Data Mining in Proteomics with Learning Classifier Systems

Jaume Bacardit[1], Michael Stout[1], Jonathan D. Hirst[2] and Natalio Krasnogor[1]

[1] Automated Scheduling, Optimization and Planning research group, School of Computer Science and IT, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK {jqb,mqs,nxk}@cs.nott.ac.uk
[2] School of Chemistry, University of Nottingham, University Park, Nottingham NG7 2RD, UK jonathan.hirst@nottingham.ac.uk

Abstract. The era of data mining has provided renewed effort in the research of certain areas of biology that for their difficulty and lack of knowledge were and are still considered unsolved problems. One such problem, which is one of the fundamental open problems in computational biology is the prediction of the 3D structure of proteins, or protein structure prediction (PSP). The human experts, with the crucial help of data mining tools, are learning how protein fold to form their structure, but are still far from providing perfect models for all kinds of proteins. Data mining and knowledge discovery are totally necessary in order to advance in the understanding of the folding process. In this context, Learning Classifier Systems (LCS) are very competitive tools. They have shown in the past their competence in many different data mining tasks. Moreover, they provide human-readable solutions to the experts that can help them understand the PSP problem. In this chapter we describe our recent efforts in applying LCS to PSP related domains. Specifically, we focus in a relevant PSP subproblem, called Coordination Number (CN) prediction. CN is a kind of simplified profile of the 3D structure of a protein. Two kinds of experiments are described, the first of them analyzing different ways to represent the basic composition of proteins, its primary sequence, and the second one assessing different data sources and problem definition methods for performing competent CN prediction. In all the experiments LCS show their competence in terms of both accurate predictions and explanatory power.

1 Introduction

The prediction of the 3D structures of proteins is both a fundamental and difficult problem in computational biology. The usual approach to solve this problem is to use a divide-and-conquer approach and thus predict some specific attributes of a protein native structure, such as the secondary structure, solvent accessibility or coordination number. Accurate predictions of these subproblems and proper understanding of the contribution of each subproblem and the rationale behind these predictions is crucial to integrating them successfully into a final 3D protein structure prediction (PSP).

Learning Classifier Systems (LCS) [1, 2] are a class of evolutionary computation based machine learning techniques that could be used to tackle these issues in PSP. As they have shown in the past their competence on data mining problems [3, 4] using diverse LCSs such as XCS [5], GALE [6] or GAssist [7]. Importantly, they provide human-readable and highly interpretable solutions to the prediction problem, usually rule sets. The understanding of these solutions can lead to improvements in the way the information is represented and also more efficient integration of them into the final 3D structure prediction.

This chapter shows some competitive advantages of LCS over other techniques and also some of the challenges facing LCSs as they are applied to mining PSP datasets. Our chapter collects our recent research [8–10] using GAssist [7], a recent Pittsburgh approach LCS [2]. GAssist was applied to a PSP problem called coordination number (CN), which is defined as the prediction, for a given residue, of the number of residues from the same protein that are in contact with it. Two residues are said to be in contact when the distance between the two is below a certain threshold. The CN feature is a simplified profile of the density of the 3D structure of a protein, and therefore can be helpful in constraining the vast search space of the full PSP problem.

The chapter is divided in two parts. In the first part we analyze some alternative representations for the most basic form of information of a protein: its primary sequence. The primary sequence of a protein is a chain formed by 20 possible types of amino acids. Therefore the simplest way of representing a protein is by a string of a 20-letter alphabet. Moreover, these amino acids can be clustered based on physical and chemical properties, which lead to more simplified alphabets of representing the primary chain. The HP alphabet (hydrophobic/polar) is perhaps the best known example of reduced alphabets. This simplification is usually combined with a reduction of the number of spatial degrees of freedom by restricting the atom or residue locations to those of a lattice [11, 12]. We compare some of these representations for coordination number prediction and test them using GAssist as well as other machine learning techniques.

In the second part of the chapter we show our evaluation of various types of input information, class definitions and learning algorithms applied to coordination number prediction of real proteins. Our aim is to perform a rigorous evaluation of the contribution of various kinds of information and problem definitions towards predicting CN and also analyze the explanatory power that LCSs can offer in this dataset. All the reported experiments show how LCSs can perform competitively with other learning techniques. We also show that the solutions obtained are human-readable and have rich explanatory power. This is another important advantage of GAssist as the biologists are not only interested in the quality of the predictions, but also the reasons behind them.

The rest of the chapter is structured as follows: First, section 2 will contain background information and related work about proteins, CN prediction and HP lattice models. Next, section 3 will describe the main characteristics of GAssist, our machine learning system. Section 4 will detail the experimental procedure. The results of applying this experimental procedure will be reported in section

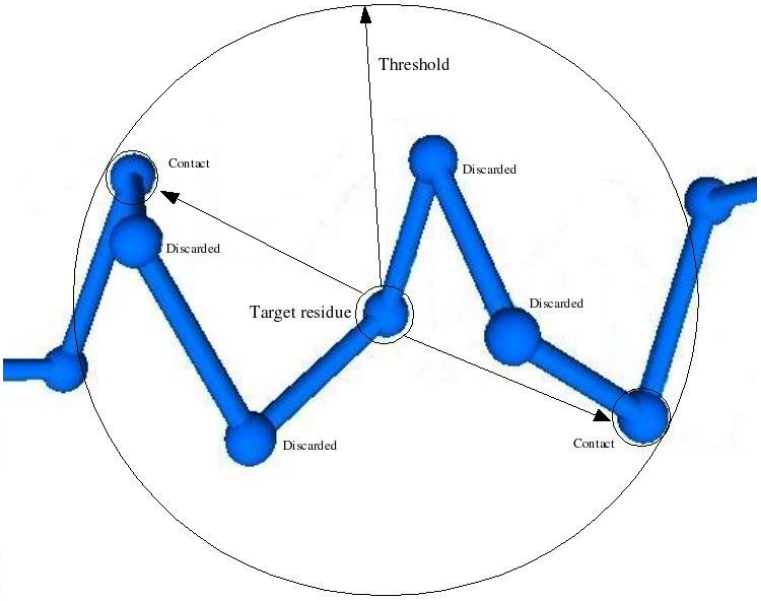
5. Next, section 6 will discuss the results presented in the previous section and, finally, section 7 will describe the conclusions and further work.

2 Problem definition

2.1 Protein Structure and Coordination Number Prediction

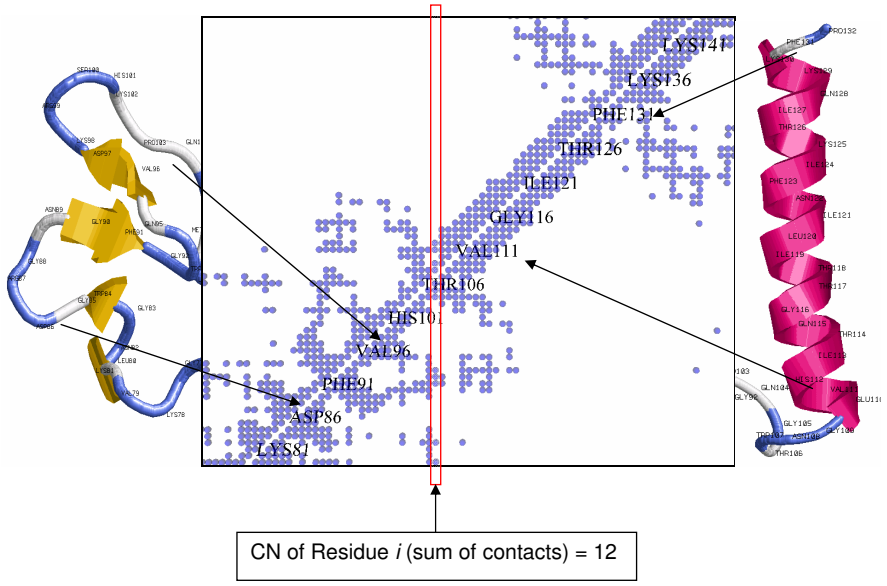
Proteins are heteropolymer molecules constructed as a chain of amino acids of 20 different types. This string of amino acids is known as the primary sequence. In the native state, the chain folds to create a 3D structure. The primary sequence arranges itself into secondary structure, consisting of local structures such as alpha helices, beta sheets or coils. These local structures can group in several conformations or domains forming a tertiary structure. Secondary and tertiary structure may form concomitantly. 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 the end product of the folding process indicating the number of other residues that end up near the target residue. Some of these contacts can be close in the protein chain but some other can be quite far apart, trivial contacts such as those with the immediate neighbour residues are ignored. Figure 1 contains a graphical representation of the CN of a residue in an alpha helix, given a minimum chain separation (ignored trivial contacts) of two. In this example, the CN of the target residue is two.

Fig. 1. Graphical representation of the CN of a residue



This problem is closely related to contact map (CM) prediction that seeks to predict, for all possible pairs of residues of a protein, whether they are in contact or not. When the contact map is represented as a binary matrix, the CN of a residue is the count of the number of ones in the row of the map associated with that residue. Figure 2 shows the relation between the native structure of a protein, a contact map and the coordination number of a residue. It also shows how different secondary structure elements of a protein are reflected as different kind of patterns in a contact map.

Fig. 2. Relation between a protein native structure and its contact map and the coordination number of its residues



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 [14], hidden markov models [15], a combination of self-organizing maps and genetic programming [16] and support vector machines [17].

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 [15], 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, besides the amino acid (AA) type of the residues in the primary chain, such as global information of the protein chain [13], position-specific scoring matrices (PSSM) computed from multiple sequences alignments [15, 14, 17, 16, 13] (mainly using PSI-BLAST [18]), predicted secondary structure [14, 17], predicted solvent accessibility [14], physical characteristics of the residues [8] or sequence conservation [17]. Contact maps for any protein dataset could be easily generated through our protein structure comparison web server at `http://www.procksi.net/` and used as raw data for data mining tasks.

2.2 HP Models

As protein structure prediction remains an unsolved problem, researchers have resorted to simplified protein models to try to gain understanding of both the process of folding and the algorithms needed to predict it [19, 20, 11, 21, 12]. Approaches have included fuzzy sets, cellular automata, L-systems and memetic algorithms [22–27]. One common simplification is to focus only on a representative atom of each residues (C-alpha or C-beta atoms) rather than all the atoms in the protein. A further simplification is to reduce the number of residue types to less than twenty by using representations based, for instance, on physicochemical properties such as hydrophobicity, as in the so called hydrophobic/polar (HP) models. A further simplification is to reduce the number of spatial degrees of freedom by restricting the atom or residue locations to those of a lattice [11, 12]. Lattices of various geometries have been explored, e.g., two-dimensional triangular and square geometries or three-dimensional diamond and face centered cubic [25].

In the HP model (and its variants) the 20 residue types are reduced to two classes: non-polar or hydrophobic (H) and polar (P) or hydrophilic. An n residue protein is represented by a sequence $s \in \{H, P\}^+$ with $|s| = n$. The sequence s is mapped to a lattice, where each residue in s occupies a different lattice cell and the mapping is required to be self-avoiding. The energy potential in the HP model reflects the propensity of hydrophobic residues to form a hydrophobic core.

In the HP model, optimal (i.e. native) structures are those that minimize the following energy potential:

$$E(s) = \sum_{i<j \ ; \ 1 \leq i,j \leq n} (\Delta_{i,j} \epsilon_{i,j}) \tag{1}$$

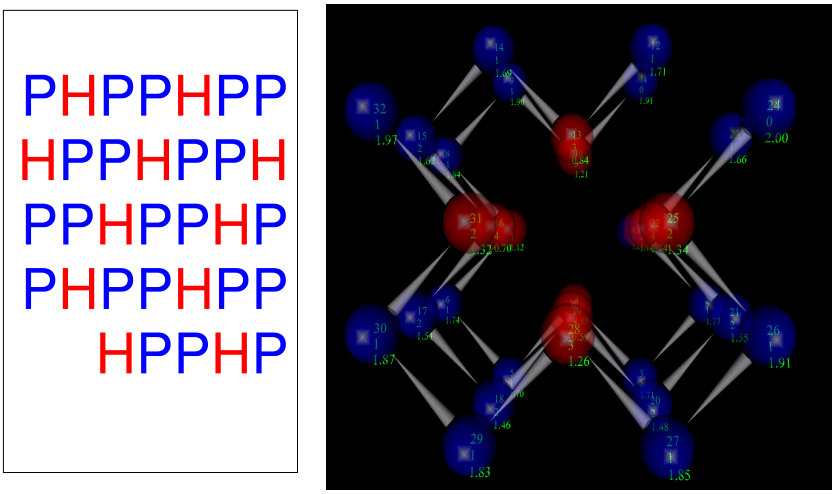
where

$$\Delta_{i,j} = \begin{cases} 1 \text{ if } i, j \text{ are in contact and } |i-j| > 1 \\ 0 \text{ otherwise} \end{cases} \tag{2}$$

In the standard HP model, contacts that are HP and PP are assigned an energy of 0 and an HH contact is assigned an energy of -1. Figure 3 shows a

protein sequence represented with the HP alphabet and its optimal structure when using a 3D cubic lattice.

Fig. 3. Representation of a protein sequence using HP alphabet and its optimal structure in the 3D cubic lattice



3 The GAssist Learning Classifier System

GAssist [7] is a Pittsburgh Genetic–Based Machine Learning system descendant of GABIL [28]. 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.

We have used the GABIL [28] rule-based knowledge representation for nominal attributes and the adaptive discretization intervals (ADI) rule representation [7] 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 [29].

A fitness function based on the Minimum Description Length (MDL) principle [30] is used. The MDL principle is a metric applied to a theory (a rule set

here) which balances its complexity and accuracy. Our specific MDL formulation promotes rule sets with as few rules as possible as well as rules containing predicates as simple as possible. The details and rationale of this fitness formula are explained in [7].

The system also uses a windowing scheme called ILAS (incremental learning with alternating strata) [31] to reduce the run-time of the system, specially for dataset with hundreds of thousands of instances, as in this chapter. This mechanism divides the training set into several non-overlapping strata and chooses a different stratum at each GA iteration for the fitness computations of the individuals. ILAS empirically showed in previous experiments not only to reduce the computational cost of GAssist but also to apply generalization pressure (complementary to the one applied by the MDL-based fitness function) that helped generating more compact and accurate solutions. Figure 4 shows the pseudocode of the ILAS windowing scheme.

Fig. 4. Pseudocode of the Incremental Learning with Alternating Strata scheme

Procedure Incremental Learning with Alternating Strata
Input : $Examples, NumStrata, NumIterations$
Initialize GA
$Examples$ = ReorderExamples($Examples$,$NumStrata$)
$Iteration = 0$
$StratumSize = size(Examples)/NumStrata$
While $Iteration < NumIterations$
  If $Iteration = NumIterations - 1$
    $TrainingSeg = Examples$
  Else
    $CurrentStratum = Iteration \ mod \ NumStrata$
    $TrainingSeg$ = examples from
      $Examples[CurrentStratum \cdot StratumSize]$ to
      $Examples[(CurrentStratum + 1) \cdot StratumSize]$
  EndIf
  Run one iteration of the GA with $TrainingSeg$
  $Iteration = Iteration + 1$
EndWhile
Output : Best set of rules from GA population

Finally, we have used a ensemble mechanism wrapped over GAssist to boost its performance. We generate several rule sets using GAssist with different initial random seeds and combine them as an ensemble, producing a consensus prediction using a simple majority vote. This approach is similar to Bagging [32] but simpler as, unlike Bagging, it does not need to scramble the training set to genereate slightly different classifiers to combine them as an ensemble. In previous work [33] we empirically evaluated this ensemble mechanism over a set of 25 real-world datasets from the University of California at Irvine (UCI)

repository [34]. On average, the ensemble obtained a test accuracy 2.5% higher than the standalone GAssist.

GAssist used its standard parameters [7] 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.

4 Experimental framework

4.1 Experimental battery I: Primary sequence and coordination number

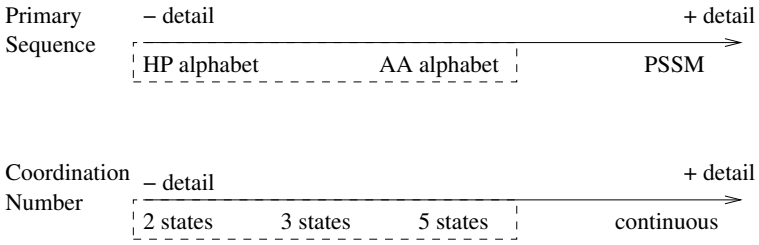
The first part of the experiments reported in this chapter will focus on the relation between the primary sequence attributes of a protein and CN prediction. To analyze this relation we will compare the CN prediction for simplified HP lattice model proteins (Lattice-HP) with the prediction of the same feature for real proteins using either all twenty amino acid types (Real-AA) or using only the HP representation (Real-HP). The characteristics of each dataset are summarized in table 1 and detailed in the rest of this subsection.

Table 1. Details of the data sets used in these experiments.

Name	**Lattice-HP**	**Real-HP/Real-AA**
Type	3D Cubic Lattice	Real Proteins
Number of Sequences	15	1050
Minimum Sequence Length	27	80
Maximum Sequence Length	48	2329
Total Hydrophobic	316	170493
Total Polar	309	84850
Total Residues	625	255343

Moreover, also several degrees of precision for the prediction of CN will be evaluated. This feature can be defined either as a integer variable of high cardinality or directly as a continuous variable. Therefore, in order to use classification methods for CN prediction, this feature has to be transformed into a more or less reduced finite set of labels. Thus, the experiments in this first battery contains datasets with varying degree on precision in two dimensions: the primary sequence (inputs of the classification problem) and the CN feature (output of the classification problem). Figure 5 shows these two dimensions of detail of both the primary sequence and the CN, naming some of the different possible choices of representation for both of them, marking with a dashed area the ones that have been used in this chapter. It is important to remark, however, that in some cases a larger degree of detail does not mean that the problem becomes more difficult, one such case is treating the CN as a continuous variable or as a discrete one. In the first case the problem becomes a regression domain, in the second one it becomes a classification domain. Both problems are equally challenging.

Fig. 5. Degrees of precision in each dimension of the CN prediction problem. The dashed areas mark the representation options explored in this chapter.



Real proteins dataset The datasets based on real proteins (Real-AA and Real-HP) use the CN definition proposed by Kinjo et al. [13] defined as follows. The distance used to determine contact is defined using the C_β atom (C_α for glycine) of the residues. The boundary of the sphere defined by the distance cutoff $d_c \in \Re^+$ is made smooth by using a sigmoid function. Also, a minimum chain separation of two residues is required. Formally, the CN (O_i^p) of the residue i of protein chain p is computed as:

$$O_i^p = \sum_{j:|j-i|>2} \frac{1}{1 + exp(w(r_{ij} - d_c))} \tag{3}$$

where r_{ij} is the distance between the C_β atoms of the ith and jth residues. The constant w determines the sharpness of the boundary of the sphere. A value of three for w was used for all the experiments. CN was computed using a distance cutoff of 10 Å. In previous work [9] we empirically tested different distance cutoffs, and 10 Å was the one that obtained highest accuracy.

This CN definition is real-valued. Therefore, it has to be converted into a set of finite classes so that it can be used as a classification dataset. The continuous CN domain will be discretized using the Uniform Frequency discretization algorithm [35]. Three numbers of states will tested for the experiments reported in this chapter, dividing the CN domain into two, three and five states.

The real protein dataset (Real-AA) was selected from PDB-REPRDB [36], an online server that automatically selects curated protein subsets given a set of criteria with the following conditions: less than 30% sequence identity, sequence length greater than 50, no membrane proteins, no nonstandard residues, no chain breaks, resolution better than 2 Å and having a crystallographic R factor better than 20%. Chains that had no entry in the HSSP [37] database were discarded. The final data set contains 1050 protein chains.

Each instance of the dataset has as class the CN of a residue belonging to some of these 1050 protein chains, and as input information we will use local information of the target residue and its closest neighbours in the chain: To generate the instance in the dataset we will use the most usual method, define a sliding window, centered initially over the first residue of a chain and that

will move one residue at a time. This window is centered over the target and includes $\pm N$ residues, meaning that the CN of the target will be predicted using as inputs the primary sequence representation of the residues in the window. Windows were generated for one, two and three residues at each side of the central residue. The set was divided randomly into ten pairs of training and test set using 950 proteins for training and 100 for testing in each set. These sets act in a similar way to a ten-fold cross-validation. The proteins included in each partition are reported in `http://maccl01.genes.nig.ac.jp/~akinjo/sippre/suppl/list/`. This same dataset was used to generate a real protein HP sequence dataset (Real-HP) by assigning each residue a value of Hydrophobic or Polar as shown in Table 2, following Broome and Hecht [38].

Table 2. Assignment of residues as Hydrophobic or Polar.

Residue (one letter code)	**Assignment**
ACFGILMPSTVWY	Hydrophobic
DEHKRQN	Polar

HP Lattice-based datasets For the Lattice-HP study, a set of structures from Hart's Tortilla Benchmark Collection (`http://www.cs.sandia.gov/tech_reports/compbio/tortilla-hp-benchmarks.html`) was used. This consisted of 15 structures on the simple cubic lattice, in which each residue can have up to six neighbours in the lattice, therefore the maximum CN value that a residue can have is also 6. Again, windows were generated for one, two and three residues at each side of a central residue and the CN class of the central residue assigned as the class of the instance. The instance set was partitioned into training/test sets using stratified ten-fold cross-validation. The process was repeated ten times to generate 10 sets of cross-validation folds. Each reported accuracy will be, therefore, the average of one hundred values. As in the real dataset, the CN of this domain will be divided into two, three and five states using an uniform frequency discretization algorithm.

Comparison of LCS performance The performance of GAssist in the Lattice-HP, Real-HP and Real-AA datasets will be compared against two well known machine learning methods. We chose two C4.5 [39] and Naive Bayes [40], using the WEKA [41] implementation of both of them. 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 [42] for multiple pair-wise comparisons has been used.

4.2 Experimental battery II: assessment of input information sources and class partitions for coordination number

The second part of the experiments reported in this chapter focuses exclusively on real proteins, assessing two different dimensions of the CN problem: (1) what is the contribution of several types of input information and (2) different criteria to define the classes of the CN domain. We will explore six different sets of input information, two class partition criteria and three different numbers of class partitions, testing in total 36 datasets.

Class definitions and protein dataset We will use again the Kinjo et al. [13] definition of CN as well as their protein dataset and training/test partitions, detailed in the previous subsection. As explained previously, the CN definition is continuous, and has to be discretized to handle the dataset as a classification domain. Previously, we used the uniform frequency (UF) unsupervised discretization to generate the class partitions. For these experiments, besides UF we will also test the uniform-length (UL) unsupervised discretization [35]. For each of them, as in the first stage experiments, we will test three numbers of states, dividing the CN domain into two, three and five states.

Explored input information Six sets of input attributes are evaluated in this battery of experiments. The first set corresponds to the representation used for the experiments in the first part of the chapter: 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*. 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 partition criteria (uniform lenght and uniform frequency) described above. 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 a window of residues around the target residue, using the PSI-PRED predictor [43]. 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 3 summarizes the input attributes used in the datasets, and table 4 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. Unlike battery I of experiments, Here we have only used one window size: four residues at each side of the target has been used, for both AA-type and PredSS types of input information.

Table 3. 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 ±4 residues around the target residue	nominal	9 attributes
PredAveCN	Predicted average CN of a protein	nominal	1 attribute
PredSS	Predicted secondary structure of the ±4 residues around the target residue	nominal+real-valued	18 attributes

Table 4. Definition of the input attributes for all the used datasets

Domain	Attributes	#real-valued att.	#nominal att.	total #att.
CN1	AA-type	0	9	9
CN2	AA-type,PredSS	9	18	27
CN3	AA-type,Len,FreqRes	21	9	30
CN4	AA-type,Len,FreqRes,PredSS	30	18	48
CN5	AA-type,PredAveCN	0	10	10
CN6	AA-type,PredAveCN,PredSS	9	19	28

Performance measure The accuracy metric used for these experiments is not the standard machine learning accuracy metric (#correct examples/#total examples). As is usual in the protein structure prediction field [13, 43], 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, protein-wise, accuracy metric. Because different chains have different lengths, the used measure can differ 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.

Comparison of LCS performance The performance of GAssist on these 36 datasets (6 sets of input attributes, 2 class definition criteria, 3 numbers of classes) will be compared against three other machine learning systems: C4.5 [39], a rule induction system, Naive Bayes [40], a Bayesian learning algorithm and LIBSVM [44], a support vector machine using RBF kernels. We have used the WEKA implementations [41] 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%. Again, the Bonferroni correction [42] for multiple pair-wise comparisons has been used.

5 Results

5.1 Experimental battery I

Lattice-HP datasets Table 5 compares the results of two, three and five state CN predictions for a range of window sizes for the GAssist LCS, Naive Bayes and C4.5 using the Lattice-HP dataset. As the number of states is increased the accuracy decreases from around 80% to around 51% for all algorithms. For each state as the window size is increased the accuracy increases by around 0.1-0.2%. With the exception of the C4.5 algorithm which shows a decrease in accuracy with increasing window size in two and three state predictions. There were no significant differences detected in these tests and thus all learning methods showed similar performance.

Table 5. Lattice-HP Prediction Accuracies.

Number of States	Algorithm	Window Size		
		1	2	3
2	GAssist	79.8 ±4.9	80.2 ±5.0	80.0 ±5.3
	C4.5	80.2 ±4.9	79.9 ±5.0	79.7 ±5.1
	NaiveBayes	79.8 ±4.9	80.0 ±4.9	80.2 ±5.0
3	GAssist	67.4 ±4.9	67.8 ±4.1	67.3 ±5.0
	C4.5	67.5 ±4.8	67.6 ±4.2	66.6 ±5.0
	NaiveBayes	67.2 ±4.6	67.3 ±4.4	67.5 ±4.8
5	GAssist	51.4 ±4.6	51.3 ±4.2	52.7 ±5.3
	C4.5	51.7 ±4.5	51.0 ±4.1	52.2 ±5.1
	NaiveBayes	51.7 ±4.6	52.3 ±4.3	51.9 ±5.6

Real proteins Table 6 compares the results of two, three and five state CN predictions on real proteins for the GAssist LCS, Naive Bayes and C4.5 using the Real-AA and Real-HP datasets.

When an HP sequence representation was used, an increase in the number of states is accompanied by a decrease in accuracy from around 63-64% to around 29-30% for all algorithms. For each state, as the window size is increased the accuracy increases by around 1%. Again, no significant differences were found between the methods for the Real-HP datasets.

Using full residue information, an increase in the number of states is accompanied by a decrease in accuracy from around 68% to around 34% for all

Table 6. CN Prediction Accuracies for the Real-HP and Real-AA datasets. A • means that GAssist outperformed the Algorithm to the left (5% t-test significance). A ∘ label means that the Algorithm on the left outperformed GAssist (5% t-test significance)

State	Algorithm	HP Based			Residue Based		
		Window Size			Window Size		
		1	2	3	1	2	3
2	GAssist	63.6±0.6	63.9±0.6	64.4±0.5	67.5±0.4	67.9±0.4	68.2±0.4
	C4.5	63.6±0.6	63.9±0.6	64.4±0.5	67.3±0.4	67.5±0.3	67.8±0.3
	NaiveBayes	63.6±0.6	63.9±0.6	64.3±0.5	67.6±0.4	68.0±0.4	68.8±0.3∘
3	GAssist	44.9±0.5	45.1±0.5	45.6±0.4	48.8±0.4	49.0±0.4	49.3±0.4
	C4.5	44.9±0.5	45.1±0.5	45.8±0.4	48.8±0.3	48.7±0.3	49.1±0.3
	NaiveBayes	44.7±0.5	45.2±0.5	45.7±0.4	49.0±0.4	49.6±0.5∘	50.7±0.3∘
5	GAssist	29.0±0.3	29.6±0.5	30.1±0.5	32.2±0.3	32.5±0.3	32.7±0.4
	C4.5	29.0±0.3	29.7±0.4	30.4±0.5	31.9±0.4	31.4±0.4•	31.0±0.5•
	NaiveBayes	29.0±0.3	29.7±0.4	30.1±0.5	33.0±0.2∘	33.9±0.3∘	34.7±0.4∘

algorithms. For each state, as the window size is increased, the accuracy increases by around 0.5%, with the exception of the C4.5 algorithm which shows a decrease in accuracy with increasing window size in five state predictions. The LCS outperformed C4.5 two times and was outperformed by Naive Bayes six times.

Most interestingly, moving from HP sequence representation to full residue type sequence information only results in a 3.8% accuracy increase for two states (64.4% vs 68.2%), 3.3% for three states (45.6% vs 49.3%) and 2.6% for the five states class definition (30.1% vs 32.7%).

Brief estimation of Information Loss In order to understand the effect of using a lower-dimensionality profile of a protein chain such as the HP model, we have computed some simple statistics on the datasets. Two measures are computed:

$$\text{redundancy} = 1 - \frac{\#\text{unique instances}}{\#\text{total instances}} \tag{4}$$

$$\text{inconsistency} = \frac{\left(\frac{\#\text{unique instances}}{\#\text{unique antecedents}}\right) - 1}{\#states - 1} \tag{5}$$

The redundancy metric in equation 4 illustrates the effect of reducing the alphabet and the window size: creating many copies of the same instances. The inconsistency metric in equation 5 shows how this reduction creates inconsistent instances: instances with equal input attributes (antecedent) but different class. For the sake of clarity this measure has been normalized for the different number of target states. Table 7 shows these ratios. For two-states and window size of one, the Real-HP dataset shows the most extreme case: any possible antecedent appears in the data set associated to both classes. Fortunately, the proportions

of the two classes for each antecedent are different, and the system can still learn. We see how the Real-HP dataset is highly redundant and how the Real-AA dataset of window size two and three presents very low redundancy and inconsistency rate. This shows both why the window size has to be large enough and also why we have to use a rich enough primary sequence representation.

Table 7. Redundancy and inconsistency rate of the tested real-proteins datasets

		HP representation		AA representation	
States	**Window Size**	**Redundancy**	**Inconsistency**	**Redundancy**	**Inconsistency**
2	1	99.99%	100.000%	93.69%	90.02%
	2	99.94%	92.50%	6.14%	3.85%
	3	99.75%	81.71%	0.21%	0.05%
3	1	99.98%	96.88%	90.90%	87.01%
	2	99.92%	86.25%	4.50%	2.84%
	3	99.66%	76.00%	0.17%	0.04%
5	1	99.97%	93.75%	85.84%	81.52%
	2	99.86%	86.25%	2.97%	1.84%
	3	99.46%	74.36%	0.14%	0.03%

5.2 Experimental battery II

This second part of the experiments of the chapter is also divided itself in two parts. The first explains the results of the experiments defined in section 4. In the second part we perform an interpretability analysis of the results obtained by GAssist on these datasets. Some discussion follows.

Results of the learning experiments We tested the selected learning systems on the 36 datasets, as summarized in table 8. Each value is the protein-wise accuracy metric defined in previous section and averaged over the ten test sets. The t-tests applied to these results are summarized in table 9, 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% and the bonferroni correction.

From the tested sets of input attributes, we can say that all the different kind of attributes sources contribute to increasing the predictive accuracy of the tested systems, as all CN2-CN6 datasets obtain higher performance than CN1. 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.

The two ways of adding global protein information to the instances, by either explicitly adding global protein descriptors or by addind the predicted protein average coordination number manage to obtain similar performance levels, by looking at the differences between CN3-CN5 and CN4-CN6. 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.

Table 8. 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 states	3 states	5 states	2 states	3 states	5 states
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 9. Number of times a method significantly outperforms another in the battery II of experiments, according to t-tests with 95% confidence level and Bonferroni correction for multiple comparisons

	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 reason of this is two-fold. First of all, GAssist has to explore a larger search space. Also, 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.

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, it might happen that the class frontiers separate examples that are practically equal. Nevertheless, it could be worth to study 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.

Looking at the specific results of each learning method, we can observe that both GAssist and C4.5 obtain their highest accuracy in the CN6 dataset, Naive Bayes 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.

Table 10. 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 states	3 states	5 states	2 states	3 states	5 states
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

Interpretability and explanatory power of GAssist results Table 10 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, that is, attributes that GAssist considered to be relevant for that rule. 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 were expressed in the CN4 - 2 classes - UL dataset (out of 42 attributes). In comparison, C4.5 (using pruning) sometimes generated solutions with as many as 8000 leaves, and LIBSVM 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, where $AA_{\pm n}$ 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 and the symbol X is used to indicate the end of chain, for the case when the window overlaps with the beginning or the end of the protein chain:

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

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 subset of the amino acids. When the number of AA types that the predicate for a certain residue can take includes more than than ten letters, that is, half of the alphabet, 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 11 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 it is quite a good metric for the generality degree of the predicates associated to 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 (such as a correlation with the cycle of an alpha helix) or if it is just the effect of a GA positional bias [45].

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 [46]. Hydrophobic residues are

Table 11. 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

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. This also matches with all the observations we did in battery I of experiments of this chapter.

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. These types of AA usually appear only 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 feature (high CN) that is almost exclusive of residues placed in the core of a protein.

These two observations illustrate how easy is to interpret the solutions generated by GAssist, in oppositions to decision trees of 8000 leaves, such as the ones that C4.5 can produce on these datasets, or the almost inexistent explanatory power of LIBSVM or Naive Bayes. GAssist can provide added value to the PSP experts, as not only has pretty good data mining capacity, but also it can do knowledge discovery via the explanatory power of the solutions it produces.

Table 12 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) are 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. Also, all of these latter residues are hydrophobic.

Table 13 analyzes these rules from a slightly different point of view: ranking the AA types for their frequency of appearance in each window position. Previously we observed that the predicate for the central residue of all rules takes a different form compared to the rest of predicates. This issue is reflected perfectly by this ranking. 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

Table 12. 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

than 95% until position 15th 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 window positions, and relating the predicates generated by GAssist with physical/chemical properties.

6 Discussion

In this section we will discuss the results presented in the previous section. As the rest of the chapter, this discussion is divided in two main parts, corresponding to the two reported batteries of experiments. Moreover, we will also briefly describe some other research work we have done which is related to this chapter.

6.1 Battery of experiments I

The LCS and other machine learning algorithms preformed at similar levels for these CN prediction tasks. Generally, increasing the number of states leads to a reduction in prediction accuracy. Reduction of input information from full residue type to HP sequence reduces the accuracy of prediction. The algorithms were, however, all capable of predictions using HP sequence that were within 4% of the accuracies obtained using full residue type sequences, considering that the size of the representation is ten times smaller (20-letter alphabet vs. 2-letter alphabet).

For all of the algorithms studied, in the case of the most informative five state predictions, moving from HP lattice to real protein HP sequences leads to a reduction of CN prediction accuracy from levels of around 50% to levels of around 30%. The significant reduction in the spatial degrees of freedom in the Lattice-HP models leads to an improvement in prediction accuracy of around 20%.

Table 13. Ranking of appearance of the AA type by window position in the generated rules for the CN 1 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

In contrast, moving from the real protein HP sequences to real protein full residue type sequences (for the same five state CN predictions) only a 3-5% improvement in prediction accuracy results from inclusion of this additional residue type information. This observation matches the general agreement in the com-

putation biology community that hydrophobicity is one of the main properties that guides the folding process of proteins and, thus, it is a key determinant of good CN prediction, and also that algorithmic studies of HP models are relevant.

The rules that result from a reduced two letter alphabet are simpler and easier to understand than those from the full residue type studies. For example, for the HP representation a rule set giving 62.9% accuracy is shown below (an X symbol is used to represent positions where the sliding window overlaps with the end of the chain).

1. If $AA_{-1} \notin \{x\}$ and $AA \in \{h\}$ and $AA_1 \in \{p\}$ then class is 1
2. If $AA_{-1} \in \{h\}$ and $AA \in \{h\}$ and $AA_1 \notin \{x\}$ then class is 1
3. If $AA_{-1} \in \{p\}$ and $AA \in \{h\}$ and $AA_1 \in \{h\}$ then class is 1
4. Default class is 0

In these rules, a class assignment of high is represented by 1 and low by 0. For the full residue type representation a rule set giving 67.7% accuracy is:

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

6.2 Battery of experiments II

The discussion for this battery of experiments is centered 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 (axis-parallel). GAssist is better than C4.5 at exploring the search space of the solutions that this kind of knowledge representation can offer. On the other hand, LIBSVM managed to outperform GAssist 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, analyzed in the results section, is the explanatory power of the solutions that GAssist generates. 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, as we have shown, 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. The only complexity measure that LIBSVM provides about its solutions is the

number of instances selected as support vectors, and this number can be huge, around 70-90% of the training set.

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 (mainly because of the high number of support vectors as stated above), 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 on-line web-based 3D protein structure prediction server, which integrates the coordination number predictors and also other related PSP datasets such as secondary structure prediction, solvent accessibility and disulfide bonding. Such a server would be queried simultaneously by multiple users that would normally want to predict tens, if not hundreds, of protein structural features with as few time delays as possible. Our experience with the 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.

6.3 Brief description of other related work

The work presented in this chapter is just a part of an outgoing line of research funded by the UK Engineering and Physical Sciences Research Council. We have continued our efforts in this research line and in this subsection we would like to briefly summarize some of the most closely related of them to the experiments described in this chapter.

The first of them [47] studies an automated method to produce alphabet reduction of the primary sequence. As we have seen in this chapter, there is a certain performance gap (less than 4%) between the accuracy obtained using the HP representation and the accuracy obtained using the AA representation. Can this gap be significantly reduced using a slightly higher alphabet size than two? We used an automated information theory-based evolutionary computation method to find the proper alphabet reduction policy, and tailor it specifically for the PSP feature being predicted. The method uses the Extended Compact Genetic Algorithm (ECGA) [48] using the Mutual Information metric [49] as fitness function. Afterwards, the produced alphabet reduction was verified by learning the dataset with reduced alphabet using BioHEL [50], a recent Learning Classifier System using the Iterative Rule Learning paradigm [51] combined with several of the features of GAssist such as the MDL fitness function, the explicit default rule mechanism and the ILAS windowing scheme. Our experiments determined that we can produce reduced alphabets applied to the Coordination Number prediction dataset with only three letters that can obtain an accuracy which is only 0.6% lower than the accuracy obtained by the AA representation.

Therefore, it represents substantial progres when compared to the standard reduced alphabets used in the literature such as the HP alphabet.

Another related work [52] studies alternative definitions of the coordination number metric based on alternative ways of defining the neighbourhood (the residues in contact with) of a certain residue. These neighbourhood definitions are based on graph theory, specifically on Proximity Graphs [53], such as the Delaunay Tessellation and the Minimum Spanning Tree, among others. Four neighbourhood definitions that produced four alternative coordination number metrics were tested, using all the protocol of class partitions criteria and the six sets of input data described in the second battery of experiments of this chapter. The evaluation process identified which measures are easier to predict than others. The explanatory power of the produced rules was also analyzed.

7 Conclusions and further work

In this chapter we have described our recent experiments with Learning Classifier Systems applied to a Bioinformatics problem called Coordination Number prediction. This problem belongs to the family of problems derived from Protein Structure Prediction. The associated datasets are a challenge to LCS for many reasons: (a) very large datasets with at least hundreds of thousands of instances, (b) noisy data and (c) challenging feature selection due to the incomplete expert understanding of these domains.

We applied a Pittsburgh approach LCS called GAssist to several variants of this problem and compared its performance to some standard machine learning methods. In general GAssist showed good performance, only being outperformed by Support Vector Machines in some of the datasets. Nevertheless, GAssist showed a competitive advantage against SVM in some aspects, especially: run-time in an exploitation environment and explanatory power. It is very difficult to understand the rationale behind SVM predictions. On the other hand, GAssist produces very small and compact rule sets for all datasets. These rule sets were easy to interpret by the domain experts and the explanations behind the predictions were acknowledged to be sound.

We have assessed two classes of input information about this domain, how to represent the primary sequence of the protein and what information can we add to improve the CN prediction, as well as assessing several definitions of classes for the coordination number prediction domain. The performance difference between the HP and the AA representations is significant but not huge. In future work we would like to investigate if we can find other kinds of reduced alphabets where this performance difference becomes minimal. We have already started to obtain successful results in this area, as briefly summarized in the discussion section. We would also like to evaluate other kinds of input information for CN prediction not tested yet, such as position specific scoring matrices or relative solvent accessibility, as well as assessing what is the real quantity of information that each of the tested class definitions is able to provide, 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. Finally, we would like to investigate how can we improve the learning process of GAssist, in order to improve its performance and scalability. We have also started to obtain better results in this scope with the development of BioHEL [50], which has better scalability capacity than GAssist, and manages to obtain similar performance to LIBSVM in many datasets. Other interesting alternatives could be hyper-ellipsoidal conditions [54], neural network conditions [55] or some kind of SVM-LCS hybrid.

8 Acknowledgments

We acknowledge the support of the UK Engineering and Physical Sciences Research Council (EPSRC) under grants GR/T07534/01, EP/D061571/1 and GR/S64530/01 and the Biotechnology and Biological Sciences Research Council (BBSRC) under grant BB/C511764/1. We are grateful for the use of the University of Nottingham's High Performance Computer.

References

1. Holland, J.H., Reitman, J.S.: Cognitive systems based on adaptive algorithms. In Hayes-Roth, D., Waterman, F., eds.: Pattern-directed Inference Systems. Academic Press, New York (1978) 313–329
2. Smith, S.: A Learning System Based on Genetic Algorithms. PhD thesis, University of Pittsburgh (1980)
3. Bernadó, E., Llorà, X., Garrell, J.M.: XCS and GALE: a comparative study of two learning classifier systems with six other learning algorithms on classification tasks. In: Fourth International Workshop on Learning Classifier Systems - IWLCS-2001. (2001) 337–341
4. Bacardit, J., Butz, M.V.: Data mining in learning classifier systems: Comparing xcs with gassist. In: Advances at the frontier of Learning Classifier Systems. Springer-Verlag (2007) 282–290
5. Wilson, S.W.: Classifier fitness based on accuracy. Evolutionary Computation **3** (1995) 149–175
6. Llorà, X., Garrell, J.M.: Knowledge-independent data mining with fine-grained parallel evolutionary algorithms. In: Proceedings of the Third Genetic and Evolutionary Computation Conference, Morgan Kaufmann (2001) 461–468
7. Bacardit, J.: Pittsburgh Genetics-Based Machine Learning in the Data Mining era: Representations, generalization, and run-time. PhD thesis, Ramon Llull University, Barcelona, Catalonia, Spain (2004)
8. Stout, M., Bacardit, J., Hirst, J.D., Krasnogor, N., Blazewicz, J.: From HP lattice models to real proteins: Coordination number prediction using learning classifier systems. In: Applications of Evolutionary Computing, EvoWorkshops 2006, Springer LNCS 3907 (2006) 208–220

9. Bacardit, J., Stout, M., Krasnogor, N., Hirst, J.D., Blazewicz, J.: Coordination number prediction using learning classifier systems: performance and interpretability. In: GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation, ACM Press (2006) 247–254
10. Stout, M., Bacardit, J., Hirst, J.D., Krasnogor, N.: Prediction of residue exposure and contact number for simplified hp lattice model proteins using learning classifier systems. In a. Ruan, D., D'hondt, P., Fantoni, P.F., Cock, M.D., Nachtegael, M., Kerre, E.E., eds.: Proceedings of the 7th International FLINS Conference on Applied Artificial Intelligence, Genova, Italy, World Scientific (2006) 601–608
11. Hinds, D.A., Levitt, M.: A lattice model for protein-structure prediction at low resolution. Proc. National Academy Sciences U.S.A. **89** (1992) 2536–2540
12. Yue, K., Fiebig, K.M., Thomas, P.D., Sun, C.H., Shakhnovich, E.I., Dill, K.A.: A test of lattice protein folding algorithms. Proc. Natl. Acad. Sci. USA **92** (1995) 325–329
13. Kinjo, A.R., Horimoto, K., Nishikawa, K.: Predicting absolute contact numbers of native protein structure from amino acid sequence. Proteins **58** (2005) 158–165
14. Baldi, P., Pollastri, G.: The principled design of large-scale recursive neural network architectures dag-rnns and the protein structure prediction problem. Journal of Machine Learning Research **4** (2003) 575 – 602
15. Shao, Y., Bystroff, C.: Predicting interresidue contacts using templates and pathways. Proteins **53** (2003) 497–502
16. MacCallum, R.: Striped sheets and protein contact prediction. Bioinformatics **20** (2004) I224–I231
17. Zhao, Y., Karypis, G.: Prediction of contact maps using support vector machines. In: Proceedings of the IEEE Symposium on BioInformatics and BioEngineering. (2003) 26–36
18. Altschul, S.F., Madden, T.L., Scher, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped blast and psi-blast: a new generation of protein database search programs. Nucleic Acids Res **25** (1997) 3389–3402
19. Abe, H., Go, N.: Noninteracting local-structure model of folding and unfolding transition in globular proteins. ii. application to two-dimensional lattice proteins. Biopolymers **20** (1981) 1013–1031
20. Hart, W.E., Istrail, S.: Crystallographical universal approximability: A complexity theory of protein folding algorithms on crystal lattices. Technical Report SAND95-1294, Sandia National Labs, Albuquerque, NM (1995)
21. Hart, W., Istrail, S.: Robust proofs of NP-hardness for protein folding: General lattices and energy potentials. Journal of Computational Biology (1997) 1–20
22. Escuela, G., Ochoa, G., Krasnogor, N.: Evolving l-systems to capture protein structure native conformations. In: Proceedings of the 8th European Conference on Genetic Programming (EuroGP 2005), Lecture Notes in Computer Sciences 3447, pp 73-84, Springer-Verlag, Berlin (2005)
23. Krasnogor, N., Pelta, D.: Fuzzy memes in multimeme algorithms: a fuzzy-evolutionary hybrid. In Verdegay, J., ed.: Fuzzy Sets based Heuristics for Optimization, Springer (2002)
24. Krasnogor, N., Hart, W., Smith, J., Pelta, D.: Protein structure prediction with evolutionary algorithms. In Banzhaf, W., Daida, J., Eiben, A., Garzon, M., Honavar, V., Jakaiela, M., Smith, R., eds.: GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann (1999)
25. Krasnogor, N., Blackburne, B., Burke, E., Hirst, J.: Multimeme algorithms for protein structure prediction. In: Proceedings of the Parallel Problem Solving from Nature VII. Lecture Notes in Computer Science. Volume 2439. (2002) 769–778

26. Krasnogor, N., de la Cananl, E., Pelta, D., Marcos, D., Risi, W.: Encoding and crossover mismatch in a molecular design problem. In Bentley, P., ed.: AID98: Proceedings of the Workshop on Artificial Intelligence in Design 1998. (1998)
27. Krasnogor, N., Pelta, D., Marcos, D., Risi, W.: Protein structure prediction as a complex adaptive system. In: Proceedings of Frontiers in Evolutionary Algorithms 1998. (1998)
28. DeJong, K.A., Spears, W.M., Gordon, D.F.: Using genetic algorithms for concept learning. Machine Learning **13** (1993) 161–188
29. Bacardit, J.: 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., ACM Press (2005) 1843–1850
30. Rissanen, J.: Modeling by shortest data description. Automatica **vol. 14** (1978) 465–471
31. Bacardit, J., Goldberg, D., Butz, M., Llorà, X., Garrell, J.M.: Speeding-up pittsburgh learning classifier systems: Modeling time and accuracy. In: Parallel Problem Solving from Nature - PPSN 2004, Springer-Verlag, LNCS 3242 (2004) 1021–1031
32. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
33. Bacardit, J., Krasnogor, N.: Empirical evaluation of ensemble techniques for a pittsburgh learning classifier system. In: Proceedings of the 9th International Workshop on Learning Classifier Systems, (to appear), LNAI, Springer-Verlag (2007)
34. Blake, C., Keogh, E., Merz, C.: UCI repository of machine learning databases (1998) (www.ics.uci.edu/mlearn/MLRepository.html).
35. Liu, H., Hussain, F., Tam, C.L., Dash, M.: Discretization: An enabling technique. Data Mining and Knowledge Discovery **6** (2002) 393–423
36. Noguchi, T., Matsuda, H., Akiyama, Y.: Pdb-reprdb: a database of representative protein chains from the protein data bank (pdb). Nucleic Acids Res **29** (2001) 219–220
37. Sander, C., Schneider, R.: Database of homology-derived protein structures. Proteins **9** (1991) 56–68
38. Broome, B., Hecht, M.: Nature disfavors sequences of alternating polar and nonpolar amino acids: implications for amyloidogenesis. J Mol Biol **296** (2000) 961–968
39. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
40. John, G.H., Langley, P.: Estimating continuous distributions in Bayesian classifiers. In: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, San Mateo (1995) 338–345
41. Witten, I.H., Frank, E.: Data Mining: practical machine learning tools and techniques with java implementations. Morgan Kaufmann (2000)
42. Miller, R.G.: Simultaneous Statistical Inference. Springer Verlag, New York (1981) Heidelberger, Berlin.
43. Jones, D.: Protein secondary structure prediction based on position-specific scoring matrices. J Mol Biol **292** (1999) 195–202
44. Chang, C.C., Lin, C.J.: 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`.
45. Booker, L.: Recombination distribution for genetic algorithms. In: Foundations of Genetic Algorithms 2, Morgan Kaufmann (1993) 29–44
46. Livingstone, C.D., Barton, G.J.: Protein sequence alignments: a strategy for the hierarchical analysis of residue conservation. Computer Applications in the Biosciences **9** (1993) 745–756

47. Bacardit, J., Stout, M., Hirst, J.D., Sastry, K., Llorà, X., Krasnogor, N.: Automated alphabet reduction method with evolutionary algorithms for protein structure prediction. In: GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM Press (2007) to appear
48. Harik, G.: Linkage learning via probabilistic modeling in the ecga. Technical Report 99010, Illinois Genetic Algorithms Lab, University of Illinois at Urbana-Champaign (1999)
49. Cover, T.M., Thomas, J.A.: Elements of Information Theory. John Wiley & sons (1991)
50. Bacardit, J., Krasnogor, N.: Biohel: Bioinformatics-oriented hierarchical evolutionary learning. Nottingham eprints, University of Nottingham (2006)
51. Venturini, G.: Sia: A supervised inductive algorithm with genetic search for learning attributes based concepts. In Brazdil, P.B., ed.: Machine Learning: ECML-93 - Proc. of the European Conference on Machine Learning. Springer-Verlag, Berlin, Heidelberg (1993) 280–296
52. Stout, M., Bacardit, J., Hirst, J.D., Smith, R.E., Krasnogor, N.: Prediction of topological contacts in proteins using learning classifier systems. Soft Computing (2007) Special Issue on Evolutionary and Metaheuristic–based Data Mining (EMBDM), to appear
53. Preparata, F.P.: Computational geometry : an introduction / Franco P. Preparata, Michael Ian Shamos. Texts and monographs in computer science. Springer-Verlag (1985)
54. Butz, M.V., Lanzi, P.L., Wilson, S.W.: Hyper-ellipsoidal conditions in xcs: rotation, linear approximation, and solution structure. In: GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation, New York, NY, USA, ACM Press (2006) 1457–1464
55. O'Hara, T., Bull, L.: Backpropagation in accuracy-based neural learning classifier systems. In: Advances at the frontier of Learning Classifier Systems. Springer-Verlag (2007) 25–39