

Coordination Number Prediction Using Learning Classifier Systems: Performance and interpretability

Jaume Bacardit^{*}
ASAP research group, School
of Computer Science and IT,
University of Nottingham,
Jubilee Campus, Nottingham,
NG8 1BB, UK
jqb@cs.nott.ac.uk

Michael Stout
ASAP research group, School
of Computer Science and IT,
University of Nottingham,
Jubilee Campus, Nottingham,
NG8 1BB, UK
mqs@cs.nott.ac.uk

Natalio Krasnogor[†]
ASAP research group, School
of Computer Science and IT,
University of Nottingham,
Jubilee Campus, Nottingham,
NG8 1BB, UK
nrx@cs.nott.ac.uk

Jonathan D. Hirst
School of Chemistry,
University of Nottingham,
University Park, Nottingham
NG7 2RD, UK
jonathan.hirst
@nottingham.ac.uk

Jacek Blazewicz
Poznan University of
Technology, Institute of
Computing Science, ul.
Piotrowo 3a, 60-965 Poznan,
Poland.
jblazewicz
@cs.put.poznan.pl

ABSTRACT

The prediction of the coordination number (CN) of an amino acid in a protein structure has recently received renewed attention. In a recent paper, Kinjo et al. proposed a re-valued definition of CN and a criterion to map it onto a finite set of classes, in order to predict it using classification approaches. The literature reports several kinds of input information used for CN prediction. The aim of this paper is to assess the performance of a state-of-the-art learning method, Learning Classifier Systems (LCS) on this CN definition, with various degrees of precision, based on several combinations of input attributes. Moreover, we will compare the LCS performance to other well-known learning techniques. Our experiments are also intended to determine the minimum set of input information needed to achieve good predictive performance, so as to generate competent yet simple and interpretable classification rules. Thus, the generated predictors (rule sets) are analyzed for their interpretability.

*Corresponding author

†Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

GECCO '06, July 8–12, 2006, Seattle, Washington, USA.
Copyright 2006 ACM 1-59593-186-4/06/0007 ...\$5.00.

Categories and Subject Descriptors

I.2.6 [Artificial Intelligence]: Learning—*Concept Learning, Induction*; J.3 [Computer Applications]: Life and Medical Sciences—*Biology and Genetics*

General Terms

Algorithms, Experimentation, Performance

Keywords

Evolutionary Algorithms, Learning Classifier Systems, Rule Induction, Bioinformatics, Protein Structure Prediction, Coordination Number Prediction

1 Introduction

One of the main open problems in computational biology is the prediction of the 3D structure of protein chains. One approach to this problem is to predict some attributes of a protein, such as the secondary structure, the solvent accessibility or the coordination number, and then integrate this knowledge into a full 3D structure predictor.

The coordination number (CN) problem is defined as the prediction, for a given residue, of the number of residues from the same protein that are in contact with it in the native state. Two residues are said to be in contact when the distance between them is below a certain threshold. This problem is closely related to contact map (CM) prediction that predicts, for all possible pairs of residues of a protein, if they are in contact or not creating a bidimensional binary map. The coordination number of a residue is the count of the number of ones in the row of the map associated to the residue. Contact maps for any protein dataset could be easily generated through our protein structure comparison web server at <http://www.procksi.net/>.

Recently, there has been renewed interest in CN and CM prediction [5, 13, 16, 22, 26], due partially to the availabil-

ity of larger datasets and to advances in software and hardware. Most studies use the definition of contact informally stated above, which is essentially a categorical representation. However, Kinjo et al. [13] have proposed a real-valued definition of contact, and used linear regression to predict the CN based on the amino acid (AA) type of the protein primary sequence and global information about the protein. In order to compare the performance of their method with the classification approaches to CN prediction, they proposed a method to define class boundaries.

The goal of this paper is to assess the predictive performance of LCS within the problem definition proposed by Kinjo et al., treating directly the CN prediction as a classification problem. The chosen machine learning algorithm belongs to the family of Learning Classifier Systems (LCS) [24, 9], which are rule-based machine learning systems using evolutionary computation [10] as the search mechanism. Specifically, we have used a recent system called GAssist, which generates accurate, compact and interpretable solutions [2].

We will test the chosen CN definition using six different sets of input attributes, representing an increasingly rich information space in which learning takes place. The first of these sets is the simplest: the amino acid (AA) type of a window of residues around the residue for which coordination number is being predicted. We call this residue the target residue. The next sets will incrementally add extra information, such as global protein information or predicted properties of the protein, as it is usually assumed that a richer dataset will yield better predictions. We carefully assess the value of this assumption. The performance of GAssist on these datasets will be compared against several alternative learning mechanisms, and the performance of all these machine learning paradigms will be discussed. Finally, we will analyze the solutions generated by GAssist.

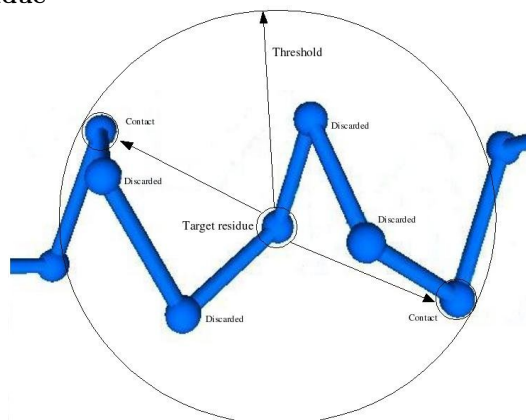
The rest of the paper is structured as follows: Section 2 will contain a brief summary of background information and several recent CM/CN prediction approaches. Section 3 will describe the particular CN definition, class partitions, performance measures and datasets used in this paper. Section 4 will describe the main characteristics of GAssist. Section 5 will report the results of the experiments. In section 6 we will discuss our results and, finally, section 7 will describe the conclusions and further work.

2 Background and related work

Proteins are heteropolymer molecules constructed as a chain of residues or amino acids of 20 different type. This string of amino acids is known as primary sequence. In native state, the chain folds to create a 3D structure. It is thought that this folding process has several steps. The first step, called secondary structure, consists of local structures such as alpha helix or beta sheets. These local structures can group in several conformations or domains forming a tertiary structure. The final 3D structure of a protein consists of one or more domains. In this context, the coordination number of a certain residue is a profile of this folding process indicating the number of other residues that, after the folding process, end up being near the target residue. Some of these contacts can be close in the protein chain but some other can be quite far apart, some trivial contacts such as those with the immediate neighbour residues are discarded. Figure 1 contains a graphical representation of the CN of a residue

for an alpha helix, given a minimum chain separation (discarded trivial contacts) of two. In this example, the CN is two.

Figure 1: Graphical representation of the CN of a residue



There is a large literature in CN and CM prediction, in which a variety of machine learning paradigms have been used, such as linear regression [13], neural networks [5], hidden markov models [22], a combination of self-organizing maps and genetic programming [16] or support vector machines [26].

There are two usual definitions of the distance used to determine whether or not there is contact between two residues. Some methods use the Euclidean distance between the C_α atoms of the two residues [22], while other methods use the C_β atom (C_α for glycine) [13]. Also, several methods discard the contacts between neighbouring residues in the primary chain by counting only contacts with a chain separation greater than a certain minimum. There are also many different distance thresholds.

Several kinds of input information are used in CN prediction beside the AA type of the residues in the primary chain, such as global information of the protein chain [13], data extracted from multiple sequences alignments [22, 5, 26, 16, 13] (mainly from PSI-BLAST [1]), predicted secondary structure [5, 26], predicted solvent accessibility [5], physical characteristics of the residues [23] or sequence conservation [26].

Finally, there are two approaches for the methods that deal with this problem using classification. On the one hand, the absolute CN could be predicted, assigning a class to each possible value of CN. On the other hand, some studies propose criteria to group instances with close CN, such as separating the instances that have lower or higher CN than the average of the training set [5], or defining classes in a way that guarantees uniform class distribution [13].

3 Problem definition

3.1 Definition of coordination number

We have used the definition of Kinjo et al. [13] of CN as it is the most recent work for this domain, and because its non-crisp definition can give a smoother fitness landscape. The distance used is defined using the C_β atom (C_α for glycine) of the residues. Next, the boundary of the sphere around the residue defined by the distance cutoff $d_c \in \mathbb{R}^+$ is made smooth by using a sigmoid function. A minimum

chain separation of two residues is required. Formally, the CN, N_i^p , of residue i in protein chain p is computed as:

$$N_i^p = \sum_{j:|j-i|>2} \frac{1}{1 + \exp(w(r_{ij} - d_c))} \quad (1)$$

where r_{ij} is the distance between the C_β atoms of the i th and j th residues. The constant w determines the sharpness of the boundary of the sphere.

3.2 Conversion of the real-valued CN definition into a classification domain

In order to predict real-valued CN using classification techniques, we need to map the continuous domain onto a finite set of categories. In this paper we will test two different criteria to generate datasets with n classes: The first one (**uniform frequency - UF**) was used by Kinjo et al. Partition the CN domain into a set of classes that contain equal number of examples. The second criterion (**uniform length - UL**) partitions the CN domain into a set of classes that cover segments of equal length of the domain. These two criteria correspond to the two more widely known unsupervised discretization algorithms of the same names [14].

3.3 Protein dataset

We have used the dataset and training/test partitions proposed by Kinjo et al. The protein chains were selected from PDB-REPRDB [18] with the following conditions: less than 30% of sequence identity, sequence length greater than 50, no membrane proteins, no nonstandard residues, no chain breaks, resolution better than 2 Å and a crystallographic R factor better than 20%. Chains that had no entry in the HSSP [21] database were discarded. The final data set contains 1050 protein chains and 257560 residues.

3.4 Definition of the training and tests sets

This definition includes two parts: the generation of the training and test partitions and the proposal of the sets of input information from which the CN is predicted.

The set was divided randomly into ten pairs of training and test set using 950 proteins for training and 100 for test in each set, using bootstrap. The proteins included in each partition are reported in <http://maccl01.genes.nig.ac.jp/~akinjo/sippre/suppl/list/>. We have placed a copy of the dataset at <http://www.asap.cs.nott.ac.uk/~jqb/datasetsGECCO-2006.tar.gz> (127MB).

We enriched the CN definition with six different set of input attributes. The first set represents the simplest set of input data: the AA type of the residues in a window around the target one. The following sets add extra information, such as global protein information or predicted characteristics of the protein. The set of input attributes are labeled **CN1** through **CN6** in the rest of this paper. This allows us to assess rigorously whether additional information is of benefit, and the degree of usefulness of each kind of extra data.

The global protein information consists of 21 real-valued attributes. The first attribute is the length of the protein chain (number of residues). The other 20 attributes contain the frequency of each AA type in the protein chain. Two types of predicted information have been used. The first is the average real-valued CN of a protein chain [13], called **PredAveCN**. This feature was predicted using GAssist itself. PredAveCN was partitioned into 10 classes (10 different

states in the PredAveCN domain), using the two criteria defined in subsection 3.2. This protein-wise feature was predicted from the 21 global protein attributes stated above, that is, the protein length and the frequency of appearance of the 20 AA types in the chain. The second predicted information is secondary structure of the target residue, using the PSI-PRED predictor [12]. This predicted information consists of two parts: a secondary structure type (helix, strand or coil) and a confidence level ([0..9]) of the prediction.

Table 1 summarizes the input attributes used in the datasets, and table 2 describes which attributes are included in each sets of input information. CN3 and CN5 represent two different ways of aggregating the same source of information to CN1, either as global information or as a predicted information. CN2, CN4 and CN6 add the predicted secondary structure to CN1, CN3 and CN5, respectively. Finally, we will partition the real-valued CN definition into two, three and five states (classes), using the two criteria described in section 3.2. We will evaluate a total of 36 datasets (six sets of input attributes, three of classes and two class definitions) derived from the Kinjo et al.'s protein dataset.

3.5 Performance measure

The accuracy that will be reported in section 5 is not the standard machine learning accuracy metric ($\#$ correct examples/ $\#$ total examples). As is usual in the protein structure prediction field [13, 12], we will take into account the fact that each example (a residue) belongs to a protein chain. Therefore, we will first compute the standard accuracy measure for each protein chain, and then average these accuracies to obtain the final performance measure. Because different chains have different lengths, the used measure can differ greatly from the standard accuracy. The rationale for this is to mimic the real-life situation, in which a new protein is sequenced, and researchers are interested in the predicted properties based on the entire protein sequence, independent of its length.

4 The GAssist Learning Classifier System

GAssist [2] is a Pittsburgh Genetic-Based Machine Learning system descendant of GABIL [9]. The system applies an almost standard generational GA, which evolves individuals that represent complete problem solutions. An individual consists of an ordered, variable-length rule set.

A fitness function based on the Minimum Description Length (MDL) principle [20] is used. The MDL principle is a metric applied to a theory (being a rule set here) which balances the complexity and accuracy of the rule set. The details and rationale of this fitness formula are explained in [2]. The system also uses a windowing scheme called ILAS (incremental learning with alternating strata) [4] to reduce the run-time of the system, especially for dataset with hundreds of thousands of instances, as in this paper. This mechanism divides the training set into several non-overlapping subsets and chooses a different subset at each GA iteration for the fitness computations of the individuals.

We have used the GABIL [9] rule-based knowledge representation for nominal attributes and the adaptive discretization intervals (ADI) rule representation [2] for real-valued ones. Section 5 shows an example of a rule set generated by GAssist using the GABIL representation. To initialize each rule, the system chooses a training example and creates a rule that guarantees to cover this example [3].

Table 1: Input attribute definitions for the tested datasets

Att. source	Description	Type	Cardinality
Len	Number of residues in a protein chain	real-valued	1 attribute
FreqRes	Frequencies of appearance of the each AA type in the protein chain	real-valued	20 attributes
AA-type	The AA type of a window of $\pm M$ residues around the target residue	nominal	$2M+1$ attributes
PredAveCN	Predicted average CN of a protein	nominal	1 attribute
PredSS	Predicted secondary structure of the $\pm M$ residues around the target residue	nominal+real-valued	$2*(2M+1)$ attributes

Table 2: Definition of the input attributes for all the used datasets. M =window size

Domain	Attributes	#real-valued att.	#nominal att.	total #att.
CN1	AA-type	0	$2*M+1$	$2*M+1$
CN2	AA-type,PredSS	$2*M+1$	$(2*M+1)*2$	$(2*M+1)*3$
CN3	AA-type,Len,FreqRes	21	$2*M+1$	$2*M+22$
CN4	AA-type,Len,FreqRes,PredSS	$2*M+22$	$(2*M+1)*2$	$(2*M+1)*3+21$
CN5	AA-type,PredAveCN	0	$2*M+2$	$2*M+2$
CN6	AA-type,PredAveCN,PredSS	$2*M+1$	$(2*M+1)*2+1$	$(2*M+1)*3+1$

Finally, we have used a mechanism wrapped over GAssist to boost its performance. We generate several rule sets using GAssist with different random seeds and combine them as an ensemble, combining their predictions using a simple majority vote. This approach is similar to Bagging [7]. GAssist used its standard parameters [2] with the 1000 iterations for the runs in the first stage, and 20000 for the runs in the second stage, 150 strata for the ILAS windowing scheme, and 10 rule sets per ensemble.

5 Results

The results are reported in three stages. In the first stage we do some quick tests of GAssist using the CN1 dataset on a broad range of values for the distance cut-off d_c , residue window size M and two classes (low/high CN). The aim of these tests is to determine the optimal settings of the dataset for GAssist, and only test the other datasets on these settings.

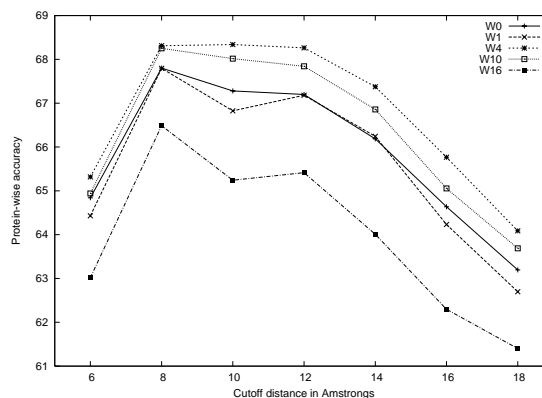
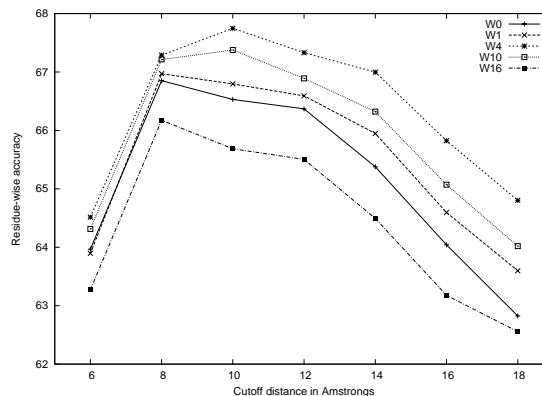
In the second stage, we test GAssist on the 36 defined datasets, and compare its performance to three other systems: C4.5 [19], a rule induction system, Naive Bayes [11], a Bayesian learning algorithm and LIBSVM [8], a support vector machine using RBF kernels. We have used the WEKA implementations [25] of both C4.5 and Naive Bayes. Student t-tests are applied to the results of the experiments to determine, for each dataset if the best method is significantly better than the other algorithms using a confidence interval of 95%. The Bonferroni correction [17] for multiple pair-wise comparisons has been used.

The third stage includes an analysis of the interpretability of the solutions generated by GAssist, extracting some general metrics from these solutions and relating the generated rules to physico-chemical properties of the proteins.

5.1 First stage

In this first stage we run GAssist using the CN1 dataset generated from the following parameters: $d_c \in \{6, 8, 10, 12, 14, 16, 18\text{\AA}\}$, window size $\in \{0, 1, 4, 10, 16\}$ and constant w of CN definition: 3. We summarize these results in figures 2 and 3, which report the accuracy of all combinations of d_c and window sizes. The first figure reports protein-wise accuracy and the second figure residue-wise accuracy. In these experiments, both the residue-wise and the protein-wise accuracy are reported. The best combination of parameters is d_c of 10Å and a window size of 4 (4 residues each side of the target one), which gives a protein-wise accuracy of 68.3% and a residue-wise accuracy of 67.7% for

the two-classes UF dataset.

Figure 2: Protein-wise accuracy for the first stage experiments**Figure 3: Residue-wise accuracy for the first stage experiments**

5.2 Second stage

Using the parameters optimized in the previous experiment, we tested the selected learning systems on the CN1..CN6 datasets, as summarized in table 3. Each value is the accuracy definition stated in section 3 averaged over the test sets. The t-tests applied to these results are summarized in table 4, where each cell counts how many times the method in the row significantly outperforms the method in the column with a confidence level of 95%.

Table 3: Accuracy of the tested systems on the CN1..CN6 datasets. A ● marks methods that were significantly outperformed by GAssist, while a ○ marks methods that significantly outperformed GAssist in that dataset. Student T-tests with 95% confidence level were applied

Dataset	System	Uniform frequency class def.			Uniform length class def.		
		2 classes	3 classes	5 classes	2 classes	3 classes	5 classes
CN1	GAssist	69.0±0.5	50.7±0.5	34.2±0.3	75.9±0.8	63.8±0.9	46.5±0.9
	Naive Bayes	68.7±0.5	50.7±0.6	34.5±0.5	76.3±0.7	64.0±0.8	47.0±0.8
	C4.5	68.1±0.4●	49.4±0.4●	30.9±0.6●	75.0±0.7	63.3±0.9	46.1±0.9
	LIBSVM	68.9±0.4	51.4±0.6	35.5±0.6○	77.4±0.8○	65.0±0.8○	46.9±0.8
CN2	GAssist	71.0±0.5	53.6±0.4	35.9±0.4	79.0±0.7	65.8±0.9	47.0±0.9
	Naive Bayes	66.3±0.7●	49.8±0.6●	33.4±0.5●	72.1±0.7●	61.3±1.0●	39.9±0.7●
	C4.5	70.6±0.6	52.8±0.4●	33.6±0.4●	77.9±0.6●	66.7±0.9	46.5±1.0
	LIBSVM	72.7±0.6○	57.0±0.6○	39.0±0.5○	79.9±0.6○	69.1±1.0○	48.7±0.9○
CN3	GAssist	70.9±0.5	52.6±0.7	35.7±0.6	77.2±1.1	65.1±0.9	47.0±0.8
	Naive Bayes	67.7±0.7●	50.4±0.9●	34.1±0.8●	76.2±0.9	62.5±1.1●	43.5±1.4●
	C4.5	69.9±0.5●	50.1±0.7●	31.1±0.6●	77.0±0.9	65.1±0.7	44.0±0.7●
	LIBSVM	72.0±0.4○	55.3±0.8○	38.0±0.5○	79.3±1.0○	68.1±0.8○	47.2±0.7
CN4	GAssist	72.7±0.4	55.3±0.6	37.5±0.4	80.1±0.8	66.9±0.9	47.7±0.9
	Naive Bayes	69.8±0.8●	52.7±0.9●	36.1±0.9●	76.9±1.0●	64.2±0.9●	43.7±1.2●
	C4.5	72.2±0.4	53.4±0.5●	34.0±0.5●	79.1±0.7	67.6±0.7	45.2±0.7●
	LIBSVM	75.9±0.4○	59.9±0.7○	41.9±0.4○	81.7±0.7○	71.5±0.8○	50.8±0.9○
CN5	GAssist	71.2±0.5	52.9±0.9	35.9±0.8	77.2±0.9	65.3±0.8	47.1±0.8
	Naive Bayes	71.5±0.5	54.0±0.8○	37.3±0.7○	78.4±0.8	67.2±0.8○	48.7±0.7○
	C4.5	70.3±0.6	51.7±0.8●	33.1±0.8●	77.1±1.0	65.8±0.7	47.0±0.8
	LIBSVM	72.0±0.6○	55.0±0.8○	37.8±0.7○	79.1±0.9○	67.9±0.7○	47.7±0.9
CN6	GAssist	72.9±0.4	55.9±0.7	37.8±0.7	80.3±0.7	67.3±0.8	47.8±0.8
	Naive Bayes	68.5±0.5●	52.0±0.7●	35.1±0.6●	75.2±0.9●	63.9±0.8●	42.8±0.5●
	C4.5	72.4±0.5	54.5±0.6●	35.5±0.6●	79.5±0.7	68.4±0.7○	48.1±0.8
	LIBSVM	75.8±0.4○	59.8±0.6○	41.7±0.6○	81.5±0.8○	71.3±0.7○	50.8±0.8○

Table 4: Number of times a method significantly outperforms another

	GAssist	Naive Bayes	C4.5	LIBSVM	Times outperforming
GAssist	-	23	17	0	40
Naive Bayes	4	-	11	0	15
C4.5	1	16	-	0	17
LIBSVM	31	26	34	-	91
Times outperformed	36	65	62	0	

The results show some general trends across all learning methods. First of all, the UL class definition leads to better accuracy than the UF definition for all datasets. This reflects the capacity of the UL definition to adapt itself to the physical reality of the proteins as its criterion is based on the dimensions of the CN domain. The UF definition, a priori, may look more appropriate from a machine learning point of view, as it creates well balanced class distributions. However, in this case the class frontiers may separate examples that are practically equal. Nevertheless, it may be worth studying the amount of information contributed by both measures. It may be possible that the UF definition, although leading to lower accuracy, provides more added value to a final 3D protein structure predictor.

From the tested sets of input attributes, we can say that all the different kind of attributes contribute to increasing the predictive accuracy of the tested systems. We can quantify the contribution of the predicted secondary structure information as an accuracy increase of 2-3% on most datasets and learning systems, comparing the performance of CN1-CN2, CN3-CN4 and CN5-CN6.

Another general observation is that the use of either

global protein information or the protein-wise predicted average CN is equivalent, as we observe that most systems achieve similar performance in the CN3-CN5 and CN4-CN6 datasets. The contribution of this kind of input information to the accuracy increase is 1.5-2%. GAssist had an average run-time ranging from 9.5 to 14 hours in the CN3 dataset, while it had a run-time ranging from 0.3 to 1.1 hours in the CN5 dataset. The main reason for this is the larger search space, although the mix or real-valued and nominal attributes requires the use of a less efficient knowledge representation. Considering this issue and the fact that the solutions generated by the CN5 datasets use less attributes than the ones generated by CN3 (therefore, more readable) it is reasonable to recommend the use of the latter kind input attribute for future experiments.

Looking at the specific results of each learning method, we observe that both GAssist and C4.5 obtain their highest accuracy in the CN6 dataset, while Naive Bayes obtains its highest accuracy in the CN5 dataset, and LIBSVM in CN4. LIBSVM achieves the best accuracy in 33 of the 36 datasets, as reflected by the t-tests, where LIBSVM outperforms the other methods in 91 of 108 times, and it is never significantly outperformed. The t-tests place GAssist in the second position of the ranking for both the number of times it outperforms C4.5 and Naive Bayes and the number of times it is outperformed by the other methods. Finally, both C4.5 and Naive Bayes perform comparably, at the bottom of the ranking.

5.3 Interpretability and explanatory power of GAssist results

Table 5 summarizes two simple metrics of the solutions: the average number of rules per rule set and the average number of expressed attributes in the generated rules. We see that GAssist creates compact solutions, ranging from just 2 rules in the CN1 - 2 classes - UL dataset to 7.5 rules in the CN1 - 5 classes - UF dataset. At most, an average of 11.8 attributes

Table 5: Complexity measures of the GAssist solutions on the CN1..CN6 datasets. #rules = average number of rules per rule set. Exp. Att.= average number of expressed attributes per rule

Dataset	Metric	Uniform frequency class def.			Uniform length class def.		
		2 classes	3 classes	5 classes	2 classes	3 classes	5 classes
CN1	#rules	6.5±1.1	6.4±0.8	7.5±0.7	2.0±0.0	7.1±0.6	5.4±0.6
	Exp. Att	6.6±3.2	6.4±3.1	6.9±3.0	4.2±4.2	7.2±3.0	6.3±3.3
CN2	#rules	6.7±1.0	6.5±0.7	7.1±0.3	5.0±0.1	5.8±0.7	5.8±0.7
	Exp. Att	9.9±4.7	9.3±4.6	9.8±4.5	11.5±6.0	8.0±4.3	9.5±4.9
CN3	#rules	5.4±0.6	5.4±0.5	6.2±0.4	4.1±1.5	6.3±0.7	5.6±0.6
	Exp. Att	7.5±4.0	7.2±3.9	7.7±3.8	8.2±4.8	6.4±3.7	7.6±3.9
CN4	#rules	5.9±1.0	6.5±0.7	6.9±0.4	5.0±0.2	5.7±0.7	5.6±0.6
	Exp. Att	9.8±5.0	9.7±4.8	10.0±4.7	11.8±6.4	7.4±4.7	9.7±5.1
CN5	#rules	6.3±0.9	6.6±1.0	6.5±0.7	2.0±0.3	6.6±0.6	5.6±0.7
	Exp. Att	7.3±3.5	7.2±3.4	7.3±3.4	4.5±4.5	6.8±3.3	7.0±3.6
CN6	#rules	6.4±1.0	7.1±0.7	7.0±0.4	5.0±0.2	6.4±0.7	5.8±0.7
	Exp. Att	10.0±4.9	10.3±4.7	9.9±4.6	13.1±7.5	9.0±5.4	10.3±5.8

were expressed in the CN4 - 2 classes - UL dataset, which has a total of 42 attributes. In comparison, C4.5 sometimes generated solutions with as many as 8000 leaves, and LIB-SVM used around 160000 instances from the training set as support vectors. No simple complexity measure can be extracted from Naive Bayes.

The case of the CN1 dataset using the uniform length classes definition and two classes is especially interesting. In this dataset GAssist always generated solutions with just two rules, obtaining an average accuracy of 75.9%. One such rule set is shown below:

1. If $AA_{-4} \notin \{X\}$ and $AA_{-3} \notin \{D, E, Q\}$ and $AA_{-1} \notin \{D, E, Q\}$ and $AA \in \{A, C, F, I, L, M, V, W\}$ and $AA_1 \notin \{D, E, P\}$ and $AA_2 \notin \{X\}$ and $AA_3 \notin \{D, E, K, P, X\}$ and $AA_4 \notin \{E, K, P, Q, R, W, X\}$ then class is 1
2. Default class is 0

$AA_{\pm n}$ denotes the AA type at the position $\pm n$ in respect to the target residue. The AA type is represented using the one letter code, plus the symbol X that indicates end of chain, for the case when the window overlaps with the beginning or the end of the protein chain.

We see two types of predicates: those stating if the AA type of a certain position of the window belongs or does not belong to a certain set. When the number of AA types that the predicate for a certain residue can take includes more than 10, GAssist generates the complementary predicate to produce a more compact solution. Therefore, all the predicates defined as \in are more specific than the ones defined as \notin . The more specific attributes are usually also the most relevant ones, and in this rule set we only have one such predicate: the one associated to the target residue. It is reasonable to expect that the more relevant attributes are those associated directly to the residue for which we are predicting its CN.

Table 6 contains the average number of AA types included in the predicate associated to each window position, for all the rule sets produced for this dataset, which is a generality (complementary of specific) metric of each window position. This table also reports the percentage of times that each window position was expressed in the generated rule. A non expressed attribute is irrelevant for the prediction. We observe that the window positions to the right of the target residue are more relevant than the ones to the left, and that the window positions ± 2 are the most irrelevant ones. Further analysis should be performed to determine if there is a

physical explanation for this issue or if it is the effect of a GA positional bias [6].

Table 6: Expression and generality rate for the rule sets generated by GAssist for the CN1 dataset and uniform-length class definition

Window position	Expression rate	Generality rate
-4	95%	94.5%±4.6
-3	99%	88.1%±4.3
-2	57%	98.2%±2.5
-1	100%	84.7%±5.7
0	100%	39.4%±2.3
1	100%	83.5%±3.2
2	80%	96.2%±3.1
3	100%	78.8%±5.8
4	100%	78.5%±4.7

Moreover, we can extract a simple physico-chemical explanation of such predicates: the set of AA types contained in the predicate associated to the target residue (A,C,F,I,L,M,V,W) are all hydrophobic [15]. Hydrophobic residues are usually found in the inner part of a protein in native state. Therefore it is logical that they present higher CN than the other residues, as this rule predicts high CN. On the other hand, from the rest of predicates of the rule set, the more frequently appearing AA types in the negated predicates are D and E, which are negatively charged. This type of AA usually appears on the surface of the proteins, so it is sensible that they are not included in the predicates of a rule intended for predicting a high CN, that as stated before, usually appears in the core of proteins.

Table 7 extends this analysis to all the rule-sets generated for this dataset, reporting two metrics: (1) the frequency of appearance of each AA type for each window position, and (2) the average appearance frequency of each AA type for all positions. From this average we obtain a ranking of specificity of each AA type: Glutamine (E) and Proline (P) the two AA types appearing less often. On the other hand Alanine (A), Cysteine (C), Phenylalanine (F), Isoleucine (I), Leucine (L), Methionine (M) and Valine (V) appear in more than 95% of all positions, therefore being the less specific AA types for predicting a high value for the CN.

Table 8 analyzes these rules from a slightly different point of view: ranking the AA types for each window position. As we have already observed that the predicate for the central residue takes a different form compared to the rest of predicates, we analyze the positions featured in the predicates. For the central residue, after the 8th AA type in the ranking all frequencies are very close to 0, for the other ones, we do not find a frequency less than 95% until position 15th

Table 7: Frequency of appearance in percentage of each AA type by window position in the generated rules for the CN1 dataset and uniform-length class definitions

Pos.	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
-4	100	96	100	44	99	100	98	100	93	100	100	100	100	84	94	100	100	100	83	99
-3	100	94	14	12	100	100	100	100	100	100	100	89	94	66	100	100	100	100	94	99
-2	100	94	100	98	100	100	99	100	100	100	99	97	88	98	99	98	99	100	94	100
-1	100	94	41	43	98	100	100	100	18	100	100	100	8	96	99	100	100	100	96	100
0	100	100	0	0	100	0	0	100	0	100	100	0	0	0	0	0	0	100	82	5
1	100	97	4	4	100	99	98	100	96	100	100	81	0	94	100	99	100	100	99	100
2	100	98	98	100	100	100	98	100	97	100	97	100	38	100	100	100	100	100	98	100
3	100	98	55	11	100	100	100	100	1	100	100	96	2	47	66	100	99	100	100	100
4	100	99	94	1	100	100	98	100	1	100	100	96	66	1	28	100	100	100	85	100
Ave.	100.0	96.7	56.2	34.8	99.7	88.8	87.9	100.0	56.2	100.0	99.6	84.3	44.0	65.1	76.2	88.6	88.7	100.0	92.3	44.6

of the ranking. For the non-central positions, the interesting columns are the ones at the bottom of the ranking. We can observe that Proline (P) and Glutamine (E) are the less frequent AA types for seven of the nine window positions.

Therefore, we can extract sound explanations from the generated rules, and we have found more paths of analysis: analyzing the specificity degree of the used attributes and windows positions, and relating the generating predicates with physical/chemical properties.

6 Discussion

The discussion section is focused on the comparison of learning methods in these datasets. GAssist performs better than Naive Bayes and C4.5 but worse than LIBSVM. The direct comparison of GAssist and C4.5 is clearly favorable to GAssist, and this is important as these two systems use a very similar knowledge representation. GAssist is better than C4.5 at exploring the search space of the kind of solutions that an axis-parallel knowledge representation can offer. The comparison with LIBSVM is less favorable for GAssist, as it was outperformed in most datasets, especially in those with real-valued input attributes, which may indicate that the non-linear knowledge representation used by LIBSVM is superior for this kind of data.

Nevertheless, there are several strong points that back the use of GAssist. The first of them, already analyzed in the previous section is the explanatory power of GAssist solutions. From a pure machine learning point of view, these solutions are extremely compact, both in number of rules and also in number of expressed attributes. Moreover, it is quite easy to extract practical real-world explanations from the generated rules. On the other hand, it is quite difficult to extract an explanation from LIBSVM solutions. Actually the only easy metric to measure the size of the LIBSVM solutions is the number of examples selected in the training stage as support vectors. This number of examples can be huge, containing around 70-90% of the training set (approx. 236000 instances).

Another issue of concern is the run-time. Although GAssist is not a fast learning system, it is considerably faster than LIBSVM. GAssist run time on these datasets ranged between 0.3 to 24 hours, while LIBSVM run time ranged from 21 hours to 4.5 days. Even more critical is the time spent at the test stage. While LIBSVM in some cases took hours to predict all examples in the test set, GAssist used approximately about a minute to use its ensemble of rule-sets to produce the test predictions. This issue is very important, because the final goal of the line of research where this work is included is to create an online web-based 3D protein structure prediction server, where multiple users would normally want to predict tens, if not hundreds, of protein structural

features at a time. Our experience with www.proksi.net web server for protein structure comparison indicates that in an exploitation environment such as this the run-time is critical, and in this aspect GAssist can be faster than LIBSVM by two orders of magnitude.

7 Conclusions and further work

In this paper we tested whether using classification we could improve the accuracy over the current state-of-the-art of predicting protein residues CN. The current state-of-the-art [13] uses a real-valued definition of CN that was predicted as a regression, rather than classification, problem. The tests reported in this paper included six different sets of input attributes for the domain, some of them using predicted information, or global protein chain information. We have two criteria to convert the real-valued CN definition into a finite set of classes, and three version of the dataset with two, three and five classes. All these datasets were tested using four different machine learning approaches, but mainly focusing our efforts in an evolutionary computation-based machine learning system, called GAssist.

The experiments were useful from several aspects. We identified the contribution that each of the tested types of input information can provide to the predictive performance of the system. We compared two different class definitions for the CN domain, and we assessed the performance of GAssist on these domains, comparing it to three other machine learning systems. From this comparison, LIBSVM gives better predictive performance than GAssist, but on the other hand, it is much slower and the generated solutions have little explanatory power. The explanatory power is the strongest point of GAssist, as it can generate very compact solutions that are easy to interpret and easy to justify by using domain knowledge such as physical properties of the proteins.

Finally, there are several lines of further work. Focusing on the current kind of experiments, we will test other kind of input information and improve GAssist in order to boost the performance on the current CN definitions. On a higher level, we have investigated two possible class partitions criteria. It would be interesting to investigate which of these options, plus other CN definitions gives more information to the user, as the final goal of predicting CN is to integrate this predictions into a 3D protein structure prediction system. The explanatory analysis of the generated rule-sets would also be very useful, and not just to understand the GAssist predictions, but also in order to identify information that can be fed back to GAssist to improve its learning process and to better understand aspects of protein folding.

Table 8: Ranking of appearance of the AA type by window position in the generated rules for the CN1 dataset and uniform-length class definitions

Pos.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
-4	V-100	T-100	S-100	P-100	N-100	M-100	L-100	I-100	G-100	D-100	A-100	Y-99	F-99	H-98	C-96	R-94	K-93	Q-84	W-83	E-44
-3	V-100	T-100	S-100	R-100	M-100	L-100	K-100	I-100	H-100	G-100	F-100	A-100	Y-99	W-94	P-94	C-94	N-89	Q-66	D-14	E-12
-2	Y-100	V-100	L-100	K-100	I-100	G-100	F-100	D-100	A-100	T-99	R-99	M-99	H-99	S-98	Q-98	E-98	N-97	W-94	C-94	P-88
-1	Y-100	V-100	T-100	S-100	N-100	M-100	L-100	I-100	H-100	G-100	A-100	R-99	F-98	W-96	Q-96	C-94	E-43	D-41	K-18	P-8
0	V-100	M-100	L-100	I-100	F-100	C-100	A-100	W-82	Y-5	T-0	S-0	R-0	Q-0	P-0	N-0	K-0	H-0	G-0	E-0	D-0
1	Y-100	V-100	T-100	R-100	M-100	L-100	I-100	F-100	A-100	W-99	S-99	G-99	H-98	C-97	K-96	Q-94	N-81	E-4	D-4	P-0
2	Y-100	V-100	T-100	S-100	R-100	Q-100	N-100	L-100	I-100	G-100	F-100	E-100	A-100	W-98	H-98	D-98	C-98	M-97	K-97	P-38
3	Y-100	W-100	V-100	S-100	M-100	L-100	I-100	H-100	G-100	F-100	A-100	T-99	C-98	N-96	R-66	D-55	Q-47	E-11	P-2	K-1
4	Y-100	V-100	T-100	S-100	M-100	L-100	I-100	G-100	F-100	A-100	C-99	H-98	N-96	D-94	W-85	P-66	R-28	Q-1	K-1	E-1

8 Acknowledgments

We acknowledge the support of the UK Engineering and Physical Sciences Research Council (EPSRC) under grants GR/T07534/01, GR/62052/01 and GR/S64530/01 and the Biotechnology and Biological Sciences Research Council (BBSRC) under grant BB/C511764/1.

9 References

- [1] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res*, 25:3389–3402, 1997.
- [2] J. Bacardit. *Pittsburgh Genetics-Based Machine Learning in the Data Mining era: Representations, generalization, and run-time*. PhD thesis, Ramon Llull University, Barcelona, Catalonia, Spain, 2004. <http://www.cs.nott.ac.uk/~jqb/publications/thesis.pdf>.
- [3] J. Bacardit. Analysis of the initialization stage of a pittsburgh approach learning classifier system. In *GECCO 2005: Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1843–1850. ACM Press, 2005.
- [4] J. Bacardit, D. Goldberg, M. Butz, X. Llorà, and J. M. Garrell. Speeding-up pittsburgh learning classifier systems: Modeling time and accuracy. In *Parallel Problem Solving from Nature - PPSN 2004*, pages 1021–1031. Springer-Verlag, LNCS 3242, 2004.
- [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*, 4:575–602, 2003.
- [6] L. Booker. Recombination distribution for genetic algorithms. In *Foundations of Genetic Algorithms 2*, pages 29–44. Morgan Kaufmann, 1993.
- [7] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [8] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*. Department of Computer Science and Information Engineering, National Taiwan University, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [9] K. A. DeJong, W. M. Spears, and D. F. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 13(2/3):161–188, 1993.
- [10] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
- [11] G. H. John and P. Langley. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann Publishers, San Mateo, 1995.
- [12] D. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *J Mol Biol*, 292:195–202, 1999.
- [13] A. R. Kinjo, K. Horimoto, and K. Nishikawa. Predicting absolute contact numbers of native protein structure from amino acid sequence. *Proteins*, 58:158–165, 2005.
- [14] H. Liu, F. Hussain, C. L. Tam, and M. Dash. Discretization: An enabling technique. *Data Mining and Knowledge Discovery*, 6(4):393–423, 2002.
- [15] C. D. Livingstone and G. J. Barton. Protein sequence alignments: a strategy for the hierarchical analysis of residue conservation. *Computer Applications in the Biosciences*, 9(6):745–756, 1993.
- [16] R. MacCallum. Striped sheets and protein contact prediction. *Bioinformatics*, 20:I224–I231, 2004.
- [17] R. G. Miller. *Simultaneous Statistical Inference*. Springer Verlag, New York, 1981. Heidelberg, Berlin.
- [18] T. Noguchi, H. Matsuda, and Y. Akiyama. Pdb-reprdb: a database of representative protein chains from the protein data bank (pdb). *Nucleic Acids Res*, 29:219–220, 2001.
- [19] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- [20] J. Rissanen. Modeling by shortest data description. *Automatica*, vol. 14:465–471, 1978.
- [21] C. Sander and R. Schneider. Database of homology-derived protein structures. *Proteins*, 9:56–68, 1991.
- [22] Y. Shao and C. Bystroff. Predicting interresidue contacts using templates and pathways. *Proteins*, 53:497–502, 2003.
- [23] M. Stout, J. Bacardit, J. Hirst, N. Krasnogor, and J. Blazewicz. From hp lattice models to real proteins: coordination number prediction using learning classifier systems. In *4th European Workshop on Evolutionary Computation and Machine Learning in Bioinformatics 2006 (to appear)*, 2006.
- [24] S. W. Wilson. Classifier fitness based on accuracy. *Evolutionary Computation*, 3(2):149–175, 1995.
- [25] I. H. Witten and E. Frank. *Data Mining: practical machine learning tools and techniques with java implementations*. Morgan Kaufmann, 2000.
- [26] Y. Zhao and G. Karypis. Prediction of contact maps using support vector machines. In *Proc. of the IEEE Symposium on Bioinformatics and BioEngineering*, pages 26–36. IEEE Computer Society, 2003.