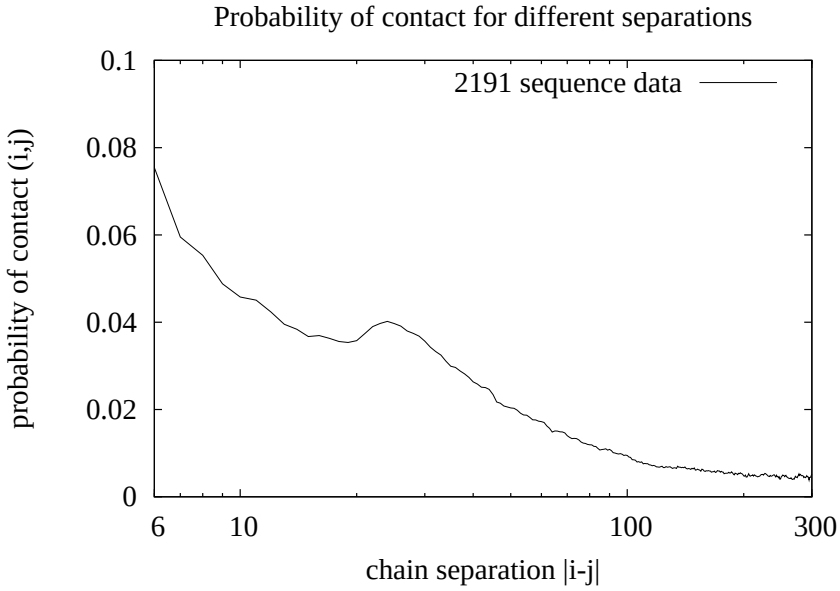


Figure 7.1: This is a plot of probability of contact vs. separation using a set of 2191 sequences. This is the basis for the new null model. Most evaluations of contact predictions do not take the impact of separation into account although almost all evaluations consider only contacts with separation ≥ 6. I limited this plot to those greater than 6 because the graph quickly increases to one for separation = 1. The brief increase near separation of 24 is likely caused by common beta-barrels where there are often separations of 24-26 for contacts along parallel beta sheets (see Figure 7.2).

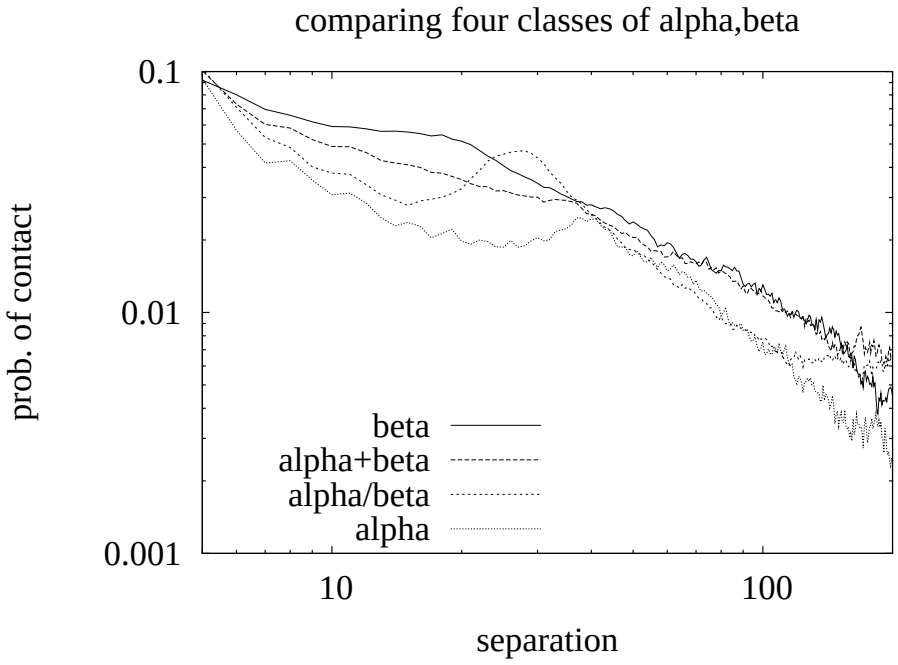


Figure 7.2: This is a plot of probability of contact vs. separation using four sets of proteins. The classes are alpha, proteins that are predominately α helices; beta, proteins that are predominately β sheets; alpha/beta, sequences where helices and strands alternate (as in β barrels); and alpha+beta, proteins where the helices and sheets are generally separated (possibly by domains).

that both residues for prediction are in helices should be included as another factor for probability weighting along with separation.

Let x be a given value for predictions per residue. Let P_s be the list of possible pairs for sequence s with separation ≥ 9, ordered by decreasing likelihood, and $P_s(x)$ be the first $\lfloor x * length(s) \rfloor$ pairs. Finally I define $P(x)$ as the set $\bigcup P_s(x)$ for $s \in S$. Then accuracy is calculated as:

$$\text{accuracy}(x) = \frac{1}{|P(x)|} \sum_{p \in P(x)} \text{contact}(p)$$

where $\text{contact}(p) = 1$ if the two residues of p are in contact and 0 otherwise.

Note that this is not the same as defining:

$$\text{accuracy}(s, x) = \frac{1}{|P_s(x)|} \sum_{p \in P_s(x)} \text{contact}(p),$$

and then averaging over those accuracies:

$$\text{accuracy}(x) = \frac{1}{|S|} \sum_{s \in S} \text{accuracy}(s, x).$$

The former calculation adds significant weight to longer sequences, but I am assuming that accurate predictions in longer sequences are more useful.

7.2.2 Weighted Accuracy

This second evaluation method is motivated by examining the null model assumed by the CASP assessors. Their null model assumes a constant background probability over all pairs greater than the specified least separation. They assess using this model for the three different sets using separations of 6, 12, and 24, as well as for the improvement over random.

However, it is possible to improve on random as defined by the simplistic null model by using a simple strategy. For all possible pairs, I assign the probability of contact for a pair p using Prob(contact|sep(p)) where sep(p) is the separation of the two residues of p along the sequence. When pairs with separation ≥ 6, for example, are selected as predictions based on this, the expected results improve on selecting pairs with separation ≥ 6 at random. We can repeat this for minimum separations of 12 and 24 as well. A new null model is needed.

My advisor suggested a new null model where the probability of separation for a prediction be used to weight that prediction should it be correct. I came up with a formulation that satisfied the requirement of being an effective null model. I implement the weighted accuracy for a specific pair p as:

$$\text{WtdAcc}_p = \frac{\text{contact}(p)}{\text{prob}(\text{contact}(p) \,|\, \text{sep}(p))}$$

The calculation of weighted accuracy then follows from the description of the original accuracy as shown above. Given an x representing predictions per residue, and a set of

sequences, S, weighted accuracy is calculated as:

$$\mathrm{WtdAcc}(x) = \frac{1}{|P(x)|} \sum_{p \in P(x)} \frac{\mathrm{contact}(p)}{\mathrm{prob}(\mathrm{contact}(p) \,|\, \mathrm{sep}(p))}$$

In this model, random predictions have an expected value of one. Also, the simple separation-based predictor discussed above also has an expected value of one. Accurate predictions with higher separation would become more significant, which is what we would want since models using such predictions may likely be better than models that simply consider easier low-separation predictions.

One other advantage of weighted accuracy is that it can be used to evaluate predictions using other definitions of contact. Since the definition of contact can change the background probability with regard to the separation, the weighted accuracy results will allow a better comparison between different definitions.

7.2.3 Other Evaluations

There are several other evaluation methods which are limited to CASP7 results. Since they are so specific, I explain those evaluation methods while covering the results from CASP7.

7.3 Pooled Predictions Evaluation

There are two basic approaches in evaluating predictions over a set of sequences. One is to evaluate using all the predictions as a set and sorting by the value.

The approach presented in Section 7.2.1 and Section 7.2.2 is to evaluate on a sequence-by-sequence basis and combine those results. I use both approaches when evaluating the individual statistics; however, I soon dropped the pooled predictions evaluations because they are overly sensitive to the examples for which pairs score high. Because it has not been possible to get uniform calibration in every example, pooled predictions end up putting most of the weight on just a few examples. The predictions will be used on a sequence-by-sequence basis and should be evaluated the same way.

7.4 Review

This chapter dealt with the evaluation of contact predictions, both by CASP and my myself. Section 7.1 covered the methods used by CASP. In Section 7.2 I discussed the plots I used extensively for evaluations. These used both accuracy and weighted accuracy; the latter uses a superior null model. I also mentioned that a few other methods are used and explained in later chapters. In Section 7.3 I demonstrated the two different ways to process predictions: per sequence and pooled predictions. I argued that the pooled predictions evaluation are inappropriate for the expected uses of residue-residue contact predictions.

Chapter 8

Evaluation of Individual Paired Statistics

In this chapter I evaluate the individual paired statistics to determine which ones provide the best predictions. Then I evaluate the best statistics at different thinnings to determine which thinning(s) to use as inputs.

Because this phase of development of the neural network predictor does not require different sets for training, cross-training, and evaluation, I use the whole set of 1329 proteins including those with broken chains for evaluation.

8.1 Individual Comparison Using Accuracy and Weighted Accuracy

I compare correlation coefficients, weighted correlation coefficients, joint entropy, mutual information, mutual information/joint entropy, OMES, mutual informa-

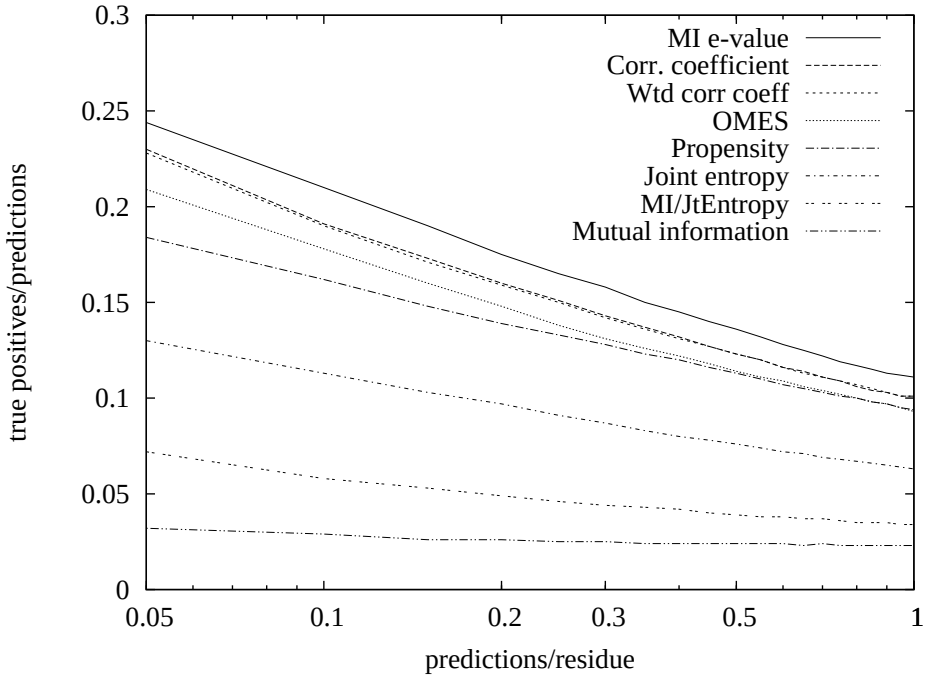


Figure 8.1: This graph shows the plot of accuracy vs. number of contact predictions for the eight paired statistics discussed in Section 8.1 using 1329 sequences. Thinning 62% is used for this plot. The key is ordered from best to worst accuracy at 0.1 predictions/residue.

tion E-value, and propensity. I plot both accuracy and weighted accuracy for both per-sequence and pooled results. Initially, I fixed the thinning at 62% only because this percentage was used in the BLOSUM62 matrix [37].

In Figure 8.1 and Figure 8.2 there are roughly three groups: mutual information and MI/joint entropy are at the bottom, joint entropy is in the middle, and the other statistics are at the top. What stands out in the figures are the poor results for mutual information. Martin and Gloor [62] used joint entropy to offset the problem that mutual information has with conservation. Their paper showed that mutual information over joint entropy is better than mutual information alone. However, Figure 8.1

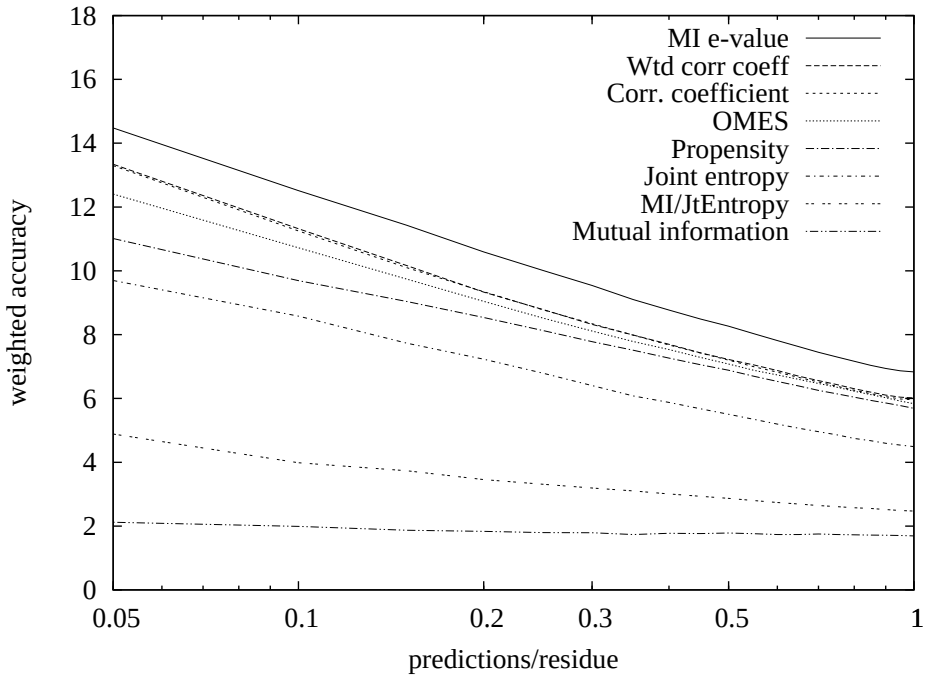


Figure 8.2: This graph shows the plot of weighted accuracy vs. number of contact predictions for the eight paired statistics discussed in Section 8.1 using 1329 sequences. Thinning 62% is used for this plot. The key is ordered from best to worst accuracy at 0.1 predictions/residue. Unlike standard accuracy, weighted accuracy values are not limited to a range of 0 to 1.

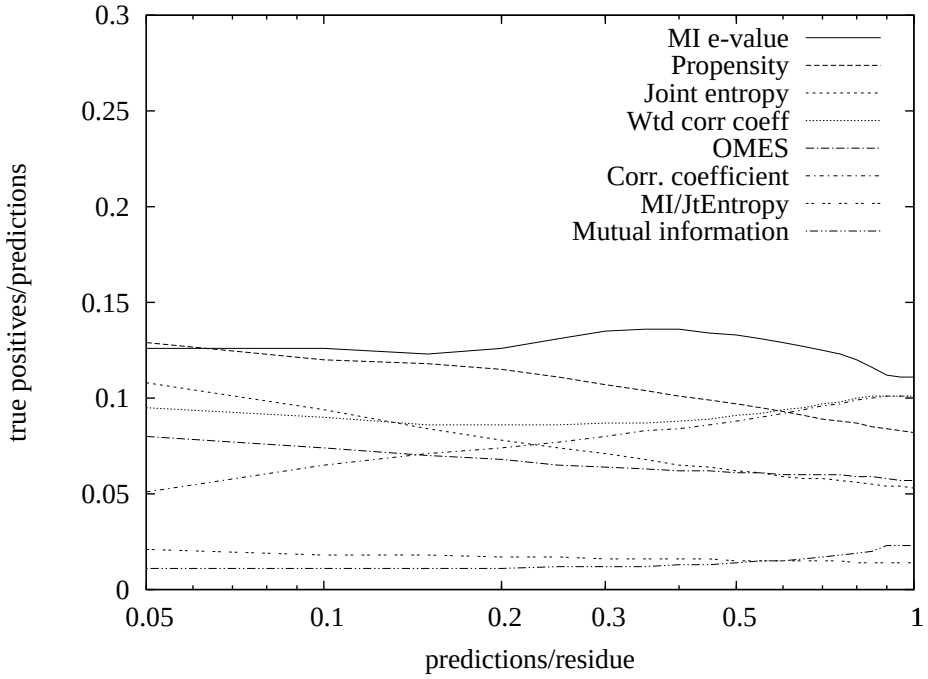


Figure 8.3: This graph shows the plot of accuracy vs. number of contact predictions for the eight paired statistics discussed in Section 8.1 using 1329 sequences. This plot pools all predictions across all sequences and is a value-based result rather than a rank-based result. The key is ordered from best to worst accuracy at 0.1 predictions/residue.

shows that joint entropy is even better by itself. It appears that the effectiveness of joint entropy is adversely affected by the use of mutual information, consequently, I decided to discard both mutual information and MI over joint entropy.

I further reduce the group of statistics used in later experiments by removing the joint entropy statistic. The statistic performs well but not as well as the top five statistics in Figure 8.1. I will later consider the inclusion of joint entropy in Section 10.2.4.

The pooled-predictions graphs, Figure 8.3 and Figure 8.4, show that the actual statistical values are uncalibrated with respect to different sequences. In fact, the higher

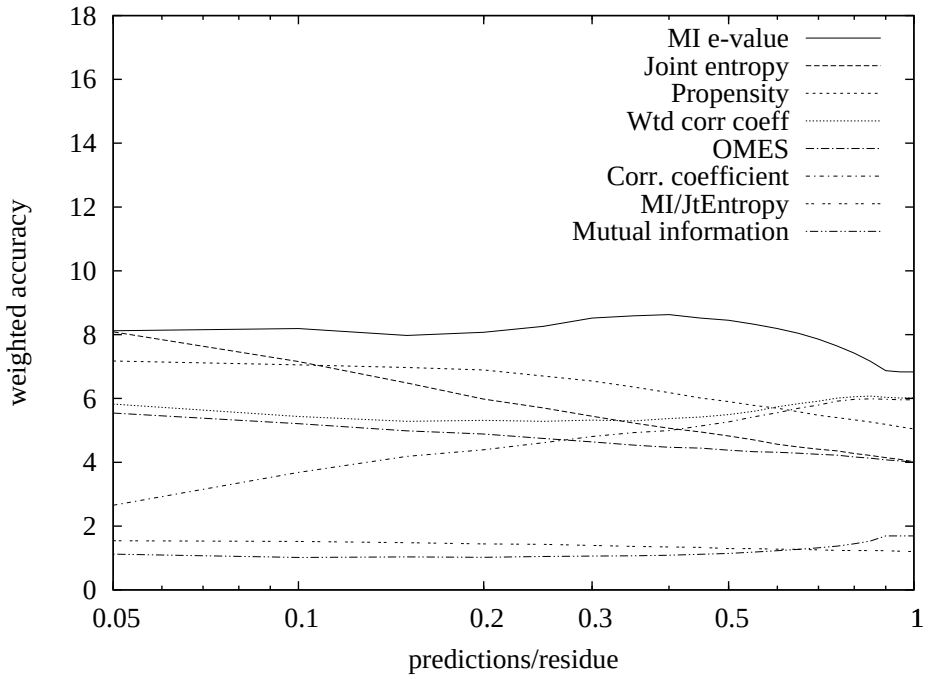


Figure 8.4: This graph shows the plot of weighted accuracy vs. number of contact predictions for the eight paired statistics discussed in Section 8.1 using 1329 sequences. This plot pools all prediction values a cross all sequences and is value-based rather than the rank-based results. The key is ordered from best to worst accuracy at 0.1 predictions/residue.

values may be worse predictions than the lower values because a target sequence with poor predictions may have all scores higher than a target with good predictions. The earlier graphs (Figure 8.1 and Figure 8.2) show that the ranking of prediction values in the context of individual sequences does show calibration, and earlier graphs suggest that the rank of the statistic provided on a per sequence basis is a better indicator as a prediction than simply its value. This provides an initial answer to the question of which to use as input that was raised in Section 5.4.2; rank appears better than raw value in correlating with accuracy.

8.2 Comparison Using Different Thinnings

As discussed in Section 6.4, I examine the effect of different thinnings on the accuracy of predictions (see Figure 8.5 and Figure 8.6). I looked at thinnings limiting pairwise sequence identity to $\leq$ 90% 70%, 62%, 40%, and 35%. I expected that thinnings close to 90% would suffer from over-representation of similar sequences which can confound the statistics. On the other hand, too much thinning may remove too much data, resulting in degradation.

Figure 8.5 shows that thinning of 90% is well below the others in accuracy and weighted accuracy; thinning of 35% is generally worse than the others. I should note that I used the set of 1329 sequences and that this set is not expected to be representative of the more difficult targets that are used for assessment of contact predictions in CASP. Since the thinning of 40% is consistently the best for accuracy with one exception where

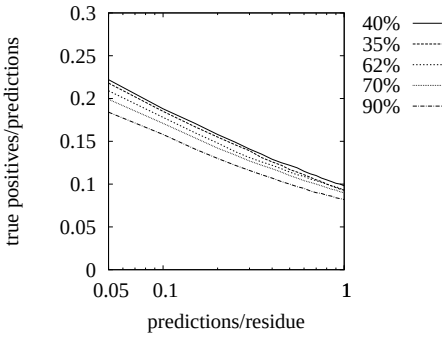
62% is the best, I decided to split the difference and use a thinning of 50% for the next major predictor. I also recognize that these more difficult targets are likely to have fewer sequences in their multiple sequence alignments and a thinning of 50%, while not quite as good as 40%, is still much better than 90% and leaves more sequences in the alignments, which is desirable.

For mutual information E-value, I find that a thinnings of 62% and 40% provided the best results. The difference is not large, but clearly the trend suggests that too much thinning or too little results in loss of accuracy.

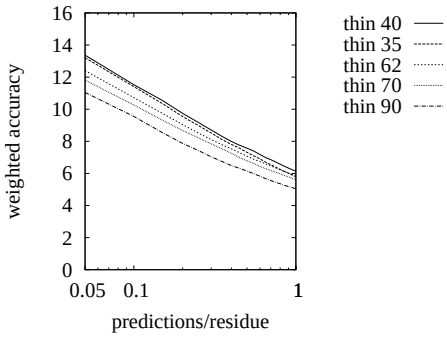
Figure 8.6 shows the results of thinnings for both the weighted and unweighted correlated coefficients. The results for both are almost identical except weighted correlation coefficient does better when the thinning is 90%. While this indicates that the weighting does help compensate for over-representation, my decision to not use thinning of 90% and the extra computation needed for the weighted version prompted me to discard the weighted correlation coefficient in favor of the unweighted version. Göbel, *et al.* [31] also expressed doubts about the effectiveness of weighting due to their own results.

8.3 Review

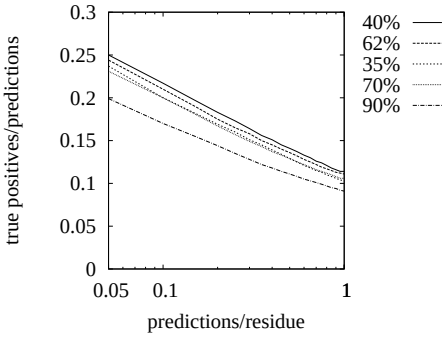
In Section 8.1 I compared a set of paired statistics using their individual values. I continued to research the best six statistics from that comparison in Section 8.2. I examined the effect of different thinnings and found that a thinning of 90% was the



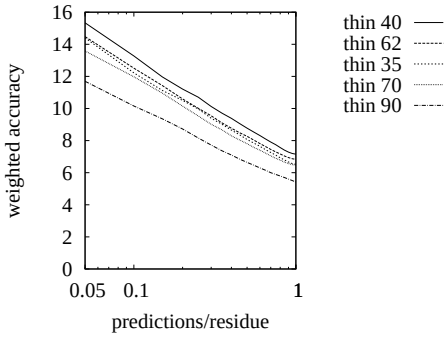
(a) thinnings for OMES (std acc)



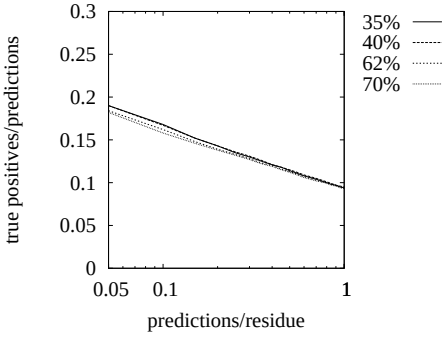
(b) thinnings for OMES (wtd acc)



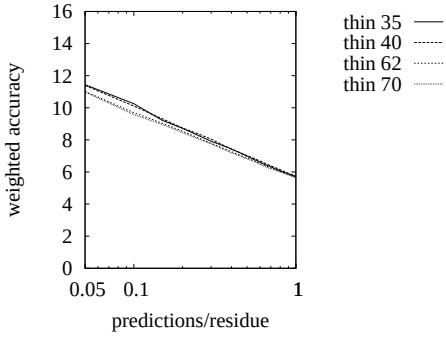
(c) thinnings for MI E-values (std acc)



(d) thinnings for MI E-values (wtd acc)

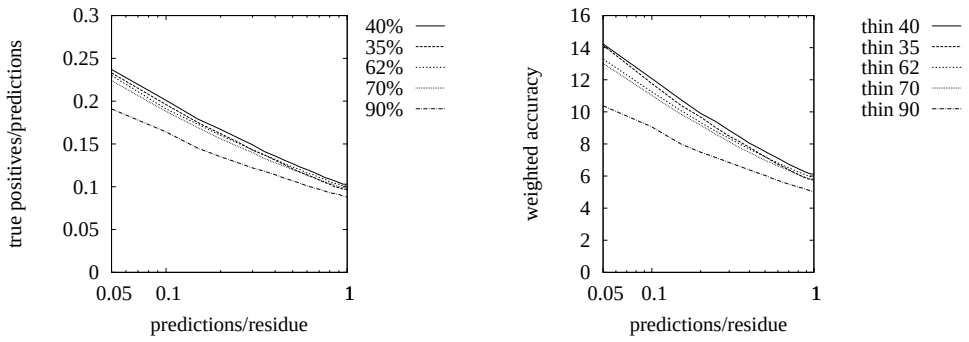


(e) thinnings for propensity (std acc)

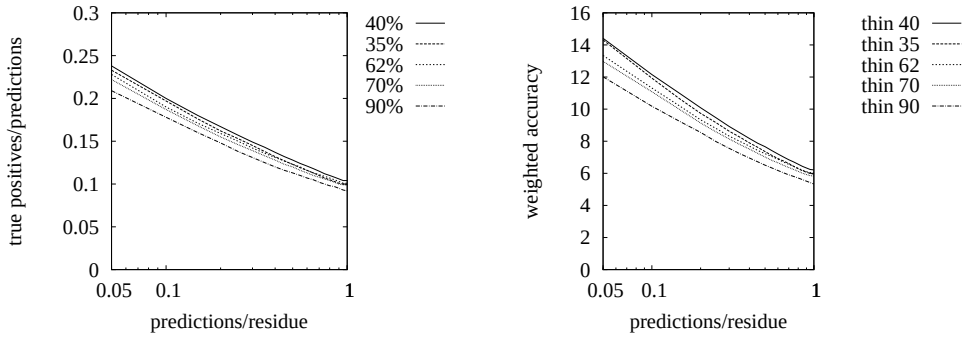


(f) thinnings for propensity (wtd acc)

Figure 8.5: I compare three statistics, OMES, MI E-value, and propensity using different thinnings of 90%, 70%, 62%, 40% and 35%. Thinning means sequences were removed from the alignment until no pairwise identity of sequences exceeded that percentage. The plots are for standard accuracy and weighted accuracy. The keys are listed in the order from best to worst using 0.1 predictions per residue.



(a) thinnings for correlated coefficient (std acc) (b) thinnings for correlated coefficient (wtd acc)



(c) thinnings for wtd. corr. coef. (std acc) (d) thinnings for wtd. corr. coef. (wtd acc)

Figure 8.6: I compare two statistics, weighted and unweighted correlation coefficients using different thinnings of 90%, 70%, 62%, 40% and 35%. Thinning means sequences were removed from the alignment until no pairwise identity of sequences exceeded that percentage. The plots are for standard accuracy and weighted accuracy. The keys are listed in the order from best to worst using 0.1 predictions per residue.

worst. I decided on using a thinning of 50% for the next stage.

Chapter 9

The CASP6 Neural Network

In this chapter I discuss the neural network built for CASP6 including the inputs, the training and cross-training data sets, the results for CASP6, and a comparison of the neural network with the best individual paired statistics. Though the results of CASP6 were inconclusive, the CASP6 neural network provided a "proof of concept" for the direction of my research.

The timing of the CASP6 experiment constrained the development of this initial neural network. Some of the early results came about two months before CASP6 started. Individual statistics would not have been good enough; we needed an effective neural network. Needless to say, we continued to use the CASP definition of contacts (see Section 2.5).

I did have time to try the use of summary statistics as an input for the burial alphabets but found that they did not perform as well as the full distribution vector (see Section 5.4.5).

9.1 Network Development

The CASP6 neural network used `lwnn` [93] as the initial source code for the program `traincontactnn`. The program `traincontactnn` was the first of the two programs I developed and used for training and generating the predictions of neural networks. The code was not limited to using a single hidden layer. However, including additional hidden layers does not extend the computational power of neural networks [22], so I restricted myself to a single hidden layer. Later I use `fann` [68] which includes RProp back-propagation as the initial source code for developing `predictlocal`, the training and prediction program I use in all later research.

9.1.1 Inputs

The inputs for the CASP6 network included both local structure alphabets and a paired statistic. I thinned the alignments to 62% due to some of the analysis discussed in the previous chapter. The selection of the two burial alphabets and the STR2 alphabet was largely based on results of predictability discussed in Section 5.3.1 and Table 5.1. The combination of burial and STR2 secondary structure alphabet also classifies very different properties. I hoped that they would provide complementary inputs. I realized that using two burial alphabets was probably redundant; however, I had not determined which of the two might be better.

Other inputs were summarily selected; the length of the sequence and the separation are included as log values (see Section 5.4.3). For the amino acid distribution,

I used `estimate-dist` which uses Henikoff relative weighting [38] to add 0.5 bits of information per position. The choice of 0.5 bits is a result of earlier research showing its effectiveness in fold-recognition using SAM.

For all alphabets and for the amino acid distribution I used a window 3 residues wide around residues i, j (see Section 5.4.1). The choice of the window size was arbitrary.

I had not finished the analysis of thinning before I chose the inputs for the CASP6 network, so I entered several different thinnings. This created a limitation. By removing sequences, thinning with lower percentages could result in fewer available pairs of columns for consideration because removing sequences will reduce the number of residues in some of the columns to the point where only the original residue in the target is left. This effectively eliminates that column from being used for paired statistics (although we can still calculate propensity). This, in turn, limited all other statistics to this smaller set of pairs of columns since I had no proper way to deal with missing data. For CASP6 I used MI E-values as the sole paired statistic, but used thinnings of 62%, 40%, 35%, and 30%. This limited the pairs of columns to those available to thin 30%. As mentioned in the previous chapter, the difference of thinnings below 62% is not that significant and later predictors use only the one thinning of 62%. Besides using the pair i, j for the MI E-value input, I arbitrarily included paired statistics for $(i-1, j-1)$, $(i-1, j)$, $(i, j-1)$, $(i, j+1)$, $(i+1, j)$, and $(i+1, j+1)$ as inputs as well.

9.1.2 Training

There were 280 inputs. I set the size of the hidden layer to 240 nodes which is considerably more than necessary and slowed down the training. Again, this was an arbitrary choice and later networks use smaller hidden layers.

The data sets used for training came from the set of 400 unbroken chains. The first 200 of the four hundred were used for training, the next 50 were used for cross-training, and the last 150 were used for testing. Every four epochs, the program ran the cross-training and stored a copy of the neural network on the hard drive. I manually inspected the cross-training results to select the final choice of the stored networks.

9.2 Results and Discussion

I submitted predictions to CASP6. However, I arbitrarily set a cutoff limit for predictions, submitting only those predictions whose projected probabilities were above 18%. I later found out that the evaluation for CASP6 required a minimum number of predictions with separations ≥ 6, ≥ 12, and ≥ 24. My cutoff limit resulted in too few predictions being submitted, and I failed to qualify for evaluation.

Note that MI E-values using different thinnings are part of the input; I used no other paired statistics. The use of two different burial alphabets is an example of overlapping statistics; I discard BURIAL-14-7 in later networks.

I further discovered that a bug in `traincontactnn` had limited the STR2 alphabet to two summary statistics. Summary statistics are meaningless for the classifications

in STR2 and their input was almost certainly worthless!

Nevertheless, at that time a comparison of the CASP6 network to individual statistics showed a significant improvement in predictability (data not shown). This provided incentive to my advisor and myself to continue doing research in contact predictions.

9.3 Review

This chapter covered the CASP6 predictor, the predictor's inputs, its training, and the results of CASP6. Section 9.1 described the details of the inputs and the training, while Section 9.2 covered the unfortunate results of CASP6. Nevertheless, the predictor demonstrated enough good results; I continued to do more research on contact predictions.

Chapter 10

Matching Paired Statistics

This chapter covers research to determine which paired statistics should be included in a neural network and whether or not to use rank instead of value (see Section 5.4.2). The intention of the research is to avoid inputs that do not improve the predictability of the network as well as to determine the best form of input. The CASP6 network included only MI E-value as a paired statistic. From the results reported in Chapter 8, I had four different paired statistics to consider in combination as inputs. Along with the raw value and rank of the statistic's value, I also decided to analyze the z-value defined as $\frac{value - \mu}{\sigma}$ where μ is the mean of the statistic's values with respect to the sequence, and σ is the standard deviation.

There are several other issues discussed earlier that now become relevant. This research and later research limits the thinning to 62%; the rationale was discussed in Section 8.2. The new program, `predictlocal`, stops training automatically when it determines that overfitting is occurring (this is discussed in detail in Section 4.4). Prior

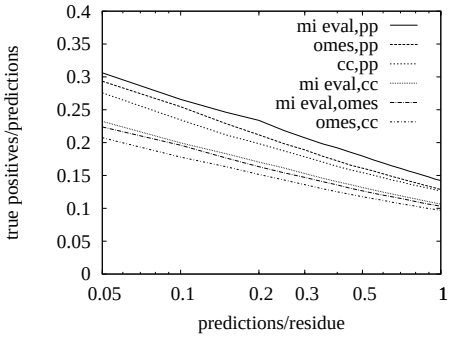
to this, I manually determined the best stopping point. Finally, I note that size of the hidden layers used in this chapter and later are much smaller than the size of the hidden layer of the CASP6 neural network.

10.1 Neural Network Inputs and Training

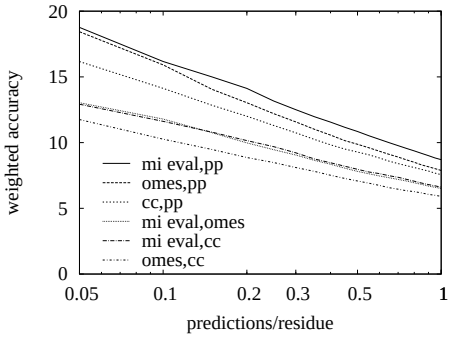
The training used the set of 400 unbroken sequences described in Section 6.1 with a cross-training set using broken sequences indexed 1 to 100. The final evaluation used 300 broken sequences indexed from 301 to 600.

The inputs for the small neural networks used for matching two paired statistics are sequence length, statistic 1 input, and statistic 2 input. The size of the hidden layer is two. Since I am using four statistics, MI E-value, propensity, correlated coefficient, and OMES, I will be comparing $\binom{4}{2}$, or 6, small networks.

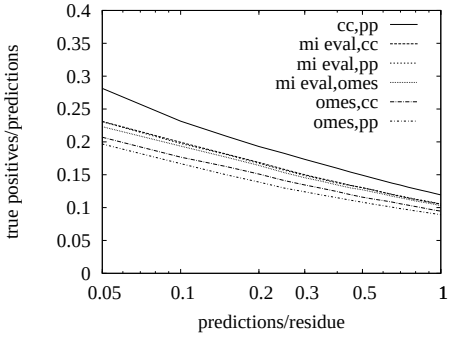
Note that I did not specify the input type for the statistics. I am also going to compare three different types of inputs: value, z-value, and log(rank). I will do each of these in separate comparisons rather than have 24 comparisons in one graph. Furthermore, I do not include separation as an input. Using separation as an input will be examined in Section 10.2.3.



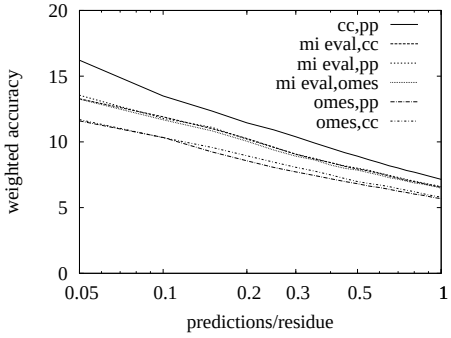
(a) Log(rank) Input, Standard accuracy



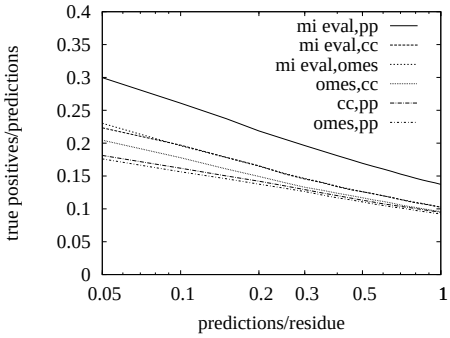
(b) Log(rank) Input, Weighted accuracy



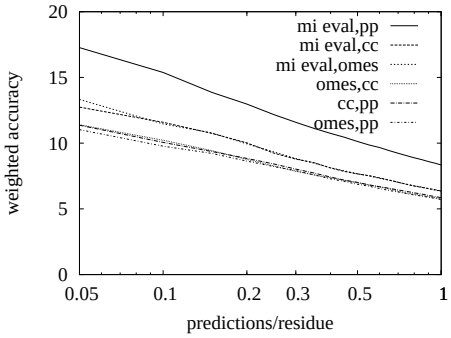
(c) Value Input, Standard accuracy



(d) Value Input, Weighted accuracy



(e) Z-value Input, Standard accuracy



(f) Z-value Input, Weighted accuracy

Figure 10.1: I compare small neural networks having pairs of statistics plus the log of the sequence length as inputs. Labels are pp) propensity, mi eval) MI E-value, cc) correlation coefficient, and omes) OMES. The results cover standard and weighted accuracy. Keys are listed in order from best to worst using 0.1 predictions per residue. Each row represents differing input formats for the statistics: the first, log(rank), the second, value, and the third, z-value (see Section 5.4).

10.2 Results

10.2.1 Comparison of Value, Z-value, and Rank

Figure 10.1 shows the results of pairing paired statistics. Furthermore, the plots show the results of using value, rank, and z-value as inputs. First I compare the differences caused by different input formats. Between rank and value it is clear that rank produces overall better results. At 0.10 predictions per residue the rank has three lines above 0.24 while value has all but one below 0.21; rank has the best overall results. The results in weighted accuracy matched those of accuracy and show even stronger results for rank. Similarly, rank improves on z-value. Rank becomes the obvious choice of input formats.

10.2.2 Comparison of Pairs

Given that rank is the best input format, the results shown in Figure 10.1 find that for both accuracy and weighted accuracy the best pair is MI E-value and propensity. The conclusion is to use MI E-value and propensity as the best pair.

Propensity is also part of the best three pairs. Possibly this is because propensity provides an indication of burial in the core of the protein, resulting in greater likelihood of contact. But if propensity is only an indication of both residues being in the core, I would expect that the separation for such contact pairs would generally be smaller than contact pairs predicted by other pairs of statistics that do not include propensity. Nevertheless, the weighted accuracy, which scores higher for contact pairs

with greater separation, has propensity as one of the statistics in the best three pairings.

10.2.3 Impact of Including Separation

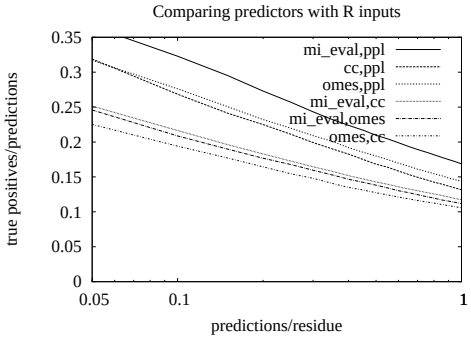
The first set of small neural networks did not include the separation as an input. I examined the results of including this in Figure 10.2. The results show a dramatic increase in accuracy for the MI E-value and propensity pairing, and for rank input in general. However, weighted accuracy drops for all results. This should not be surprising. The small networks learn that pairs of residues with large separation are more unlikely to be in contact and that smaller separations should be given higher neural network scores.

Again, the same pairing of MI E-value and propensity perform the best. These two joined separation as basic inputs for the CASP7 predictor.

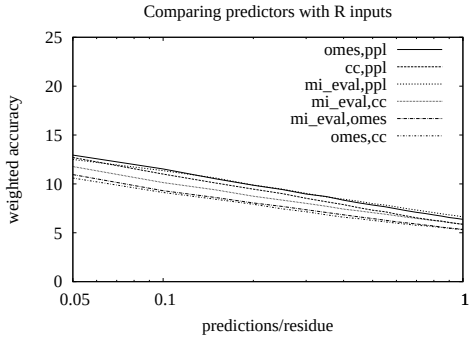
While I feel that the weighted accuracy results should be the more important results, CASP6 and later CASPs have put most of the weight on accuracy with a modest nod to separation by evaluating three different sets using separations of ≥ 6, ≥ 12, and ≥ 24. With this in mind, I decided to include separation as one of the inputs for all later CASP predictors.

10.2.4 Inclusion of Joint Entropy

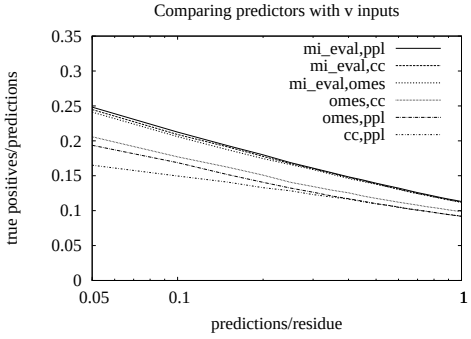
Professor Karplus suggested including joint entropy as an extra paired statistic along with MI E-value and propensity. Joint entropy is one of the two statistics in the second best performing group of individual statistics (see Figure 10.3). I used two



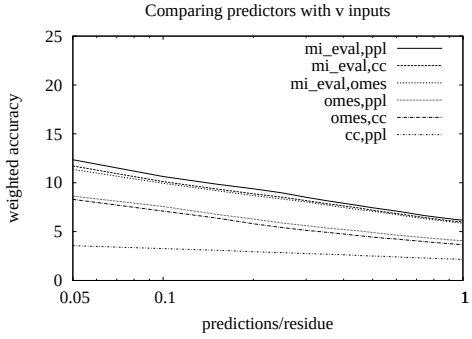
(a) Log(rank) Input: Standard accuracy



(b) Log(rank) Input: Weighted accuracy

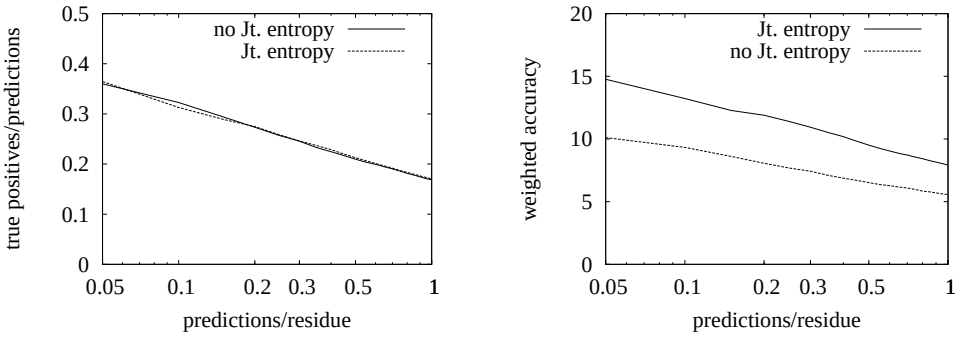


(c) Value Input: Standard accuracy



(d) Value Input: Weighted accuracy

Figure 10.2: These plots show the inclusion of separation as an input. I compare small neural networks having pairs of statistics plus the log of the sequence length and the log of separation as inputs. Labels are pp) propensity, mi eval) MI E-value, cc) correlation coefficient, and omes) OMES. The results cover standard and weighted accuracy. Keys are listed in order from best to worst using at 0.1 predictions per residue. Each row represents different input formats for the statistics: the first is log(rank) of the statistics, the second is the value (see Section 5.4).



(a) Standard accuracy

(b) Weighted accuracy

Figure 10.3: Two small neural networks are tested. Both have MI E-value and propensity inputs; One includes the paired statistic joint entropy and the other does not.

small neural networks as before except I arbitrarily increased the hidden layer to three. They are trained using 400 unbroken chains, cross-trained with 100 broken chains, and evaluated with a separate set of 300 broken chains.

While there is no improvement in standard accuracy, Figure 10.3 shows substantial improvement in weighted accuracy; this implies that including joint entropy helps detect contacts with greater separation while maintaining the same overall accuracy. That is sufficient to include joint entropy as an input. I note that no testing is done for value or z-value.

10.3 Review

This chapter dealt with determining which pairs of paired statistics had the best accuracies. I discussed in Section 10.1 how I designed a small neural network to combine the statistics. I also wanted to determine the best input format. In Section 10.2 I found that rank as input provided the best results. Based on this I found that MI

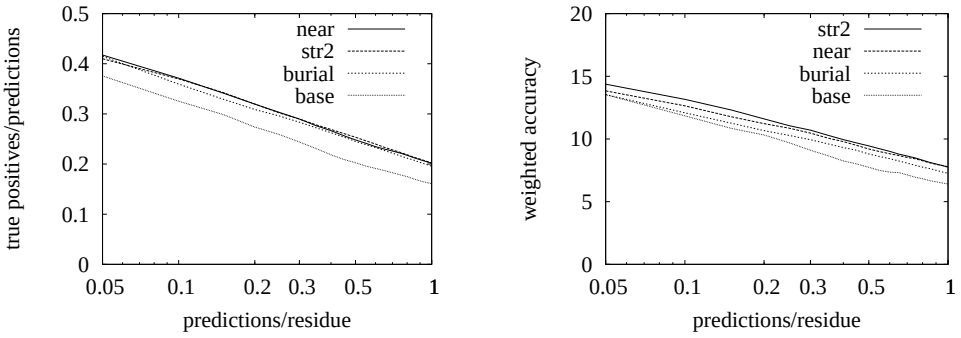
E-value and propensity were the best pair. Finally, I tested using log(separation) as an input. While the addition of separation as input resulted in lower weighted accuracy, it made a significant improvement to the standard accuracy. Considering the assessment methods of CASP, I chose to include separation as an input.

Chapter 11

Local Structure Inputs

Having established the choices of paired statistics, I focus on the selection of local structure inputs. Including the amino acid distribution is an obvious choice to consider as is including the entropy of amino acid distribution at both columns. For local structure predictions as inputs, I consider three local structure alphabets: STR2, NEAR-BACKBONE-11, and BURIAL-14-7.

A problem with including multiple alphabets while using windows is the rapid increase in the number of inputs. The use of both burial alphabets and the STR2 alphabet with windows of size 3 require 186 inputs. Having a large set of inputs can result in poorer predictions and a significant increase in training time. Therefore, I work to limit the number of alphabets unless results demonstrate otherwise.



(a) Standard accuracy

(b) Weighted accuracy

Figure 11.1: These plots show the results of the the three neural networks discussed in Section 11.1. Each network includes one of STR2, BURIAL-14-7, or NEAR-BACKBONE-11. The fourth consists of only the basic neural network and is included only for comparison. The keys are listed from best to worst using 0.1 predictions per residue.

11.1 Testing of Three Alphabets

I use a medium size neural network with log(sequence length), log(separation), amino acid distribution (window size 3), entropy for each column (no window) and the paired statistics: joint entropy, propensity, and MI E-value. The initial choice of window size is arbitrary; I will later test different window sizes (see Section 11.3). To this base of inputs I include one of three alphabets, STR2, BURIAL-14-7, or NEAR-BACKBONE-11, using a window of 3. The results are shown in Figure 11.1. The hidden layer is size 11. I train on 400 unbroken chains, and cross-train using the first 100 sequences of the broken chains. Final evaluation uses the last 300 sequences of the broken chains.

11.2 Results

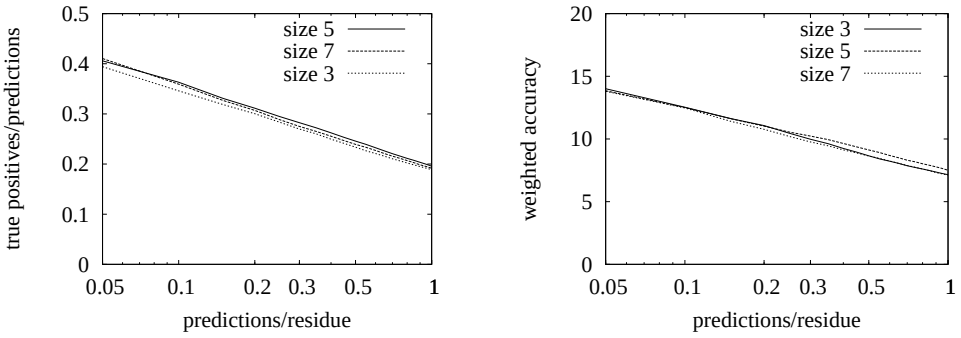
The results (see Figure 11.1) find that STR2 performs the best, with NEAR-BACKBONE-11 either second or tied for second. The actual differences are quite small. Based on this, I decided to use STR2 and NEAR-BACKBONE-11 as the two local prediction alphabets. Since BURIAL-14-7 also measures the same property as NEAR-BACKBONE-11, I am anticipating that including BURIAL-14-7 provides no additional complementary information.

11.3 Testing Different Window Sizes

The use of a window size of 3 for the local alphabets, amino acid distribution, STR2, and NEAR-BACKBONE-11, is arbitrary. Some informal experiments indicated that a window of 3 is better than a window of 1. To determine an optimum size, I increase the basic neural network discussed in Section 11.1 by including both STR2 and NEAR-BACKBONE-11. I arbitrarily select and examine three different window sizes: 3, 5, and 7 for the three local alphabets. The results are shown in Figure 11.2. The training, cross-training, and evaluation sets are unchanged from the previous section.

11.4 Results of Different Window Sizes

The results shown in Figure 11.2 show that a window size of 5 nominally provides the best predictions. I also assume that the behavior is uni-modal with size 5 performing better than size 3 and size 7. Based on this, I am setting the window size



(a) Standard accuracy

(b) Weighted accuracy

Figure 11.2: These plots show the results of the the three neural networks discussed in Section 11.3. The keys are listed from best to worst using 0.1 predictions per residue.

for the CASP7 network to 5.

11.5 The CASP7 (449a) Predictor

The results of the previous work provides the basis for the inputs and their formats for the CASP7 predictor. I started with log(sequence length) and log(separation). For local alphabets I included AA, STR2, and NEAR-BACKBONE-11, all with a window of 5. I added entropy for the amino acid distribution for both i and j, with a window of 1.

For paired statistics I included joint entropy, mutual information E-value, and propensity. The format used was log(rank). I decided to arbitrarily include the z-value of MI E-value as well. I also included the value 1/(no. of pairs). A low number of pairs can indicate to the network that the paired information is likely to be weak.

The final number of inputs for the CASP7 predictor was 449. This was both larger and more carefully selected than the CASP6 predictor. Because I submitted too

few predictions for CASP6, I submitted far more predictions than needed for CASP7; there was no penalty for submitting extra predictions. The submissions were part of the Karplus lab participation in CASP7.

11.6 CASP7 Results

The 449a predictor was assessed as the best residue-residue contact predictor at CASP7.

Figure 11.3 compares the CASP7 network using the final evaluation test set of 300 broken sequences with single pair features and the best neural network that combined two pair features. The extra inputs of the CASP7 network make a substantial improvement in accuracy.

11.6.1 Comparison of CASP7 Predictions to Tertiary-based Predictions

CASP7 used the two main categories of difficulty, template-based modeling (tbm) and free-modeling (fm), as discussed in Section 2.4. The initial targets were chains; for evaluation the assessors split these into domains. Only free-modeling domains were assessed for residue-residue contact predictions; however, I consider both categories in this analysis.

I compared the accuracy of my predictions to those of the group (Baker lab) which achieved the best over-all results in free-modeling predictions. I derived contact predictions from their tertiary model by taking all pairs of residues in their model with

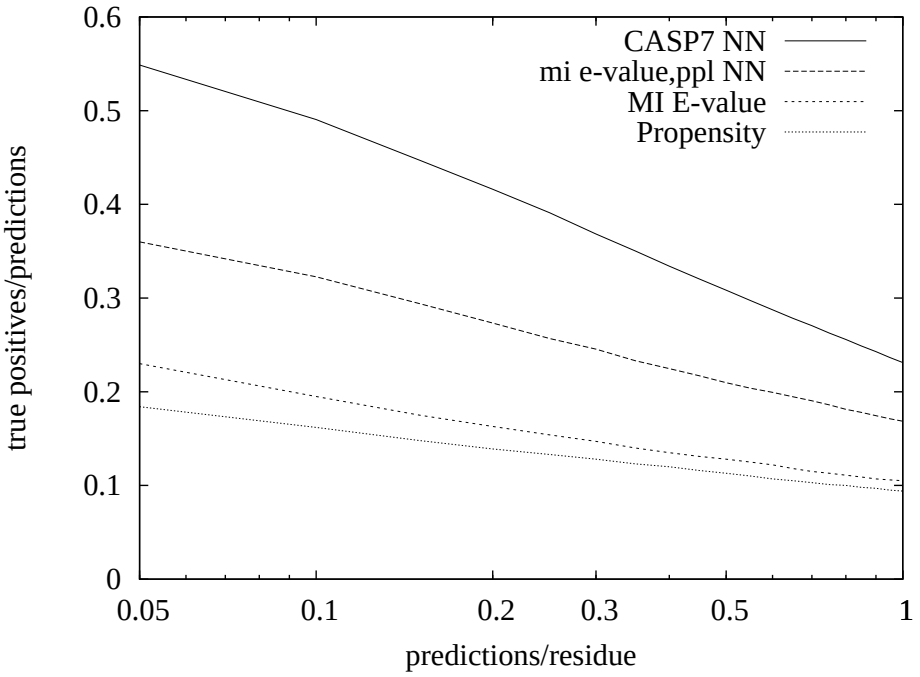


Figure 11.3: The plot shows accuracy vs. prediction for the full cross-validation test. The final CASP7 predictor uses 449 inputs compared to the next highest which uses only 3 inputs: propensity (log(rank)), mutual information E-value (log(rank)), and log of separation. The accuracy for the individual paired statistics is included for comparison.

separation greater than 8, and sorting them in increasing residue-residue distance. Then I took the top $L/10$ pairs from their pairs and compared them to the accuracy of the top $L/10$ pairs from my predictions. I did so for both template-based and free-modeling. The results are shown in Figure 11.4. As expected, the predictions from template-based modeling did much better than my own, but my predictions for free-modeling did better.

There are a number of instances where my contact predictions did better than the derived contact predictions. Possibly this is because some of the derived predictions fell between residue pairs that are not in the same template. Tertiary modeling may use partial templates, and the orientation of these partial templates is not necessarily known. Good contact predictions may help in these instances. However, this comparison does not do an effective job of assessing how useful (or useless) contact predictions may be in improving free-modeling tertiary predictions. In Chapter 13 I discuss a *value-added* assessment that should provide a better comparison.

While my predictions for free-modeling targets are generally better than the tertiary-structure-based predictions, they are worse than the contact predictions for template-based targets (see Figure 11.5). This plot also shows that when free-modeling predictions are restricted to multiple sequence alignments with 15 or more sequences, the results can be comparable to the template-based targets. This impact of size of the multiple sequence alignment continues to be significant especially considering the results of CASP8 (see Section 12.4). The point is not that I need to find more sequences to find templates, but simply that I need to find more sequences to generate better predictions. This underscores the need for more and better sequences in the alignments where those

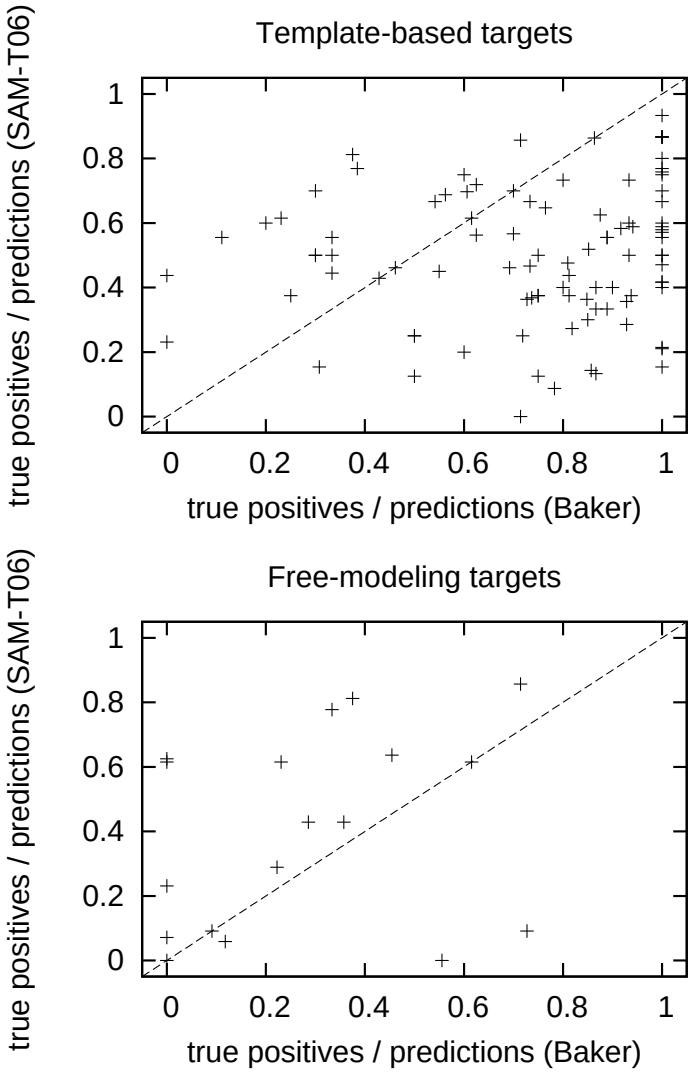


Figure 11.4: Two scatter-plots comparing contact predictions by my neural network compared to contact predictions drawn from the Baker group model 1. The diagonal line separates predictions where I do better (above the line) compared to when the Baker group does better. The first plot is for targets considered template-based by the assessors. The second plot is for the more difficult free-template targets. Both sets are limited to the best 0.1 predictions per residue and separation ≥ 9. SAM-T06 predictions improved on Baker 12 of 18 targets or 67%.

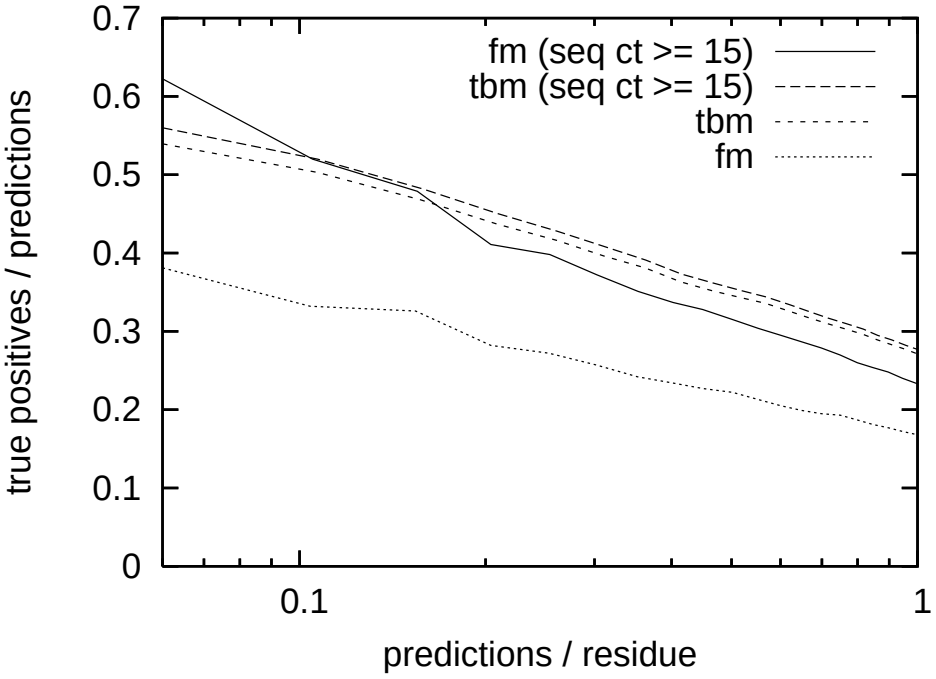


Figure 11.5: I plot accuracy vs. predictions per residue for free-modeling (fm) and template-based modeling (tbm). The difference in quality appears to be due to the number of sequences in the multiple alignments. When I restrict my predictions to only those targets that have 15 or more sequences in the multiple sequence alignment, the difference in accuracy between free modeling and template-based modeling domains almost disappears.

sequences are drawn from the non-redundant database and not from the PDB.

11.7 Review

The development of a predictor for CASP7 continued in Section 11.1 where I tested four alphabets, STR2, STR4, NEAR-BACKBONE-11, and BURIAL-14-7. I continued to use STR2, and I selected NEAR-BACKBONE-11 of the two burial alphabets in Section 11.2. From the results in Section 11.4, I decided on a window size of 5 for the alphabets in the new predictor. The final specifications for the CASP7 predictor were covered in Section 11.5. Finally, in Section 11.6 I reported the success at CASP7 and concluded with a favorable comparison to derived contact predictions based on tertiary structure predictions.

Chapter 12

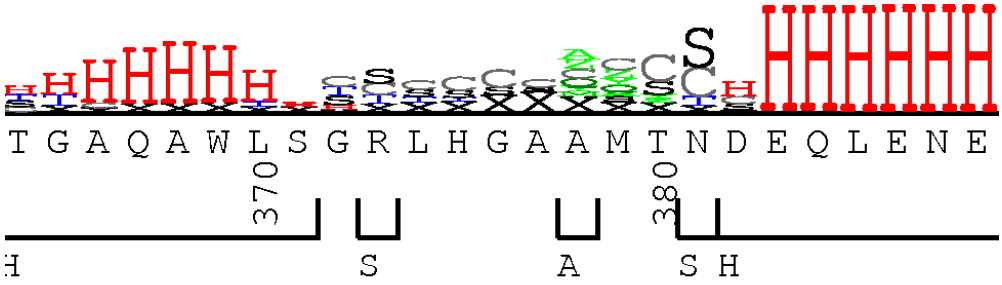
New Alphabets and The Two-stage Predictor

Following the success of CASP7, I investigated new inputs, including new local structure predictions based on new classifications of the hydrogen bonds in the structure developed by Grant Thiltgen. This chapter has four sections. The first section provides the results of including these new inputs, which I show provide an improvement to the CASP7 predictor. Professor Karplus suggested I evaluate a predictor based on local alphabets without the paired statistics. The second section examines the results of this evaluation which demonstrates that local predictions are powerful in their own respect. This leads to the design of a two-stage predictor discussed and evaluated in the third section. The final section shows the results of the improved CASP7 predictor and the new two-stage predictor at CASP8. I discuss the challenge that CASP8 provided.

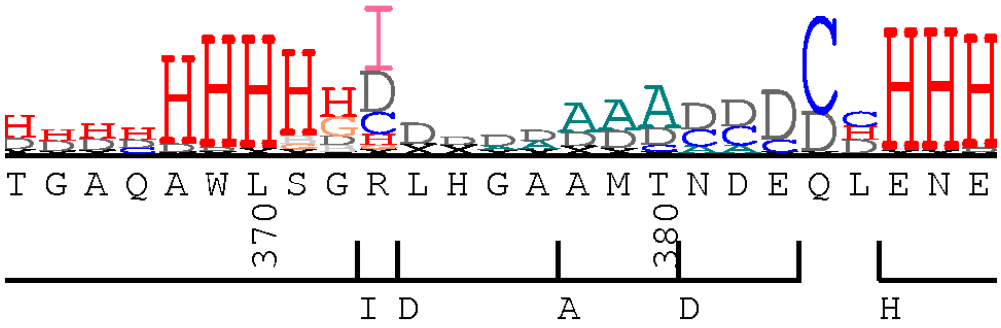
12.1 H-Bond Classification Alphabets and a new str Alphabet as Inputs

Development on new alphabets continued during and after CASP7. In this section I evaluate using two new alphabets as inputs and show the results. The first is from a set of alphabets using backbone h-bonds for classification developed by Grant Thiltgen [56]. He also developed a second alphabet, called STR4 [4], that draws upon STR2 and the h-bond alphabets.

I was interested in the h-bond alphabet because I felt it might complement the STR2 alphabet. Figure 12.1(a) shows a graphic presentation of STR2 predictions for a section of a structure [84]. The letters correspond to a letter of the alphabet for classification and their height represents the relative entropy of predicted distribution to the background distribution, i.e., a measure of the strength of the prediction. The section around residue number 375 of target T0318 in CASP represents a turn in the structure. Note that the heights of all the letters are diminished around the turn implying that the classifier is unsure of the actual class. Although the turn represents a sharp change in the structure, the predictions fail to represent this. I anticipated that the new h-bond alphabets would focus more strongly on changes in the structure at such turns, and would provide a contrast to STR2. In Figure 12.1(b), the region around residue 375 represents a better distinction of the structural changes as they appear at the turn. At residue 373 the letter I is prominent. This represents an h-bond from that residue's backbone nitrogen to the oxygen of the fifth residue further down the



(a) STR2 logo



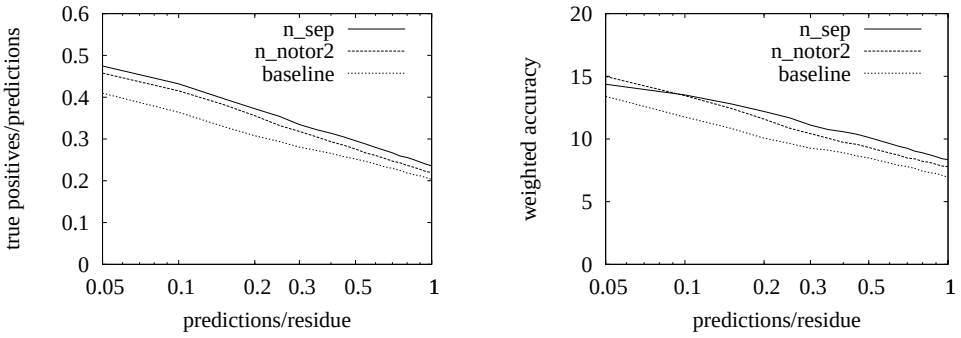
(b) N_SEP logo

Figure 12.1: Logo representations of STR2 and N_SEP alphabets. The higher the letter the stronger the likelihood. The N_SEP logo appears shifted to the right due to the definition of classes in the N_SEP alphabet. For example, the H class is for residues whose backbone nitrogen is the donor in a hydrogen bond. It is not until the fourth residue in the helix that we have such a bond to the first residue.

sequence.

This characteristic of identifying specific residues as part of a turn plus the high scores for predictability shown in Table 5.1 for the h-bond alphabets made them desirable for evaluation. From that Table 5.1, I chose the two best h-bond alphabets, N_SEP and N_NOTOR2, for evaluation.

I used a medium size network as the basis for testing the h-bond alphabets. It consists of log(sequence length), log(separation), amino acid distribution (window 5),



(a) Standard accuracy

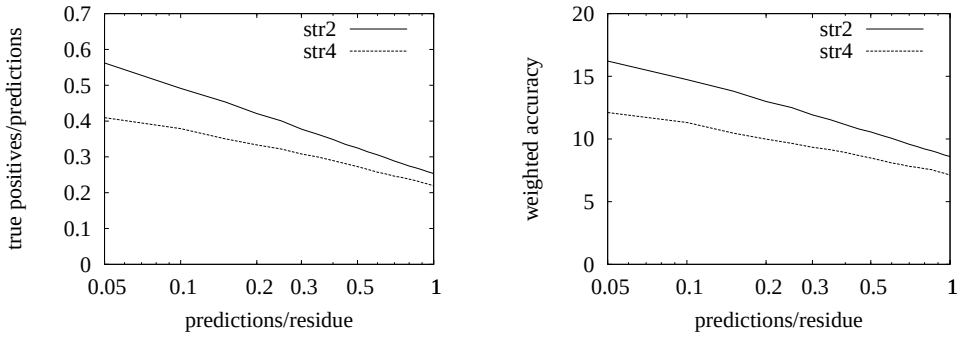
(b) Weighted accuracy

Figure 12.2: These plots show the results of three neural networks. The first is a baseline network consisting of log(sequence length), log(separation), AA alphabet, MI E-value, and propensity. The second adds the N_NOTOR2 alphabet to the baseline network, while the last adds N_SEP to the baseline. All alphabets use a window of 5. The keys are listed from best to worst using 0.1 predictions per residue.

and the two paired statistics, MI E-value, and propensity. The hidden layer has 11 nodes. To these inputs I added N_SEP for one network and N_NOTOR2 for the other, using a window of 5.

For the STR2, STR4 comparison, I used a neural network with the standard sequence length, and separation inputs, along with MI E-value and propensity. To give a good comparative test for the two alphabets, I used amino acid distribution, NEAR-BACKBONE-11, and either the STR2 or STR4 alphabets. All alphabets and the distribution used a window of 3. The size of the hidden layer was 11.

Training for all alphabet testing used the first 250 unbroken sequences and the next 50 unbroken sequences for cross-training. Training stopped after 200 epochs with no improvement.



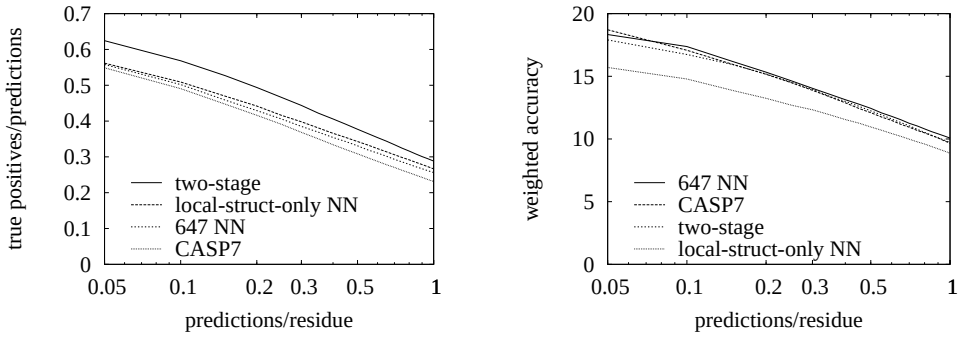
(a) Standard accuracy (b) Weighted accuracy

Figure 12.3: These plots show the results of two neural networks, one using the original STR2 alphabet and the other using the new STR4. The rest of the inputs are log(sequence length), log(separation), the alphabets AA and NEAR-BACKBONE-11, along with MI E-value and propensity. All alphabets used a window of 3.The keys are listed from best to worst using 0.1 predictions per residue.

12.1.1 Results of Including str4 or H-bond Alphabets

As shown in Figure 12.2 the N_SEP alphabet performs better than N_NOTOR2 for standard accuracy and, except for the region from 0.05 to 0.1, for weighted accuracy. Given these results, I selected N_SEP for inputs to include in the next predictor.

On the other hand, the results for STR4 show the original STR2 does much better (see Figure 12.3). The possible explanation is the choice of classes used in STR4; they attempt to bring in h-bond angles as defined in N_NOTOR to define different sections of beta strands rather than the classes of STR2 (see Section 5.3). In STR2, the strands use simple h-bonds (or lack of) to define the classes. I have already shown that N_SEP is better than N_NOTOR2 and the classes of N_SEP are simply defined by the separation between the nitrogen and oxygen atoms in the h-bond. STR4 does very well in predictability but this does not carry over to usability as inputs. I might be interested



(a) Standard accuracy

(b) Weighted accuracy

Figure 12.4: Plots comparing the new single-stage predictor, two-stage predictor, and the first stage of the two-stage predictor to the CASP7 predictor. The keys are listed from best to worst based on 0.1 predictions per residue.

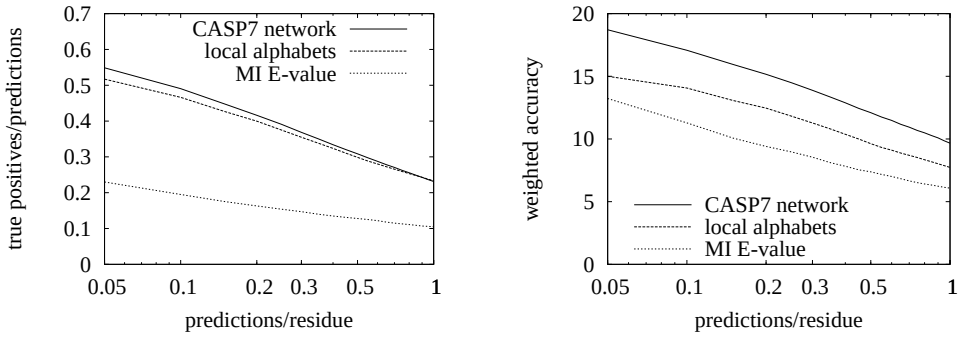
in testing a modified form of STR2 that includes some of the separation classes of N_SEP that relate to turns and loops rather than the catchall class of STR2 loops; that might be interesting.

The results of testing N_SEP led to the development of a new predictor which I will call the 647 predictor (because there are 647 inputs). This predictor starts with the inputs for the CASP7 predictor and adds N_SEP with a window of 5 (based on the window size of 5 currently used for AA and STR2 (Section 11.4)), and also adds Pearson's correlated coefficient as a paired statistic (Section 3.2). This addition is rather arbitrary and may be redundant; however, I felt that one more input could not be too much given that there were already 646 other inputs. The accuracy of this new predictor is shown in Figure 12.4.

12.1.2 Additional Training Examples

The CASP7 predictor used a set of 400 unbroken chains for its final training. While this represents a substantial number of overall examples, there is still room for more positive examples. With this in mind, I culled a new set of sequences from PISCES[96]. The PDB added many new structures from March 2003 until June 2007. I could tighten the criteria and still get more unbroken chains. The original set used resolution $\leq$ 1.8 Ångstroms, R-factor $\leq$ 0.25, and sequence similarity $\leq$ 30%. This new set limited the sequence similarity $\leq$ 20%. This gave a set of 1990 sequences. All sequences with length $<$ 50 were removed. The set was further reduced to unbroken chains. The older set of 421 unbroken chains was added to this set, and duplicates were removed. The final set has 804 sequences. This formed the new training set while a set of 100 broken chains formed the cross-training set.

However, I did not re-cull the set of ids after combining the old ids with the new ids. I have found that there was considerable sequence identity overlap in the new set of 804 sequences. If the set is culled to 30% identity using the PISCES server, only 691 ids qualify. Even relaxing to 40% identity results in only 725; a large number of the ids added to the new set are highly similar to the old set. The resulting set is a poor one and should not have been used for experimentation. More importantly, the inclusion of the new ids impacted the final set of 300 sequences that are used for evaluation. A test of identity overlap indicates that 11% of the 300 had higher than 30% identity with the training set and about 3% appear to be near or exactly identical. This can



(a) Standard accuracy

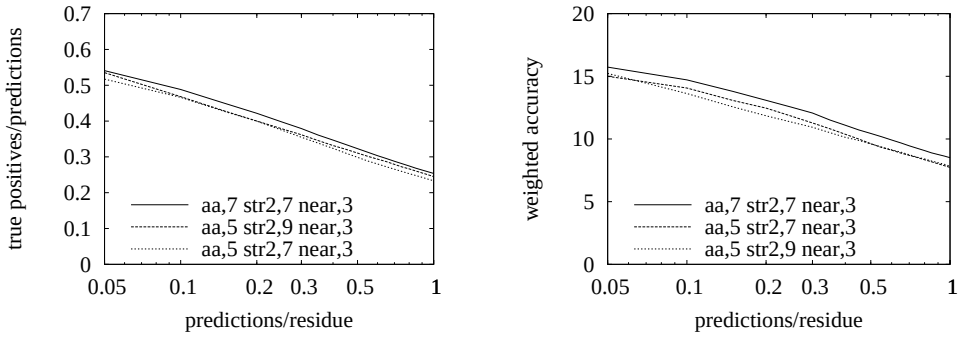
(b) Weighted accuracy

Figure 12.5: These plots compare a neural network predictor using log(sequence length), log(separation) and three local alphabets, AA, window of 5, STR2, window of 7, NEAR-BACKBONE-11, window of 3, to the CASP7 predictor and the best individual statistic, MI E-value. The window sizes for the local alphabets were determined later in testing. The keys are listed from best to worst based on 0.1 predictions per residue.

lead to better results in evaluation than when there is less than 30% identity between the training and evaluation sets. This cast doubt on the final comparisons between the CASP8 predictors and earlier predictors as the earlier predictors were trained on the clean sets.

12.2 The Power of Local Predictions

Early on I assumed that the most important inputs were the paired statistics, that the addition of the local structure predictions as inputs merely helped boost the final results, and that the local structure predictions did not provide an effective basis for a predictor by themselves. Encouraged by my advisor I investigated the effectiveness of such a predictor and find, on the contrary, that they do provide a significantly effective predictor.



(a) Standard accuracy (b) Weighted accuracy

Figure 12.6: Plots comparing local alphabet-based predictors with AA, STR2, and NEAR-BACKBONE-11. The difference is in the window sizes associated with each alphabet as shown in the key. These were the only combinations of window sizes tested. The keys are listed from best to worst based on 0.1 predictions per residue.

To test the power of a local alphabet-based predictor, I simply built a predictor with several local alphabet inputs and no paired inputs. The results, shown in Figure 12.5, are plotted against the best individual statistic and the CASP7 predictor. The local alphabet-based predictor does very well against the others. I had not expected this.

12.3 A Two-stage Predictor and Results

Because of the effectiveness of local structure alphabets as inputs, I decided to try a two-stage predictor where the output of a local alphabet-based predictor without paired statistics is used as input for a second neural network that does include the paired statistics. This approach has been done before [94], and I anticipated that the two stage could outperform a single stage predictor.

There are two advantages that could result from using two stages: the first

advantage is that the first stage output will have a significantly higher ratio of positives to negatives, thus negating the need to balance the input for the second stage (see Section 4.5). The second advantage is that the second stage now has a restricted set of pairs of residues, and this enables the second stage to improve even more than it could just using the unrestricted set of pairs.

I already had a good set of local alphabets to start with, but I was not sure what was the best set of window sizes. With this in mind, I tried various combinations as shown in Figure 12.6. The choice of a window size of 3 for NEAR-BACKBONE-11 was arbitrary; I felt that a smaller window than 5 for a burial alphabet was sufficient. The best combination I tried was a window of size 7 for AA, size 7 for STR2, size 3 for NEAR-BACKBONE-11, and size 5 for N_SEP; I used these sizes for the final local alphabet-based predictor. This predictor also included log(sequence length) and log(separation) resulting in 730 inputs and a hidden layer of 47. The accuracy results using the standard 300 evaluation sequences is shown in Figure 12.4.

The second stage is the single-stage predictor with 647 inputs discussed in Section 12.1 except for one extra input: the log of the rank of the pair with respect to the predictions of the first stage. Its accuracy is also in Figure 12.4. As shown, the two-stage outperforms the single-stage which in turn outperforms the CASP7 predictor.

The results show that the new single stage predictor does improve on the original CASP7 predictor but not as much as the two-stage predictor. Using a first stage based on local-structure predictions providing a concentrated base of predictions, the same predictor (with one extra input) as second stage can be trained to A possible

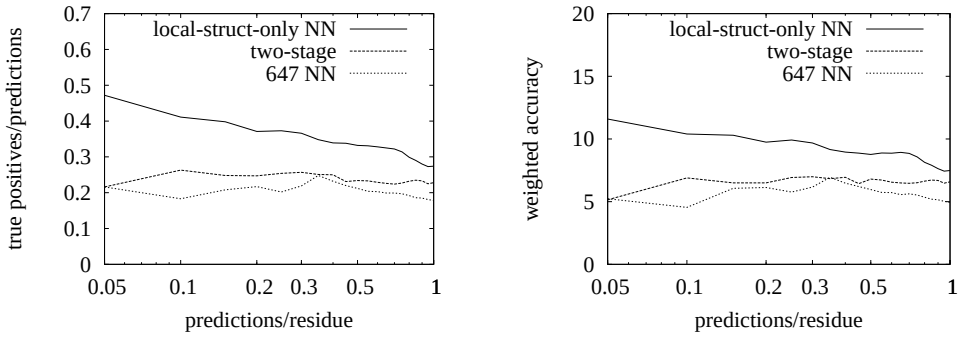
reason is that the removal of negative examples reduces the noise so more effective learning is possible.

The effectiveness of the extra input of the first-stage rank of each example is not known. This could be tested by using a second stage without the input and seeing how it compares to the results in Figure 12.4.

12.4 Reality Check: CASP8 Results

For CASP8 as for previous CASPs the contact prediction assessors limited themselves to predictions for free-modeling targets. However, in previous CASPs this simply meant that no one had submitted any template-based predictions for a target. For this CASP the assessors consider targets as free-modeling if there are no possible templates anywhere in the PDB. As a result, only 12 targets out of 120+ targets qualified as free-modeling. This made the assessment statistically less significant. But I also found that the targets typically had very few, if any, sequences in their multiple sequence alignments, and in CASP7 I had already established that having very few sequences in the alignment meant poor predictions; this still holds for the newest predictors.

Therefore, the results of CASP8 were disappointing. As shown in Figure 12.7, both the single-stage and the two-stage predictors did poorly especially when compared to the first stage predictor of the two-stage predictor. The first-stage predictor does the best by far in Figure 12.7, while the other predictors do much worse. These results are reversed from those in Figure 12.4. This is especially bad considering that the



(a) Standard accuracy

(b) Weighted accuracy

Figure 12.7: These accuracy graphs display the results for my predictions for the free-modeling targets in CASP8. Only the predictions for the single stage and two-stage predictors were submitted for evaluation.

second stage predictor is limited to the top $10 \times L$ predictions output by the first stage. It appears that the second stage fails to differentiate between the better and poorer predictions from the first stage, resulting in near random selections from the first stage predictor.

One possible improvement would be to train the neural network with examples where hard (high E-value) targets are up-sampled and easier ones are down-sampled. Also, it may be useful to include the best calibrated E-value of the MSA for the target as an input to the network. This may help the neural network adjust the weighting of the paired statistic inputs. The biggest improvement would likely come from having more sequences in the alignment. How to find and include such sequences remains to be determined.

12.5 Review

This chapter started with the introduction of new alphabets in Section 12.1. Three were selected for evaluation, N_SEP, N_NOTOR2, and STR4, of which N_SEP was chosen for a new predictor. I began to recognize the strength of local structure alphabets in Section 12.2. This revelation led to the development of a two-stage predictor, in addition to a single-stage predictor in Section 12.3 complete with improving results. But in Section 12.4 the difficulty of multiple sequence alignments with few sequences apparently prevented the strength of the new predictors from performing well at CASP8. In fact, the first stage of the two-stage predictor could generate better predictions for the CASP8 targets than either the single-stage or the two-stage predictors.

Chapter 13

Conclusions and Future Research

This chapter has two sections. The first discusses some topics not covered in previous chapters as well as an overview of what I have learned. The second covers possible future work and some preliminary results.

13.1 Conclusions

This research started as a pedagogical exercise, but has not ended that way. Initial predictions were sent to CASP6. Later predictions at CASP7 were judged the best predictions submitted. This led to a presentation at CASP7, and now to this thesis. While the results are encouraging, I have made some important observations; not all are positive.

Many Small Improvements Add Up. I observe that no single change provided a large improvement by itself. Instead, there was an accumulation of many small im-

provements: adjustments to selection of paired statistics, local alphabets, window sizes for local alphabets, thinning, rank instead of value, etc. Nevertheless, the final results for a single stage neural network for CASP8 is a considerable improvement from the initial network for CASP6.

Too Much Focus on Thinning. A lot of effort was spent on examining thinning as an improvement, but the most important result was simply to use a better thinning than 90%. There could have been more experiments examining the combination of multiple thinnings such as 50% and 70% with 62%, but the biggest problem with such combinations is the explosive growth of inputs. More time should have been spent on examining different combinations of local structure alphabets including the PB alphabet, based on protein blocks [23], and the ALPHA alphabet.

Too Much Focus on Paired Statistics. The early research attempted to find the best combination of paired statistics. In the end, I was adding paired statistics without too much concern. Paired statistics were usually sufficiently different and added so few inputs that convolution was not a problem. The most significant find was the use of rank instead of value; otherwise inclusion of paired statistics almost always helped.

How Useful is the Downsampling? The approach I used for downsampling the negative examples is similar to the technique of bagging or aggregate bootstrapping. I did no experiments to see if this actually improves on using a fixed smaller set of negative examples. If it does, then that is an interesting result by itself. It may be

that downsampling positive examples might improve the training, but nothing has been tried yet.

The "Attractive" Domain Problem. While doing my post-mortem on CASP8, I noticed a trend I had seen at CASP7. When a target consisted of two or more domains, and one of those domains was free-modeling and the other(s) were template-based, and the predictor was getting more accurate predictions, the predictor would tend to favor the template-based domains when making predictions.

This is not surprising. The template-based domains were likely to have more data per column, and the neural network would detect this from stronger local structure predictions. Then the network would make far more predictions in these domains of the sequence. As a result, the predictor would virtually ignore the harder "unattractive" free-modeling domain resulting in insufficient predictions to qualify for assessment.

Despite my awareness of this problem, I have not been working on any strategy to overcome it. Some method of analysis of where predictions fall within the sequence could be used to detect any ignored regions, and predictions adjusted to insure inclusion of those regions. It may be possible to actually use such detection to predict domains; however, much more research is needed. Unfortunately, this weakness is still a part of my prediction submission.

How Valuable is SAM? Recently I have considered what I should do for sequence alignments once I have graduated and might not have easy access to the SAM alignments. This also raises the question of how good are SAM alignments compared to other

alignment tools. Also, a new tool, CSBlast [7], is now available. Professor Karplus and I wonder if this database search and alignment method is superior to SAM. I have assumed that SAM is still one of the best, if not the best, tools for finding distant homologs, i.e. true structural homologs that have low sequence identity with the target. Since the contact predictions are expected to be used for free-modeling or hard template-based targets, alignments that can detect and include distant homologs are important.

To better judge the value of alignment tools, I would need to focus on hard targets, and do training and development using different alignment tools, before I can evaluate which alignment tool to use.

Better Comparison to Free-modeling Tertiary Models. I introduced a new evaluation, weighted accuracy, which is an improvement on the group of subsets currently used by CASP. This evaluation is only for comparing different contact predictions. There is still an outstanding question concerning the usefulness of contact predictions for model building.

It had appeared that research by John Archie established contact predictions as useful. John Archie and Kevin Karplus had done research in another category of CASP called *model quality assessment* or MQA. The goal is to assess a model and assign a value to the accuracy of the model with respect to the actual but unknown structure [4]. Archie analyzed a number of UNDERTAKER cost functions [53]. He initially found that the contact predictions were part of the set of cost functions most useful for MQA of free-modeling models, but a later test eliminated contact predictions as part

of the cost function.

When comparing my predictions to CASP7 tertiary models, I selected pairs from the tertiary models ordered by smallest distance between the residues in the pair. This selection method was arbitrary and is not necessarily the best comparison. Nor is having the tertiary model builders select their best predictions; if they have a partial template, they could use it to provide a number of excellent predictions that could swamp the accuracy of my contact predictions.

A better comparison would be to start with the top L/10 predictions from the contact predictor and take the corresponding pairs from the tertiary model and compare how well they do in predicting contacts (some pairs taken from the model may not be in contact. In that case, we treat them as predicting there is **no** contact). Then see which set provides the best predictions. Initially this may seem unfair to the tertiary model, but it is not. The real question that the comparison needs to answer is "do the predictions from the contact predictor add value to the work of the modeler"? The best comparison would be for the modeler to select its weakest pairs, and for the contact predictor to provide its best predictions that intersect with that set. By weakest pairs I am specifically thinking of those areas not covered by a template, or where different decoys have significantly different structures. If the contact predictor can show real improvement over the model at these sites, then its predictions should be useful in improving the model. Some might argue that this is giving too much advantage to the contact predictor, but in the context of the value-added question, it provides the best measure.

Unfortunately tertiary models are not submitted with data expressing a confidence value from the builder with respect to the parts of the structure (I am making a broad assumption that some form of confidence is even possible). Without confidence values, a simple comparison of the L/10 best contact predictions can still be performed.

Separation as Input. In Figure 10.2, I compared small networks with and without separation as an input. While including separation improved the standard accuracy, it significantly impacted the weighted accuracy. This apparently is due to the neural network learning that predictions with a smaller separation make better predictions. I stated that better standard accuracy was desirable for doing better at CASP. But the greater separation predictions, while riskier, are more likely to be useful. As I am feeling much more confident about my predictor, I am inclined to drop separation as an input and, in my estimation, provide what are better predictions. The best solution would be for CASP to use weighted accuracy in their assessment.

13.2 Future Research

The issues raised in the previous section helps to provide the foundation for future research.

Post-prediction Processing. Once the predictions are made, there is no effort to process them as a group to improve their accuracy. Two possible methods come to mind: filter the predictions using the predicted secondary structure, and ensure that

there are enough predictions to cover the length of the sequence.

For example, the first method could, when a residue is predicted to be on a helix, limit the number of possible predictions with respect to that residue. Weaker contact predictions involving that residue could be adjusted or removed. We already have a neural network that predicts the number of expected contacts per residue. This could be used to provide that limit directly without having to analyze predicted secondary structure. Or when a predicted contact matches two residues on different predicted beta strands, the neighboring residues along each strand could be examined to see if they also have reasonably good predictions of contact. If so, it could be possible to predict how the two strands may form a sheet.

The second method is aimed at the "attractive" domain problem. Ensuring that there are extra predictions, so no significantly long stretches of the sequence lack predictions, can prevent a domain from failing assessment at CASP because of insufficient predictions. Again, we could use the predicted number of contacts and require that at least a minimum number of predictions are made per residue.

Neural Network Changes. The neural network I use has not undergone any significant changes other than the size of the training set and the addition of new inputs. There are several areas I can explore in modifying the neural network.

Pollastri [71, 72] has been researching contact prediction by using recurrent/recursive neural networks. Because of the way in which he evaluates his results, it is difficult to determine whether or not these different types of networks have any merit.

I would want to implement these types of networks myself and do my own evaluation to determine merit.

It is possible to train several networks independently of each other and then combine them to predict contacts. Such an approach has been used successfully before for local structure predictions [44]. I only need to add code for a combination scheme to test the effectiveness of this approach.

The program, PREDICT-2ND that we use for predicting local structure predictions has a sophisticated network of inputs and hidden layers. Because it is specifically constructed around the single residue for which it is making predictions, it does not easily generalize to modeling inputs for the two residues used for contact predictions. Nevertheless, the excellent predictions made by PREDICT-2ND encourage a fresh look at trying to design an equivalent network structure for contact predictions.

The pb and alpha Alphabets. There are two local structure alphabets that I have not explored that may have merit: PB and ALPHA. The first scores well in terms of predictability but issues concerning the null model used in the multi-track version of SAM [54] has discouraged its inclusion. The second, ALPHA, is an alphabet that scores lower in predictability than most local structure alphabets. Still, it is different from those used as inputs and may complement them. It is not difficult to test ALPHA or PB with the current software. The main difficultly of adding new alphabets is the increase in the number of new inputs.

Modify Pearson's Correlation Coefficient. The Pearson's correlation coefficient contact statistic uses the same MacLachlan substitution matrix for every residue column (see Section 5.2). But it is possible to derive a substitution matrix for each column based on that column's regularized amino acid distribution (see the "Results" section of [7]). Using the derived substitution matrix in place of MacLachlan's matrix may improve the accuracy of the correlation coefficient statistic, and, hence, the performance of the final neural network.

Beta-sheet Prediction. Predicting the contacts in beta sheets is a problem. A method used for CASP6 [60] touches on this issue and others have looked at it as well [61, 18]. It would be worthwhile to train specifically for beta sheet contacts, especially to see if it is possible to predict parallel vs. anti-parallel bonding. This could help in building models.

A Random Forest as a Second Stage. There is a machine learning technique called *random forest* [13] which is based on decision trees and Robert Schapire's paper on the strength of weak learnability [82]. A proper decision tree uses entropy as the criterion to select one of n inputs for each decision node while building the tree. A random tree selects from only k of n inputs, where k is a fixed number $< n$. Each tree is then trained on a random subset of the actual examples. The collection of trees uses weighted voting to predict the classification of new samples. Schapire's result shows that if you have weak but complementary trained classifiers, you can establish weighted voting so that the collection is a better classifier than its individual units. The randomness causes

the trees to be complementary in their decisions, thus satisfying the main criterion of Schapire's paper.

Ordinarily a random forest is used where there may be limited and/or missing data, but, because they deal well with many inputs and can help identify irrelevant inputs, they may be useful as a second stage. Clearly the method is different from neural networks and that may imply that what they learn may complement what is learned by a neural network. Either in tandem or as two stages, the resulting predictions could be better than those from neural networks alone.

Transmembrane Protein Prediction. Transmembrane proteins are inherently difficult to crystallize if they can be crystallized at all. While there are tens of thousands of protein structures in the PDB, only a few hundred of them are transmembrane protein structures.

Most transmembrane proteins can be roughly divided into structural regions given their sequences. It may be possible to predict contacts for each region, and predict their structures well enough to aid experimentalists in their research. This would require finding and teaming up with such experimentalists. The possibility of aiding experimentalists directly would be a good demonstration of the effectiveness of bioinformaticists helping wet lab research.

Bibliography

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*. Garland Publishing, Inc., 2002.

[2] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[3] S. Altschul, T. Madden, A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, September 1997.

[4] John Archie and Kevin Karplus. Applying undertaker cost functions to model quality assessment. *Proteins: Structure, Function, and Bioinformatics*, 75(3):550–555, 2009. published online 30 Sep 2008, doi:10.1002/prot.22288.

[5] P. Baldi and G. Pollastri. The principled design of large-scale recursive neural network architectures DAG-RNNs and the protein structure prediction problem. *Journal of Machine Learning Research 4*, pages 575–601, 2003.

[6] F.C. Bernstein, T. F. Koetzle, G. J. Williams, E. E. Meyer, M. D. Brice, J. R.

Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The Protein Data Bank: a computer-based archival file for macromolecular structures. *Journal of Molecular Biology*, 112:535–542, November 1977.

[7] A. Biegert and J. Soding. Sequence context-specific profiles for homology searching. *Proceedings of the National Academy of Sciences, USA*, 106(10):3770–3775, 2009.

[8] Tom L. Blundell, Harren Jhoti, and Chris Abell. High-throughput crystallography for lead discovery in drug design. *Nature Reviews Drug Discovery*, 1:45–54, 2002.

[9] Richard Bonneau, Jerry Tsai, Ingo Ruczinski, Dylan Chivian, Carol Rohl, Charlie E. M. Strauss, and David Baker. Rosetta in CASP4: progress in ab initio protein-structure prediction. *Proteins: Structure, Function, and Genetics*, 45(S5):119–126, 2001.

[10] P. Bradley, D. Chivian, J. Meiler, K.M.S. Misura, C.A. Rohl, W.R. Schief, W.J. Wedemeyer, O. Schueler-Furman, P. Murphy, and J. Schonbrun. Rosetta predictions in CASP 5: Successes, failures, and prospects for complete automation. *Proteins: Structure, Function, and Genetics*, 53(s 6):457–468, 2003.

[11] Philip Bradley, Lars Malmström, Bin Qian, Jack Schonbrun, Dylan Chivian, David E. Kim, Jens Meiler, Kira M.S. Misura, and David Baker. Free modeling with Rosetta in CASP6. *Proteins: Structure, Function, and Bioinformatics*, 61(S7):128–134, September 2005.

[12] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[13] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[14] J. S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing, Algorithms, Architectures and Applications. Proceedings of the NATO Advanced Research Workshop*, pages 227–36. Springer-Verlag, 1990.

[15] C. Bystroff and D. Baker. Blind predictions of local protein structure in casp2 targets using the i-sites library. *Proteins: Structure, Function, and Genetics*, Suppl 1:167–171, 1997.

[16] C. Bystroff, V. Thorsson, and D. Baker. HMMSTR: a hidden Markov model for local sequence-structure correlations in proteins. *Journal of Molecular Biology*, 301(1):173–190, August 2000.

[17] Prediction Center. CASP8 in numbers. http://predictioncenter.org/casp8/numbers.cgi, 2007.

[18] Jianlin Cheng and Pierre Baldi. Three-stage prediction of protein β-sheets by neural networks, alignments and graph algorithms. *Bioinformatics*, 21(Supplement 1):175–184, 2005.

[19] P. Y. Chou and G. D. Fasman. Prediction of the secondary structure of proteins from their amino acid sequence. *Adv Enzymol Relat Areas Mol Biol*, 47:45–45, 1978.

[20] Melissa S. Cline, Kevin Karplus, Richard H. Lathrop, Temple F. Smith, Robert G.

Rogers Jr., and David Haussler. Information-theoretic dissection of pairwise contact potentials. *Proteins: Structure, Function, and Genetics*, 49(1):7–14, 1 October 2002.

[21] Thomas E. Creighton. *Proteins: Structures and Molecular Properties*. W. H. Freeman, second edition edition, 1992. referenced for fact that helices are 38 percent of residues.

[22] G.V. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[23] A.G. de Brevern, C. Etchebest, and S. Hazout. Bayesian probabilistic approach for predicting backbone structures in terms of protein blocks. *Proteins: Structure, Function, and Genetics*, 41:271–287, November 2000.

[24] Eran Eyal, Milana Frenkel-Morgenstern, Vladimir Sobolev, and Shmuel Pietrokovski. A pair-to-pair amino acids substitution matrix and its applications for protein structure prediction. *Proteins: Structure, Function, and Bioinformatics*, 67:142–153, 2007.

[25] S. Fahlman. An empirical study of learning speed in backpropagation networks. Technical report, Carnegie-Mellon University, Computer Science Dept., 1988.

[26] P. Fariselli and R. Casadio. A neural network based predictor of residue contacts in proteins. *Protein Engineering*, 12(1):15–21, 1999.

[27] P. Fariselli, O. Olmea, A. Valencia, and R. Casadio. Prediction of contact maps

with neural networks and correlated mutations. *Protein Engineering*, 14(11):835–43, 2001.

[28] P. Fariselli, O. Olmea, A. Valencia, and R. Casadio. Progress in predicting inter-residue contacts of proteins with neural networks and correlated mutations. *Proteins: Structure, Function, and Genetics*, Suppl 5(1):157–162, 2001.

[29] A. Fodor and R. Aldrich. Influence of conservation on calculations of amino acid covariance in multiple sequence alignments. *Proteins: Structure, Function, and Bioinformatics*, 56:211–221, 2004.

[30] Dimitrij Frishman and Patrick Argos. Knowledge-based protein secondary structure assignment. *Proteins: Structure, Function, and Genetics*, 23:566–579, December 1995.

[31] Ulrike Göbel, Chris Sander, Reinhard Schneider, and Alfonso Valencia. Correlated mutations and residue contacts in proteins. *Proteins: Structure, Function, and Genetics*, 18:309–317, 1994.

[32] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading, 1989.

[33] Osvaldo Grana, David Baker, Robert M. MacCallum, Jens Meiler, Marco Punta, Burkhard Rost, Michael L. Tress, and Alfonso Valencia. CASP6 assessment of contact prediction. *Proteins: Structure, Function, and Bioinformatics*, Suppl 7:214–224, 2005.

[34] Nicholas Hamilton, Kevin Burrage, Mark A. Ragan, and Thomas Huber. Protein contact prediction using patterns of correlation. *Proteins: Structure, Function, and Bioinformatics*, 56:679–684, 2004.

[35] J.M. Hannan and J.M. Bishop. *Artifical Neural Networks*, volume Conference Publication No. 440, chapter A comparison of fast training algorithms over two real problems. 1997.

[36] Steven Henikoff and Jorja G. Henikoff. Automated assembly of protein blocks for database searching. *Nucleic Acids Research*, 19(23):6565–6572, 1991.

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

[38] Steven Henikoff and Jorja G. Henikoff. Position-based sequence weights. *Journal of Molecular Biology*, 243(4):574–578, November 1994.

[39] Lee H. K. Herbert. Bayesian nonparametrics via neural networks. In *ASA-SIAM Series on Statistics and Applied Probability*. Society for Industrial and Applied Mathematics, 2004.

[40] T. Hubbard, A. Murzin, S. Brenner, and C. Chothia. SCOP: a structural classification of proteins database. *Nucleic Acids Research*, 25(1):236–9, January 1997.

[41] Richard Hughey, Kevin Karplus, and Anders Krogh. SAM: Sequence alignment and modeling software system, version 3. Technical Report UCSC-CRL-99-11, Univer-

sity of California, Santa Cruz, Computer Engineering, UC Santa Cruz, CA 95064, October 1999. Available from `http://www.soe.ucsc.edu/research/compbio/sam.html`.

[42] Richard Hughey and Anders Krogh. Hidden Markov models for sequence analysis: Extension and analysis of the basic method. *Computer Applications in the Biosciences*, 12(2):95–107, 1996. Information on obtaining SAM is available at `http://www.soe.ucsc.edu/research/compbio/sam.html`.

[43] C. Igel and M. Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 50(C):105–123, 2003.

[44] D.T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292:195–202, September 1999.

[45] Wolfgang Kabsch and Chris Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, December 1983.

[46] Rachel Karchin, Melissa Cline, and Kevin Karplus. Evaluation of local structure alphabets based on residue burial. *Proteins: Structure, Function, and Genetics*, 55(3):508–518, 5 2004. doi:10.1002/prot.20008.

[47] Rachel Karchin, Melissa Cline, Yael Mandel-Gutfreund, and Kevin Karplus. Hidden Markov models that use predicted local structure for fold recognition: al-

phabets of backbone geometry. *Proteins: Structure, Function, and Genetics*, 51(4):504–514, June 2003.

[48] Kevin Karplus. Regularizers for estimating distributions of amino acids from small samples. In *Proceedings, 3rd International Conference on Intelligent Systems for Molecular Biology*, Menlo Park, CA, July 1995. AAAI/MIT Press.

[49] Kevin Karplus. Predicting protein structure using SAM, UCSC's hidden Markov model tools. In Igor F. Tsigelny, editor, *Protein Structure Prediction: Bioinformatic Approach*, IUL Biotechnology Series, pages 297–323. International University Line, La Jolla, California, 2002.

[50] Kevin Karplus. UCSC Dirichlet mixture WWW page, 2002. `http://www.soe.ucsc.edu/research/compbio/dirichlets/index.html`.

[51] Kevin Karplus, Christian Barrett, and Richard Hughey. Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856, November 1998.

[52] Kevin Karplus, Rachel Karchin, Christian Barrett, Spencer Tu, Melissa Cline, Mark Diekhans, Leslie Grate, Jonathan Casper, and Richard Hughey. What is the value added by human interventionin protein structure prediction? *Proteins: Structure, Function, and Genetics*, 45(S5):86–91, 2001.

[53] Kevin Karplus, Rachel Karchin, Jenny Draper, Jonathan Casper, Yael Mandel-Gutfreund, Mark Diekhans, and Richard Hughey. Combining local-structure,

fold-recognition, and new-fold methods for protein structure prediction. *Proteins: Structure, Function, and Genetics*, 53(S6):491–496, 15 2003.

[54] Kevin Karplus, Rachel Karchin, George Shackelford, and Richard Hughey. Calibrating E-values for hidden Markov models with reverse-sequence null models. *Bioinformatics*, 21(22):4107–4115, 2005. doi:10.1093/bioinformatics/bti629.

[55] Kevin Karplus, Sol Katzman, George Shackelford, Martina Koeva, Jenny Draper, Bret Barnes, Marcia Soriano, and Richard Hughey. SAM-T04: what's new in protein-structure prediction for CASP6. *Proteins: Structure, Function, and Bioinformatics*, 61(S7):135–142, September 2005.

[56] Sol Katzman, Christian Barrett, Grant Thiltgen, Rachel Karchin, and Kevin Karplus. Predict-2nd: a tool for generalized protein local structure prediction. *Bioinformatics*, 24(21):2453–2459, 1 2008.

[57] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, February 1994.

[58] C. Levinthal. *Mossbauer Spectroscopy in Biological Systems*, pages 22–24. University of Illinois, 1969.

[59] S. W. Lockless and R. Ranganathan. Evolutionarily conserved pathways of energetic connectivity in protein families. *Science*, 286:295–299, 1999.

[60] R.M. MacCallum. Striped sheets and protein contact prediction. *Bioinformatics*, 20 Suppl. 1:224–231, 2004.

[61] Yael Mandel-Gutfreund, S.M. Zaremba, and L. Gregoret. Contributions of residue pairing to β-sheet formation: Conservation and covariation of amino acid residue pairs on antiparallel β-strands. *Journal of Molecular Biology*, 305:1145–1159, 2001.

[62] L. C. Martin, G. B. Gloor, S. D. Dunn, and L. M. Wahl. Using information theory to search for co-evolving residues in proteins. *Bioinformatics*, 21(22):4116–4124, 2005.

[63] Andrew D. McLachlan. Tests for comparing related amino acide sequences. *Journal of Molecular Biology*, 61:409–424, 1971.

[64] J. Moult, T. Hubbard, S. Bryant, K. Fidelis, and J. Pedersen. Critical assessment of methods of protein structure prediction (CASP): round II. *Proteins: Structure, Function, and Genetics*, Supplement 1(1):2–6, 1997.

[65] J. Moult, T. Hubbard, K. Fidelis, and J. Pedersen. Critical assessment of methods of protein structure prediction (CASP): round III. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):2–6, 1999.

[66] John Moult, Krzysztof Fidelis, Burkhard Rost, Tim Hubbard, and Anna Tramontano. Critical assessment of methods of protein structure prediction (CASP)-Round 6. *Proteins: Structure, Function, and Bioinformatics*, 61(S7):3–7, 26 2005.

[67] John Moult, Krzysztof Fidelis, Adam Zemla, and Tim Hubbard. Critical assessment

of methods of protein structure prediction (CASP)-Round V. *Proteins: Structure, Function, and Genetics*, 53(S6):334–339, 2003.

[68] Steffen Nissen and Evan Nemerson. Fast artificial neural network. `http://fann.sourceforge.net/`, October 2004.

[69] O. Olmea and A. Valencia. Improving contact predictions by the combination of correlated mutations and other sources of sequence information. *Fold Design*, 2(3):S25–32, 1997.

[70] L. Pauling, R. Corey, and H.R. Branson. The structure of proteins: two hydrogen-bonded helical conformations of the polypeptide chain. *Proceedings of the National Academy of Sciences, USA*, 37(4):205–211, April 1951.

[71] G. Pollastri and P. Baldi. Predition of contact maps by giohmms and recurrent neural networks using lateral propagation from all four cardinal corners. In *Proceedings of the ISMB 2002 Conference*, 2002.

[72] G. Pollastri, P. Baldi, A. Vullo, and P. Frasconi. Prediction of protein topologies using generalized IOHMMs and RNNs. In S. Thrun, S. Becker, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1449–1456. MIT Press, Cambridge, MA, 2003.

[73] M. Punta and B. Rost. PROFcon: novel prediction of long-range contacts. *Bioinformatics*, 21(13):2960–2968, 2005.

[74] G.N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan. Stereochemistry of polypeptide chain configurations. *Journal of Molecular Biology*, 7:95–99, July 1963.

[75] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 586–591, San Francisco, CA, 1993. IEEE.

[76] B. Rost. Phd: predicting one-dimensional protein structure by profile-based neural networks. *Methods in Enzymology*, 266:525–39, 1996.

[77] B. Rost. Review: Protein secondary structure prediction continues to rise. *Journal of Structural Biology*, 134:204–218, 2001.

[78] Burkhard Rost and Chris Sander. Structure prediction of proteins—where are we now? *Current Opinion in Biotechnology*, 5:372–380, 1994.

[79] Ingo Ruczinski, Charles Kooperberg, Richard Bonneau, and David Baker. Distributions of beta sheets in proteins with application to structure prediction. *Proteins: Structure, Function, and Bioinformatics*, 48(1):85–97, 2002.

[80] A. Sali and T. L. Blundell. Comparative protein modelling by satisfaction of spatial restraints. *Journal of Molecular Biology*, 234(3):779–815, December 1993.

[81] C. Sander and R. Schneider. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins: Structure, Function, and Genetics*, 9(1):56–68, 1991.

[82] R. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, June 1990.

[83] M Schervish. *Theory of Statistics*, page 459. Springer-Verlag, New York, 1995.

[84] T.D. Schneider and R.M Stephens. Sequence logos: a new way to display consensus sequences. *Nucleic Acids Research*, 18(10):6097–100, 1990.

[85] George Shackelford and Kevin Karplus. Contact prediction using mutual information and neural nets. *Proteins: Structure, Function, and Bioinformatics*, 69(S8):159–164, 2007. doi:10.1002/prot.21791.

[86] Yu Shao and Chris Bystroff. Predicting protein inter-residue contacts using templates and pathways. Found on web, 2003.

[87] Kim T. Simons, Rich Bonneau, Ingo Ruczinski, and David Baker. Ab initio protein structure prediction of CASP III targets using ROSETTA. *Proteins: Structure, Function, and Genetics*, Supplement 3(1):171–176, 1999.

[88] Manfred J. Sippl, Peter Lackner, Francisco S. Domingues, Andreas Prlić, Rainer Maik, Antonina Andreeva, and Markus Wiederstein. Assessment of the CASP4 fold recognition category. *Proteins: Structure, Function, and Genetics*, 45(S5):55–67, 2001.

[89] K. Sjölander, K. Karplus, M. P. Brown, R. Hughey, A. Krogh, I. S. Mian, and D. Haussler. Dirichlet mixtures: A method for improving detection of weak but

significant protein sequence homology. *Computer Applications in the Biosciences*, 12(4):327–345, August 1996.

[90] Y. Solano and H. Ikeda. A comparative study of eight learning algorithms for artificial neural networks based on a real application. *IEEE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E81A:355–357, FEB 1998.

[91] William R. Taylor and Kerr Hatrick. Compensating changes in protein multiple sequence alignments. *Protein Engineering*, 7(3):341–348, 1994.

[92] G. E. Tusndy, Z. Dosztnyi, and I. Simon. Transmembrane proteins in the Protein Data Bank: identification and classification. *Bioinformatics*, 20(17):2964–2972, 2004.

[93] Peter van Rossum. Lightweight neural network. `http://lwneuralnet.sourceforge.net/`, 2003.

[94] Alessandro Vullo, Ian Walsh, and Gianluca Pollastri. A two-stage approach for improved prediction of residue contact maps. *BMC Bioinformatics*, 7(180), March 2006. http://www.biomedcentral.com/1471-2105/7/180.

[95] Björn Wallner and Arne Elofsson. All are not equal: a benchmark of different homology modeling programs. *Protein Sci*, 14:1315–1327, May 2005.

[96] G. Wang and R. L. Dunbrack, Jr. PISCES: a protein sequence culling server. *Bioinformatics*, 19:1589–1591, 2003.

[97] Z. Xiang and B. Honig. Extending the accuracy limits of prediction for side-chain conformations. *Journal of Molecular Biology*, 311:421–430, 2001.

[98] Mohammed J. Zaki, Shan Jin, and Chris Bystroff. Mining residue contacts in proteins using local structure predictons. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 33(5):789, October 2003.