Available online at www.sciencedirect.com
ScienceDirect
Journal of Biomedical Informatics 40 (2007) 203–220
Journal of Biomedical Informatics
www.elsevier.com/locate/yjbin

Applying hybrid reasoning to mine for associative features in biological data

Boris A. Galitsky [a,*], Sergey O. Kuznetsov [b], Dmitry V. Vinogradov [b]

[a] School of Computer Science and Information Systems, Birkbeck College, University of London, Malet Street, London WC1E 7HX, UK
[b] All-Russian Institute for Scientific and Technical Information (VINITI), Usievicha 20, Moscow 125190, Russia

Received 26 April 2005
Available online 22 July 2006

Abstract

We develop the means to mine for associative features in biological data. The hybrid reasoning schema for deterministic machine learning and its implementation via logic programming is presented. The methodology of mining for correlation between features is illustrated by the prediction tasks for protein secondary structure and phylogenetic profiles. The suggested methodology leads to a clearer approach to hierarchical classification of proteins and a novel way to represent evolutionary relationships. Comparative analysis of *Jasmine* and other statistical and deterministic systems (including Explanation-Based Learning and Inductive Logic Programming) are outlined. Advantages of using deterministic versus statistical data mining approaches for high-level exploration of correlation structure are analyzed.

Keywords: Deterministic machine learning; Plausible reasoning; Causal link; Protein secondary structure prediction; Phylogenetic profile analysis

1. A hybrid deterministic reasoning technique

In many scientific areas, including bioinformatics, data are being generated at a faster pace then it can be effectively analyzed. Therefore, the possibility to automate the scientific discovery process is of both theoretical and practical value. In this paper we present the system *Jasmine* which automatically generates hypotheses to explain observations, verifies these hypotheses by finding the subset of data satisfying them, falsifies some of the hypotheses by revealing inconsistencies and finally derives the explanations for the observations by means of *cause–effect* links [12].

Analyzing biological data, it is of crucial importance not just to reveal correlations between biological parameters, but also to find interpretable causal links between them. Most of the time, the machinery of statistical inference allows the former but runs into difficulties with the latter. Non-uniform, statistically distorted and incomplete biological data in bioinformatics, extracted from various sources, is frequently intractable to immediate statistical analysis which would deliver an adequate explanation of phenomena. In response to this problem, special non-statistical techniques have been developed to tackle a wide variety of problems in bioinformatics [9,10,20], and *Jasmine*, the system to be presented here, extends this approach.

A traditional characterization of scientific methodology is the "hypothetico-deductive" process. Firstly, scientific knowledge, intuition and imagination yield plausible hypotheses, and then the deductive conjectures of these hypotheses are verified experimentally [6,22]. *Jasmine* extends this methodology by implementing (i) a combination of abductive, inductive and analogical reasoning for hypotheses generation and (ii) multi-valued logic-based deductive reasoning for verification of their consistency and coverage of the domain (compare with [11,36]). Formally, all the above components can be represented as deductive inference via logic programming [1,5,26].

* Corresponding author.
E-mail addresses: galitsky@dcs.bbk.ac.uk (B.A. Galitsky), serge@viniti.ru (S.O. Kuznetsov), vin@viniti.ru (D.V. Vinogradov).
URL: http://www.dcs.bbk.ac.uk/~galitsky/Jasmine/ (B.A. Galitsky).

1532-0464/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jbi.2006.07.002

In contrast to statistical approaches, *Jasmine* represents domain knowledge in the form of an *ontology*. This provides the means both for direct inspection of biological knowledge in the model and for inference of predictions.

In keeping with most hypotheses generation strategies used in modern biology [11], *Jasmine* uses abductive reasoning in addition to its deductive and inductive capabilities. These abductive processes generate specific facts about biological entities, while general hypotheses are delivered by deductive and inductive procedures.

In this paper, results of data mining can be thought of *effects* (goal concepts) because of the structure of inter-dependencies between those features which are known and those which are subject to prediction. Making a prediction, *Jasmine* explicitly indicates which objects and which features delivered this prediction. Therefore in our prediction settings, objects and their features that deliver prediction are considered *causes* (because they are explicitly localized); the object's predicted feature can be thought of as an *effect*. Indeed, causes may not imply effects in a physical sense; however the metaphor is useful because it highlights the *explanatory* property of a prediction delivered by *Jasmine*. It is important to differentiate *Jasmine's* prediction settings, which comprise a hypotheses management environment, from pure accuracy-oriented prediction settings. The objective in the design of *Jasmine* is not simply to raise recognition accuracy, but to identify sets of features and their inter-relations in logical form, which in turn raise the recognition quality. Note also that a prediction direction (the roles of being a known feature and a goal concept) in *Jasmine* settings is easily reversed.

In terms of its deterministic methodology of prediction and implementation via logic programming, *Jasmine's* approach is close to those of Inductive Logic Programming (ILP) [20] which has been applied in management hypotheses in bioinformatics quite successfully, and Explanation-Based Learning (EBL) [39,40] class of methods which are intended to derive as general expressions as possible from available data.

EBL has been deployed in forming rules while observing a human expert, decision-making and advising systems in rather compact domains. One of the motivations for this project is to deploy the power of EBL, expressiveness of its representations and delivery of explanation (which is essential for understanding the underlying mechanisms) for more extensive and less structured domains such as bioinformatics. Therefore the desired reasoning machinery of *Jasmine* should be able to accommodate noisy and scarce date, a lack of an adequate domain theory, or its bioinformatics-specific deficiencies such as problems with completeness, correctness and tractability. To achieve this, we use more cautious prediction settings to decrease the number of false negatives and iterative application of induction–abduction procedure on the one hand, and provide an interactive environment to visualize explanations on the other hand. Furthermore, unlike the EBL approach, *Jasmine* uses negative examples to falsify hypotheses that have counter-examples. Also, defeasible logic programming [42] is integrated into *Jasmine* to provide a machinery to dynamically accept and reject hypotheses that have been assigned the status of "defeasible" at the knowledge base construction step.

The rest of the present paper is organized as follows. In Section 1 we introduce the specific problem of protein secondary structure prediction and outline the knowledge domain which will be a subject of reasoning. We then describe the reasoning schema using the reduced dataset of immunoglobulin proteins in Section 2, followed by Section 3 which describes the computation of similarity for obtaining causal links. Section 4 shows how *Jasmine* predicts the length of certain protein fragments and helps to find the physically plausible explanation for these predictions. In Section 5 *Jasmine* is applied to the problem of mining evolutionary profiles, where an original structure of causal links between occurrences of protein groups in biological species is obtained.

2. Introductory example: predicting the secondary structure of immunoglobulins

We illustrate how *Jasmine* tackles the biological problem where micro-level features lead to the macro-level goal concepts. The problem in our general terms is posed as a prediction of macro-level effects given the micro-level data. The micro-level features comprise the protein residues (amino acids in certain positions), and the macro-level is the secondary structure of a protein. The secondary structure of a protein contains information on how a sequence of amino acids is divided into fragments (strands, loops and helices in 3D space). Evidently, individual amino acids determine the secondary structure by means of "collective behavior" whose direct simulation is not computationally feasible (e.g. [47]). Prediction of secondary structure is an important step on the way of 3D structure prediction, crucial in a number of biomedical applications.

To illustrate the prediction of the secondary structure of the immunoglobulin family with *Jasmine*, we select the training dataset of sequences, where the variable length of certain loop fragments is dependent on the residues of other fragments. We form our dataset from the set of immunoglobulin sequences of variable domain, heavy chains [7]. The purpose of our analysis is to predict such secondary structure features of protein sequences as *length* (number of residues) of the certain loops (we select CC′ and CB). It is expected that various residues in protein sequences affect these lengths, and we will use *Jasmine* to help us discover such residues and the way they can determine the lengths of CC′ and CB loops.

For example, 19th position of the alignment is the 3rd position of B-keyword. B3 = R or K for the first B-keyword, and B3 = S or T for the second B-keyword. The whole B-keywords are as follows: S [L or V] [R or K] [L or V or I] [S or T] C X [G or A] and T [L or V] [S or T] [L or V or I] [S or T] C X [I or V].

Before we apply the deterministic machine learning, the hierarchical clustering approach is used to decrease the data dimension. To find inter-connections between residues in a sequence of 100 amino acids long, we divide the multiple sequence alignment into 21 fragments to search for correlation among these fragments rather than on the level of residues. After that, we divide the fragments of sequences into clusters, and use *Jasmine* to find correlation between these clusters. *Jasmine* is capable of operating with raw data; however, it would be much harder to take advantage of *Jasmine*'s results if it is fed with original data containing 1000 aligned sequences of about 100 amino acids.

We proceed to the illustration of our data pre-processing for *Jasmine*. We start with the aligned protein sequences:

0A	A	AA′	A′	A′B	B
QVQ	LV**E**	SGA	ELKK	PSG	SLRLSCHG ...
EVQ	LV**Q**	SGG	ELKK	PSG	SVRLSCJA ...

To ascend from the level of individual amino acids to the level of amino acid patterns, we divide sequences in fragments (0A A AA′ A′...) following the secondary structure of proteins. Then a clustering is performed in each fragment, taking into account the selected matrix of similarity between amino acids. All clusters for each fragment (0A, A, AA′, A...) are presented in Table 1. Now we can represent sequences by means of cluster numbers:

1	1	1	1	1	1	...
2	2	1	1	1	1	...

Notice that sequence fragments which deviate by amino acids of the same group (e.g. G-A, L-V) belong to the same cluster (B-fragments **SLRLSCHG** and **SVRLSCJA**, the seventh position is any residue X). Having performed the clustering, one observes the correlation between residues, for example, first and sixth in the sequence, 0A1 and A3 (shown in bold):

0A1 = 'Q' ↔ A3 = 'E'

0A1 = 'E' ↔ A3 = 'Q'

Table 1 presents typical representatives for each cluster (keywords) and encodes them by numbers. Table 1 indicates that the sequences within each fragment fall into one to six clusters [7,8]. Now we can attempt to predict secondary structure given cluster numbers instead of original sequence residues. Having significantly reduced the feature

Table 1
The keywords (typical sequence fragments, cluster representatives)

#	1	2	3	#	4	5	6	#	7	8	9	#	10	11	12	13	#	14	15	16	#	17	18	19	20	21	22	23	24
	0A1	0A2	0A3		A1	A2	A3		AA'1	AA'2	AA'3		A'1	A'2	A'3	A'4		A'B1	A'B2	A'B3		B1	B2	B3	B4	B5	B6	B7	B8
1	Q	V	Q	1	L	VL	E	1	S	G	GA	1	E	VL	K	K	1	P	GS	GAS	1	S	LV	RK	LVI	ST	C	X	GA
2	E			2		V	Q	2	S		P	2	G		V	K	2			E	2	T		ST				X	IV
				3		Q	E	3	W		A	3	G		V	Q	3			Q									
				4		Q	Q										4			R									

#	25	26	27	28	29-33	#	34	35	36	37	38	39	#	40	41	42	43	44	45	#	46	47	48	49	50	51	52-55	56-60
	BC1	BC2	BC3	BC4	..CB..		C1	C2	C3	C4	C5	C6		CC'1	CC'2	CC'3	CC'4	CC'5	CC'6		C'1	C'2	C'3	C'4	C'5	C'6	..C'C"	..C"..
1			FY	X		1	MI	X	W	VI	R	Q	1	X	P	GS	RK	G	L	1	E	W	VMIL	GAS	X	I		
2	S	G	G	ST		2	W	X					2	A			Q			2					X	T		
3			D	S																								

#	61	62	63	64	65	#	66	67	68	69	70	71	72	#	73	74	75	76	#	77	78	79	80	81	82	82A
	C"D1	C"D2	C"D3	C"D4	C"D5		D1	D2	D3	D4	D5	D6	D7		DE1	DE2	DE3	DE4		E1	E2	E3	E4	E5	E6	E7
1	X	S	VL	K		1	R					R		1	X		K	N	1	TS	L	Y		Q	M	N
2	Q	X	FL	Q	G	2	R	FVI	T	IM	ST	VA	D	2	X	SA	I	S	2	T	A	Y		Q	W	S
3	P	S	F	Q		3	Q					A		3	X		T	S	3	Q	F	S	LM	K	L	S
4	X	P	V	K		4	R					P							4	T	A	Y		E	L	X
																			5	Q	F	S		Q	L	N
																			6	Q	V	V		T	M	T

#	82B	82C	83	84	85	86	87	#	88	89	90	91	92	93	94	95-101	#	102	103	104	105	106	107	108	109	110	111
	EF1	EF2	EF3	EF4	EF5	EF6	EF7		F1	F2	F3	F4	F5	F6	F7	..FG..		G1	G2	G3	G4	G5	G6	G7	G8	G9	G10
1	S		R	X	E			1	A	VML	Y	YF	C	A	RK		1	X	W	G	Q	G	T	LM	V	T	V
2	S		T	A	A			2						T	X		2	X						T			
3	S	LVM	K	A	S	D	T																				
4	S		T	P	E																						
5	N		D	P	V																						

We divide the aligned sequences into 21 fragments 0A, A, AA′, A′, ... in accordance to conventional names for strands and loops [34]. For each position, we show its number within alignment and within corresponding fragment (fragments C′C″, C″, CB, FG with high variability are omitted). For each fragment, we show the keywords [7]. Gray color denotes that all keywords contain the same residues in the highlighted position. If two or three residues are shown for a position, each of them can be found in this position. 'X' indicates that the position is highly variable, and is not used to form clusters. Note, that in all cases the residues sharing a position belong to the same amino acid group.

space, we can expect to reveal rules and data correlation which are *interpretable* by a human expert.

A convenient way to represent the totality of sequence in the dataset is shown in Fig. 1, where the sequences are encoded in cluster numbers. The two-level classification were suggested in [8] to show the diversity of sequences of cluster numbers as a classification tree. Each node of a tree is a unique combination of cluster numbers; fragment E is chosen as the first-level classification criterion, and fragments {0A, A, A′, F} as the second level criterion. In accordance to the selected classification criteria, for each class, the selected classifying fragments determine not only the rest of fragments, but the number of residues in the loop fragments (in the way presented in Fig. 1). In such form, the data is well-prepared for applying *Jasmine*.

For example, the above two sequences belong to two paths of the classification tree. The first one belongs to the second class along with its only subclass (amino acids are subscribed at the classification tree). The second sequence belongs to the fourth class and its first subclass. It turns out that both these sequences have the same lengths of $CC' = 5$ and $CB = 5$.

Hence we formed the reduced feature space for such objects as protein sequences and specified the goal concept as the lengths of two loop protein regions, CC′ and CB. This is the environment to introduce the reasoning procedure that will discover the links between the features of objects and the goal concept. In the following sections, we will see whether these links obtained manually in the form of classification tree, can be treated in a more systematic way by means of automated reasoning.

3. A reasoning schema

Jasmine is based on a learning model called JSM-method [5] (in honor of John Stuart Mill, the English philosopher who proposed schemes of inductive reasoning in the 19th century). JSM-method to be presented in this section implements Mill's idea [16] that similar effects (associated features, goal concepts) are likely to follow common causes (attributes).

In this paper, rather than presenting JSM in relation to Inductive Logic Programming class of approaches, we use the Explanation-based Learning (EBL) framework to introduce our reasoning schema. Like EBL, JSM attempts to solve the problem of *inductive bias* [37], a means to select one generalization over another.

Given the *features* of *objects*, we intend to obtain *an expression for the goal concept* that includes all positive examples and excludes all negative examples, given some initial formalized background knowledge. In the EBL setting (justified generalization [39]) the expression for the goal concept is a logical consequence of background knowledge and training dataset; however, this condition is not always viable in a domain with experimental observations. EBL is designed to generalize form a single example; however, in biological domains one would prefer more reliable conclusions from multiple observations. These multiple observations (examples) may introduce inconsistencies; and the desired machine learning technique should be capable of finding consistent explanations linking possibly mutually inconsistent observations with the goal concept.

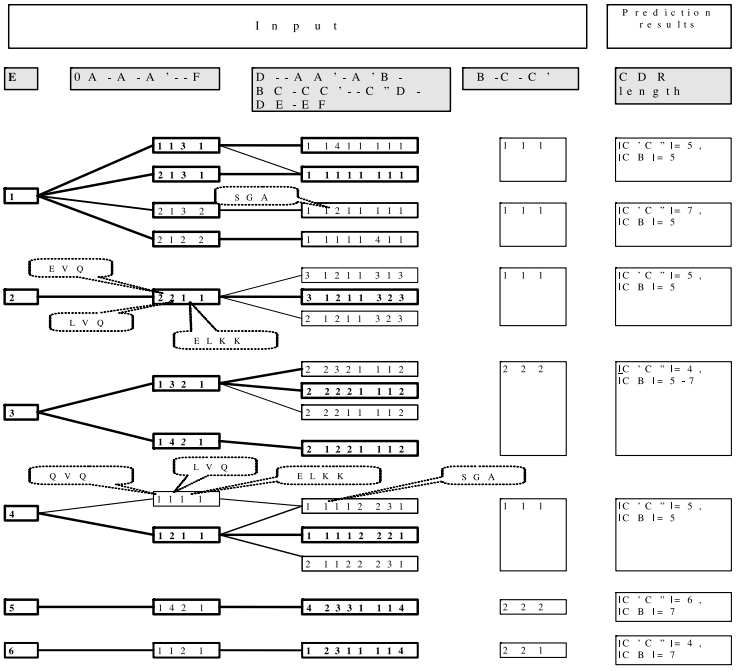

Fig. 1. The classification graph for immunoglobulin sequences, encoded via cluster numbers. Fragment E is the criterion for division into classes; fragments 0A–A –A′–F are the criteria to divide into subclasses. Each subclass has a specific length of the complimentary determining region (CDR, [8,23]) fragments which are the subjects of prediction. Exploration of the links between fragments for various prediction settings leads to the discovery of a novel interconnection pattern between the fragments with physical plausibility: fragments E, 0A,A, A′ and F have a special role in protein folding process [8].

Like EBL, the *Jasmine* environment includes the training examples (objects), operational criterion (a description of which concepts are usable), a domain theory (a set of rules that describe relationships between objects), and a goal concept which is the subject of prediction. Within *Jasmine* first-order language, objects are atoms, whereas known features and the goal concept are the terms which include these atoms. For a goal concept, there are four groups of objects with respect to the evidence they provide for this goal concept:

Positive–Negative–Inconsistent–Unknown.

An inference to obtain this goal concept (satisfied or not) can be represented as one in a respective four-valued logic [1]. The predictive machinery is based on building hypotheses,

$$goal\_concept(O) : - \ feature_1(O, \ldots), \ldots, feature_n(O, \ldots),$$

that separate examples,

where *goal_concept(O)* is to be predicted, and $features_1$, …, $feature_n \in features$ are the features the goal concept is associated with; O ranges over objects.

Desired separation is based on the *similarity* of objects in terms of features they satisfy. Usually, such similarity is domain-dependent. However, building the general framework of inductive-based prediction, we use the anti-unification of formulas that express the totality of features of the given and other objects (our features (causes) do not have to be unary predicates; they are expressed by arbitrary first-order terms).

Formal Concept Analysis [3,30], where similarity is the *meet* operation of a *lattice* (called concept lattice). For *Jasmine* we choose the anti-unification of formulas which expresses features of the pair of objects to derive a formula for similarity sub-object. Below we will be using the predicate

similar(Object1, Object2, CommonSubObject) which yields the third argument given the first and the second arguments.

We start with an abstract example of *Jasmine* setting, based on the secondary structure prediction of immunoglobulin proteins from Section 1 above. In the section below we will comment on the meanings of involved features and goal concepts. Our introductory example of JSM settings for unary predicate is as follows (from now on we use the conventional PROLOG notations for variables and constants), Fig. 2.

We start our presentation of reasoning procedure with the chart (Fig. 3), followed by the logic program representation. Let us build a framework for predicting the goal concept V of objects set by the formulas X expressing their features: *unknown(X, V)*. We are going to predict whether $V(x_1, \ldots x_n)$ holds or not, where $x_1, \ldots x_n$ are variables of the formula set X (in our example, $X = \texttt{cb5(o10)}$, $x_1 = \texttt{o10}$).

We start with the raw data, positive and negative examples, *raw Pos(X, V)* and *rawNeg(X, V)*, for the goal concept V, where X range over formulas expressing features of objects. We form the totality of intersections for these examples (positive ones, U, that satisfy *iPos(U, V)*, and negative ones, W, that satisfy *iNeg*(W, V), not shown):

$$\begin{aligned} &iPos(U,V) : - \ rawPos(X1,V), rawPos(X2,V), X1 \setminus = X2, similar(X1,X2,U), U \setminus = []. \\ &iPos(U,V) : - \ iPos(U1,V), rawPos(X1,V), similar(X1,U1,U), U \setminus = []. \end{aligned} \quad (1)$$

JSM-prediction is based on the notion of similarity between objects. Similarity between a pair of objects is a hypothetical object which obeys the common features of this pair of objects. In handling similarity JSM is close to

Above are the recursive definitions of the intersections. As the logic program clauses which actually construct the lattice for the totality of intersections for positive and negative examples, we introduce the third argument to

```
features([e1, e2, e3, e4, e5, e6, oa1, oa2, ap1, ap2, ap3, ap4, f1, f2,
   cc4, cc5, cc6, cc7, cb5, cb7]). %% Features and goal concepts
objects([o1, o2, o3, o4, o5, o6, o7, o8, o9, o10, o11, o12, o13, o14,
   o15, o16, o17, o18]).
goal_concept [cb5]).
    %% Beginning of knowledge base
e1(o1). oa1(o1). ap1(o1). ap3(o1). f1(o1).  cc5(o1). cb5(o1).
e1(o2). oa1(o2). ap1(o2). ap3(o2). f1(o2).  cc5(o2). cb5(o2).
e2(o8). oa2(o8). ap2(o8). ap1(o8). f1(o8).  cc5(o8). cb5(o8).
e3(o10). oa1(o10). a3(o10). ap2(o10). f1(o10). cc4(o10).
e3(o11). oa1(o11). a3(o11). ap2(o11). f1(o11). cc4(o11). cb5(o11).
cb7(o11).
e4(o16). oa1(o16). a1(o16). ap1(o16). f1(o16). cc5(o16). cb5(o16).
e5(o17). oa1(o17). a4(o17). ap2(o17). f1(o17). cc6(o17). cb7(o17).
e6(o18). oa1(o18). a1(o18). ap2(o18). f1(o18). cc4(o18). cb7(o18).
    %% End of knowledge base
unknown(cb5(o10)).
```

Fig. 2. A sample knowledge base for high-level mining of protein sequence data.

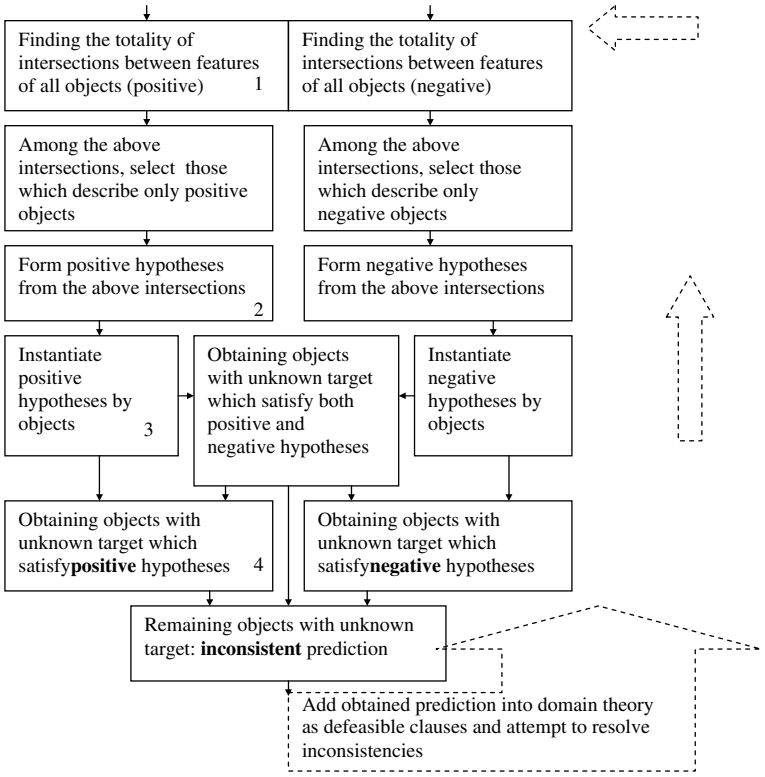

Fig. 3. The chart for reasoning procedure of *Jasmine*. Note that the prediction schema is oriented to discover which features cause the goal concept and how (the causal link) rather than just searching for common features for the goal concept (which would be much simpler, six units on the top). The respective clauses (1–4) and sample results for each numbered unit (1–4) are presented in Fig. 4.

accumulate the currently obtained intersections (the negative case is analogous):

$$
\begin{aligned}
&iPos(U,V) \text{ :- } iPos(U,V,\_).\\
&iPos(U,V,Accums) \text{ :- } rawPos(X1,V), rawPos(X2,V), X1 \backslash = X2, similar(X1,X2,U),\\
&\qquad Accums = [X1,X2], U \backslash = [].\\
&iPos(U,V,AccumsX1) \text{ :- } iPos(U1,V,Accums), !, rawPos(X1,V),\\
&\qquad not\ member(X1,Accums), similar(X1,U1,U), U \backslash = [],\\
&\qquad append(Accums,[X1],AccumsX1).
\end{aligned}
$$

As one can see, there is a "symmetric" treatment of positive and negative examples and hypotheses: *Jasmine* uses negative examples to falsify hypotheses that have counter-examples. On the contrary, a simplified EBL uses only positive examples and can be viewed as just the "left half" of Fig. 3.

To obtain the actual positive *posHyp* and negative *negHyp* hypotheses from the intersections derived above, we filter out the inconsistent hypotheses which belong to both positive and negative intersections *inconsHyp (U, V)*:

$$
\begin{aligned}
&inconsHyp(U,V) \text{ :- } iPos(U,V), iNeg(U,V).\\
&posHyp(U,V) \text{ :- } iPos(U,V), not\ inconsHyp(U,V).\\
&negHyp(U,V) \text{ :- } iNeg(U,V), not\ inconsHyp(U,V). \qquad (2)
\end{aligned}
$$

Here U is the formula expressing the features of objects. It serves as a body of clauses for hypotheses V :- U.

The following clauses deliver the totality of objects so that the features expressed by the hypotheses are *included* in the features of these objects. We derive positive and negative hypotheses *reprObjectsPos(X, V)* and *reprObjectsNeg(X, V)* where X is instantiated with objects (V is positive and negative respectively). The last clause (with the head *reprObjectsIncons(X, V)*) implements the search for the objects to be predicted so that the features expressed by both the positive and negative hypotheses are included in the features of these objects.

Two clauses above (top and middle) do not participate in prediction directly; their role is to indicate which objects deliver what kind of prediction. This form of analysis will be illustrated in Section 5, Fig. 6 (*Jasmine* user interface).

$$\begin{aligned}
&reprObjectsPos(X,V) :\text{-}\ rawPos(X,V), posHyp(U,V), similar(X,U,U).\\
&reprObjectsNeg(X,V) :\text{-}\ rawNeg(X,V), negHyp(U,V), similar(X,U,U).\\
&reprObjectsIncons(X,V) :\text{-}\ unknown(X,V), posHyp(U1,V), negHyp(U2,V),\\
&\qquad similar(X,U1,U1), similar(X,U2,U2).
\end{aligned} \tag{3}$$

Finally, we approach the clauses for prediction. For the objects with unknown goal concepts, the system predicts that they either satisfy these goal concepts, do not satisfy these goal concepts, or that the fact of satisfaction is inconsistent with the raw facts. To deliver V, a positive hypothesis has to be found so that the set of features X of an object has to include the features expressed by this hypothesis, and X should not be from *reprObjectsIncons(X, V)*. To deliver $\neg V$, a negative hypothesis has to be found so that a set of features X of an object has to include the features expressed by this hypothesis and X is not from *reprObjectsIncons(X, V)*. No prediction can be made for the objects with features expressed by X from the third clause, The first clause above (shown in bold) will serve as an entry point to predict (choose) a given goal concept among a generated list of possible goal concepts that can be obtained for the current state. The clause below is an entry point to *Jasmine* if it is integrated with other applications and/or reasoning components.

Predicate *loadRequiredSamples(As)* above forms the training dataset. If for a given dataset a prediction is inconsistent, it is worth eliminating the objects from the dataset which deliver this inconsistency. Conversely, if there are an insufficient number of positive or negative objects, additional ones are included in the dataset. A number of iterations may be required to obtain a prediction, however the iteration procedure is monotonic and deterministic: the source of inconsistency/insufficient data cases are explicitly indicated at the step where predicates *reprObjectsPos* and *reprObjectsNeg* introduced above are satisfied. This is the solution to the so called *blame assignment* problem, where by starting at the erroneous or inconsistent conclusion and tracking backward through the explanation structure, it is possible to identify pieces of domain knowledge that might have caused an error or inconsistency [43].

When the set of obtained rules *posHyp* and *negHyp* for positive and negative examples (together with the original domain theory) is applied to a more extensive (evaluation or exploration) dataset, some of these rules may not always hold. If at the first run (1–4) *Jasmine* refuses to make predictions for some objects with unknown goal concepts, then a repetitive iteration may be required, attempting to use newly generated predictions to obtain objects' goal concepts which are currently unavailable (compare with [44]). The arrows on the right of Fig. 3 illustrate this kind of iterative process.

For example, for the knowledge base Fig. 2 above, we have the following protocol and results (Fig. 4).

Hence *cb5(o10)* holds, which means that the sequence o10 has the length of loop of five amino acids.

$$\begin{aligned}
&predictIncons(X,V).\\
&\quad \textbf{\textit{predictPos}}(\boldsymbol{X},\boldsymbol{V}) :\text{-}\ \textbf{\textit{unknown}}(\boldsymbol{X},\ \boldsymbol{V}),\ \textbf{\textit{posHyp}}(\boldsymbol{U},\ \boldsymbol{V}),\ \textbf{\textit{similar}}(\boldsymbol{X},\ \boldsymbol{U},\boldsymbol{U}),\\
&\qquad\qquad \textbf{\textit{not reprObjectsIncons}}(\boldsymbol{X},\ \boldsymbol{V}).\\
&\quad predictNeg(X,V) :\text{-}\ unknown(X,V), negHyp(U,V), similar(X,U,U),\\
&\qquad\qquad not\ reprObjectsIncons(X,V).\\
&\quad predictIncons(X,V) :\text{-}\ unknown(X,V), not\ predictPos(X,V), not\ predictNeg(X,V), not\ reprObjectsIncons(X,V).
\end{aligned} \tag{4}$$

4. Computing similarity between objects

The quality of *Jasmine*-based prediction is dramatically dependent on how the similarity of objects is defined. Usually, high prediction accuracy can be achieved if the measure of similarity is sensitive to object features which determine the goal concept (explicitly or implicitly). Since most of times it is unclear in advance which features affect the goal concept, the similarity measure should take into account all available features. If the totality of selected features describing each object is expressed by formulas, a reasonable expression of similarity between a pair of objects is the following. It is a formula which is the least common generalization of the formulas for both objects, which is anti-unification, mentioned in the previous

$$\begin{aligned}
&predict_goal_concept_by_learning(GoalConceptToBePredicted,S) :-\\
&\quad findAllPossibleGoalConcepts(S,As), loadRequiredSamples(As),\\
&\quad member(EffectToBePredicted,As), predictPos(X,GoalConceptToBePredicted), X\backslash = [].
\end{aligned}$$

1. Intersections

Positive: `[[e1(_),oa1(_),ap1(_),ap3(_),f1(_),cc5(_)],`
`[ap1(_),f1(_),cc5(_)],[ap1(_),f1(_)],[oa1(_),f1(_)],`
`[oa1(_),ap1(_),f1(_),cc5(_)],`
`[e2(_),e3(_),oa2(_),ap1(_),ap2(_),f1(_)],[e3(_),ap2(_),f1(_)],[e4(_),oa1(_),ap1(_),`
`f1(_),cc5(_)]]`

Negative: `[[oa1(_),ap2(_),f1(_),cb7(_)]]`

Unassigned examples:

2. Hypotheses

Positive: `[e1(_),oa1(_),ap1(_),ap3(_),f1(_),cc5(_)],[ap1(_),f1(_),cc5(_)],`
`[ap1(_),f1(_)],[oa1(_),f1(_)],[oa1(_),ap1(_),f1(_),cc5(_3B60)],`
`[e2(_),e3(_),oa2(_),ap1(_),ap2(_),f1(_)], [e3(_),ap2(_),f1(_)],`
`[e4(_),oa1(_),ap1(_),f1(_),cc5(_)]]`

Negative: `[[oa1(_),ap2(_),f1(_),cb7(_)]]`

Contradicting hypotheses: `[]`

The clauses for hypotheses here are:
`cb5(X)>-e1(X),oa1(X),ap1(X),ap3(X),f1(X),cc5(X);ap1(X),f1(X),cc5(X);ap1(X),f1(X).`
`cb5(X)>- not ( oa1(X),ap2(X),f1(X),cb7(X)).` Note that all intersections are turned into hypotheses because there is no overlap between positive and negative ones

3. Background (positive and negative objects with respect to the goal concept `cb5`)

Positive:
`[[e1(o1),oa1(o1),ap1(o1),ap3(o1),f1(o1),cc5(o1)],[e1(o2),oa1(o2),ap1(o2),ap3(o2),`
`f1(o2),cc5(o2)],`
`[e2(o7),e3(o7),oa2(o7), ap1(o7),ap2(o7),f1(o7),cc5(o7)],`
`[e2(o8),e3(o8),oa2(o8),ap1(o8),ap2(o8),f1(o8)],`
`[e3(o11),oa1(o11),ap2(o11),f1(o11),cc4(o11),cb7(o11)],[e4(o15),`
`oa1(o15),ap1(o15),f1(o15),cc5(o15)],`
`[e4(o16),oa1(o16),ap1(o16),f1(o16),cc5(o16)]]`

Negative:
`[[e5(o17),oa1(o17),ap2(o17),f1(o17),cc6(o17),cb7(o17)],[e6(o18),oa1(o18),ap2(o18)`
`,f1(o18),cc4(o18),cb7(o18)]]`

Inconsistent: `[]`

4. Prediction for `cb5` (objects `o10`)

Positive: `[[e3(o10),oa1(o10),ap2(o10),f1(o10),cc4(o10)]]`

Negative: `[]`

Inconsistent: `[]`

Uninstantiated derived rules (confirmed hypotheses)

```
cb5(O):- e3(O), oa1(O),ap2(O), f1(O), cc4(O).
```

Fig. 4. The *Jasmine* prediction protocol. Steps are numbered in accordance to the units at Fig. 3.

section. Anti-unification is the inverse operation to the unification of formulas in logic programming. Unification is the basic operation which finds the least general (instantiated) formula (if it exists), given a pair of formulas. Anti-unification was used in [21] as a method of generalization; later this work was extended to form a theory of inductive generalization and hypothesis formation. Anti-unification, in the finite term case, was studied as the least upper bound operation in a lattice of terms [3,30].

For example, for two formulas p(a, X, f(X)) and p(Y, f(b), f(f(b))) their anti-unification (least general generalization) is p(Z1, Z2, f(Z2)). Conversely, unification of this formulas, $p(a, X, f(X)) = p(Y, f(b), f(f(b)))$ will be $p(a, f(b), f(f(b)))$. Our logic programming implementation of anti-unification for a pair of conjunctions, which can be customized to a particular knowledge domain, is presented at Fig. 5.

Although the issue of implementation of the anti-unification has been addressed in the literature (e. g. [33,35]), we present the full code to have this paper self-contained. In a given domain, additional constraints on terms can be enforced to express a domain-specific similarity. Particularly, certain arguments can be treated differently (should not be allowed to change if they are very important, or should form a special kind of constant). A domain-specific code should occur in the line shown in bold.

The simplest way to express similarity between objects is to use attribute features. For example, to express a similarity between amino acids, we use their attributes such as polarity and hydrophobic properties. These attributes may be specific to a narrow domain of analysis (immunoglobilins in our example, [23]). If one prefers to have a stronger control over comparison of similar and distinct attributes of amino acids, additional terms should be introduced to express these attributes. In our example of amino acids, if one considers hydrophobic and hydrophilic amino acids as ones which do not have any similarity,

$$hydrophobic(other\_attribute1, \ldots, other\_attribute2)$$
$$\text{and } hydrophylic(other\_attribute1, \ldots, other\_attribute2)$$

predicates should be used. Instead of *residue(sequence)* terms one would use

$$residue(sequence, hydrophobic(other\_attribute1, \ldots, other\_attribute2))$$

to take into account hydrophobicity as a potential cause of certain property of a protein. Such predicates will then be treated as desired by the anti-unification procedure. Moreover, predicates for relations between objects are naturally accepted by *Jasmine* framework. As only we explicitly incorporated hydrophobicity into our object representation, it cannot be reduced to the attribute-value logic settings.

```
similar(F1, F2, F):- antiUnifyFormulas(F1, F2, F).

antiUnifyFormulas(F1, F2, F):- clause_list(F1, F1s), clause_list(F2, F2s),
   findall( Fm, (member(T1, F1s), member(T2, F2s),
        antiUnifyTerms(T1, T2, Fm)), Fms), %finding pairs
   %Now it is necessary to sort out formulas which are not most general within the list
   findall( Fmost, (member(Fmost, Fms),
      not ( member(Fcover, Fms), Fcover \= Fmost,
            antiUnifyTerms(Fmost, Fcover, Fcover)) ), Fss),
     clause_list(F, Fss). % converting back to clause

antiUnifyTerms(Term1, Term2,Term):-
  Term1=..[Pred0|Args1],len(Args1, LA),% make sure predicates
  Term2=..[Pred0|Args2],len(Args2, LA),% have the same arity
  findall( Var, (  member(N,  [0,1,2,3,4,5,6,7,8,9,10 ]), % not more than 10 ar-
guments
    [! sublist(N, 1, Args1, [VarN1]), %loop through arguments
      sublist(N, 1, Args2, [VarN2]),
      string_term(Nstr,N), VarN1=..[Name|_],    string_term(Tstr,Name),
      concat(['z',Nstr,Tstr],ZNstr),    atom_string(ZN, ZNstr) !],
  % building a canonical argument to create a variable as a result of anti-unification   ifthenelse( not
(VarN1=VarN2),
  ifthenelse(( VarN1=..[Pred,_|_],VarN2=..[Pred,_|_]),
      ifthenelse( antiUnifyConst(VarN1, VarN2, VarN12),
       %going deeper into a subterm when an argument is a term
         (Var=VarN12),     Var=ZNstr) ),
           OR domain-specific code here for special treatment of certain arguments
    % various cases: variable vs variable, or vs constant, or constant vs constant
            Var=ZNstr),Var=VarN1)         ), Args),
      Term=..[Pred0|Args].
```

Fig. 5. The clauses for logic program for anti-unification (least general generalization) of two formulas (conjunctions of terms). Predicate *antiUnify (T1, T2, Tv)* inputs two formulas (scenarios in our case) and outputs a resultant anti-unification.

There are other *Jasmine*-compatible approaches to computing similarities except the anti-unification. In particular, it is worth mentioning the graph-based approach of finding similarities between chemical substances [2]. The operation of finding the maximum common subgraphs serves the purpose of anti-unification in such the domain. This operation was subject to further refinement expressing similarities between scenarios of multi-agent interaction, where it is quite important to take into account different roles of edges of distinct sorts.

Novice users of *Jasmine* are advised to start building the similarity operation as an intersection between objects' features (unordered set of features) and obtain an initial prediction. Then, when the explanations for predictions are observed, the users may feel that less important features occur in these explanations too frequently, and anti-unification expression should be introduced so that less important features are nested deeper into the expressions for objects' features. Another option is to build a domain-specific Prolog predicate which computes unification, introducing explicit conditions for selected variables (bold line at the Fig. 5).

5. Applying *Jasmine* to the prediction of secondary structure of proteins

Now we are ready to outline the *Jasmine* representation language for this domain, which has been introduced in Section 2 as an example of *Jasmine* reasoning. The fact (cause) that for a sequence o2 the fragment e is cluster 1 (e.g. TLYLQMN, Table 1) is expressed as e1(o2). Analogously, the fact that the fragment CB of the same sequence o2 is five amino acids long (the goal concept) is expressed as cb5(o2). The reader may suggest terms e(o2,1) and cb(o2, 5) instead, but such generalization needs to be done in a more intricate way to suite anti-unification operation over the respective terms. Note that since the predicates are not always unary, our settings are irreducible to attribute-value logic. Since we have a limited number of values for cluster numbers and fragment length, we use unary predicates for clarity.

Being a built-in solver of inverse problems, *Jasmine* is capable of finding the optimal classification structure for a refined dataset. A classification is *optimal* if classifying criteria allow achieving the highest possible precision of the data being classified (when an object is related to a class, its features are predicted with the highest accuracy). In terms of graph topology, this means a minimum number of common sub-classes for classes, common sub-subclasses for sub-classes, etc. In a general situation the optimality of the classification graph should be measured in terms of amount of inconsistencies between the data components (ambiguous classification), as well as prediction accuracy (compare with [24]).

In the domain of proteins, the main predication problem is how a sequence of amino acid residues causes the protein to fold in a certain way. This problem can be solved by physical simulation only for short sequences, therefore machine learning needs to be involved. Currently, the most efficient approaches to fold recognition are statistical

Table 2
Evaluation of recognition accuracy improvement by *Jasmine*

Relating a sequence to a class using:	Data sets (protein families)				Average accuracy improvement
	Heavy chain variable domain		Light chain variable domain		
	Mouse	Human	Mouse	Human	
Homology method	5.4	6.2	7.3	8.1	1
Hierarchical classification method	11.1	11.3	12.7	12.2	1.75
Optimized hierarchical method using *Jasmine*	13.5	13.7	13.8	14.1	2.04

(including SVM); however they do not help to understand the mechanisms of how individual residues affect the fold.

Searching through a variety of "*known features* → *goal concepts*" models in terms of consistency of generated hypotheses, *Jasmine* has delivered the set of classification criteria which were initially obtained manually, taking into account the spatial locations of immunoglobulin strands. *Jasmine* independently confirms the classification structure obtained using additional (physical) knowledge about our dataset. Therefore, in the case of immunoglobulins, the *Jasmine*-assisted search for clearer causal links generates such additional knowledge!

Let us refer again to Fig. 1 for the example of reasoning settings which lead to the construction of the classification graph. We select the positive target $|C'C''| = 5$ and search for hypotheses which could serve as criteria for this target. The negative target cases are sequences when the length of $|C'C''|$ is more or less than five amino acids. Following the reasoning protocol similar to the one depicted at Fig. 4, we look for the features (values at the nodes of the classification graph) which are common for the positive objects, and are not common for the negative objects. We observe that $A' = 3$ at the subclass level is common for all positive examples (first two subclasses at the top of Fig. 4), and are not common (although occurs in the third subclass) for negative examples.

Also, at the subclass level, $F = 1$ is common between the positive objects; however, it is also common for the two subclasses of the third class ($E = 3$); therefore $F = 1$ cannot serve as a $|C'C''|$ criterion. Analogously, all targets (predication results at Fig. 1) were analyzed, and the classification tree is built including as much desired criteria (confirmed hypotheses) as possible for the given dataset. Notice that the clustered-based values can be interpreted via amino acids ($AA' = 1$ means AA' is SG (G or A)).

We initiate the reasoning process for each particular value of the target (setting other values as *negative*). When all hypotheses for the targets have been formed, we estimate the resultant prediction accuracy and compare it with the traditional (homology-based) techniques for secondary structure predication. To draw the quantitative comparison, we then change the prediction settings towards the primary structure (amino acids) instead of the secondary structure, using the same reasoning protocol.

To quantitatively evaluate how the recognition accuracy can be increased by exploration of causal links by *Jasmine*, we predict the whole immunoglobulin sequence, given its few residues. In our evaluation, we follow the prediction settings of the study [7] where certain correlations between residues were discovered, required for such prediction. Also, we draw the comparison with the baseline approach of sequence homology computation.

Classification-based and homology-based approaches have different methodologies to predict the unknown sequence residues. The former initially determines the status of belonging to a class (and subclass), and then enumerates the possible residues as the attributes of the determined class. The latter calculates the distances from all available sequences and suggest the closest sequence as a prediction.

The set of 500 sequences were used for training (building classification patterns) and the set of 200 sequences were used for evaluation of prediction accuracy in each recognition settings (Table 2). Values in the table are the times the predictions are better than a random prediction for the totality of the residues that are the subject of the prediction. Using hierarchical classification presented in [7] improves the homology-based prediction by 175.2%. Use of *Jasmine* to optimize the classification-based prediction delivers a further 16.5% improvement.

These 16.5% of improvement is due to the fact that a higher number of correlation rules (between the features and targets) have been revealed as a result of the hybrid reasoning process, than as a result of the manually built classification graph [7]. Only simpler and incomplete versions of derived rules such as presented at the bottom of Fig. 4 can be obtained via the manual analysis of clustered (and possibly visualized) data. In particular, the manual analysis derived a simpler version of the rule `cb5(0):-e3(0), fl(0), cc4(0)`, which delivered a number false-positive prediction due to its weakness (a lesser number of constraints). On the other hand, homology-based approaches do not results in explicit rules for individual amino acids at all.

Hence the statistical-based homology method, which is a traditional one for relating a protein sequence to a protein family [19,23,46,47], can be significantly improved by an automated procedure for finding rules for occurrences of certain residues.

6. Mining the evolutionary data

Prediction of the protein function has been one of the most important tasks over the last two decades. The functional prediction is traditionally based on the observation that, two homologous proteins evolved from a common ancestral protein, are expected to have similar functions. Sequence similarity is the basis for a vast number of computational tools to derive protein homology. Recent approaches have attempted to combine the sequence information with evolutionary data [4,13,27,28]. Some approaches of comparative genomics have extended the homology method by identifying inter-connected proteins which participate in a common structural complex or certain metabolic pathways, or fuse into a single gene in some genomes. Prediction of gene functions was found to be a good application area for Inductive Logic Programming [10,11,20].

The phylogenetic profile of a protein is represented as a vector, with each component corresponding to a specific genome [14,15]. A component is equal to *one* if there is a significant homology with that protein in the corresponding genome, and equal to *zero* if it is not the case.

Considering the spectrum of problems targeting the prediction of protein *functions*, we focus on the following problem: how the phylogenetic patterns for different species are correlated with each other. This problem is tightly connected with finding the scenarios of gene evolution, given the phylogenetic tree for the species [17]. In this section, we apply *Jasmine* to the database of Clusters of Orthologous Groups of proteins (COGs), which represents a phylogenetic classification of the proteins encoded in complete genomes [25]. Discovering the structure of how phylogenetic profiles are interconnected, we will be addressing the dual problem of how the binary vectors of distribution of COGs through species are correlated. The dataset consists of the matrix of 66 (species) by 4874 (clusters of proteins). The objective of this section is to attempt to explore an alternative approach to building evolutionary tree.

The way we explore the evolutionary data is based on expressing correlations between objects or sets of objects using *causal links.* A pair of objects (or object sets) is causally linked if certain observations about the first object (or set) imply certain observations about the second object (this implication is based on *Jasmine*'s reasoning schema). The problem of finding correlation in a dataset is then formulated as finding the causal links between components of this dataset and characterizing them via the overall structure of such links. In this section we explore how these links can express evolutionary relationships.

Hence we explore the following cases of causal links (the third case is the combination of the first and the second cases):

Case (1). Whether for a given COG its existence in certain species may *provide an evidence* (imply, be causally linked) to the existence of this COG in the specified set of selected species. Here species are *features* and COGs are *objects* (#features $<<$ #objects).
Case (2). Whether for a given species occurrence of certain COGs in this species implies the occurrence of the other selected COG in this species. In this case COGs are features and species are objects.
Case (3). Whether a set of occurrences of given COGs in a given set of species implies the occurrence of the particular COG in the particular species. Either *feature–object* assignments above are possible, and the variables need to range over sets of COGs or species respectively; also, facts need to be augmented by clauses.

For the sake of simplicity in our examples, we use unary predicates, however, arbitrary terms can be used. For instance, if protein sequence information for COGs is taken into account, the expression for a feature of object will look like:

$$a3(o7, sequence(o7, cogRepresentative))\text{instead of just}a3(o7).$$

The most efficient data mining occurs in an interactive mode, when the current causal link exploration results are visualized for a user, who then can plan the further direction of exploration online. We have selected a straightforward approach to data visualization and display our matrix as a grid with columns for selected COGs and rows for selected species (Fig. 6). Having chosen a data fragment for a group of COGs and species, it is possible to perform an analysis of interconnection between phylogenetic profiles within the selected dataset (which COGs and which species data can deliver a correct prediction for a given COG and a given species).

For any cell (a given COG and a given species), we can verify whether it is consistent with a currently-selected dataset or not: we can run a prediction of the cell (binary) value, temporarily declaring it ''unknown''. In the Fig. 6 COGs are *objects*, and species are *features* of these objects which either obey a selected feature for particular object or do not obey it. The prediction protocol for a given cell is presented in Fig. 6 in the sequence and is analogous to our presentation of the *Jasmine* reasoning protocol in Section 2.

The shown *Jasmine* setting targets the causal link of Case (1) above; if the matrix is transposed, causal link of Case (2) would be a subject of *Jasmine* analysis. To approach the causal link of Case (3) above, one needs to analyze either (1) and use the generated hypotheses of (2) in addition to the facts of (1), or the reverse. Overall, these kinds of analyzes evaluate how strongly the distributions of COGs for various species are correlated.

Naturally, for two parts of our dataset sets (sub-matrices) S_1 and S_2, the higher the number (and accuracy) of

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Notes
a1	1	0	0	0	**1**	**1**	0	0	1	0	0	0	0	0	0	
a2	1	0	0	0	0	1	0	**?**	0	0	1	0	1	0	0	
a3	1	0	0	0	**1**	**1**	0	0	1	0	1	0	1	0	0	
a4	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	
a5	0	0	0	0	**1**	**1**	0	0	1	0	1	0	1	0	0	
a6	1	0	0	0	**1**	1	0	0	1	0	0	0	1	0	0	
a7	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	
a8	0	0	0	0	**1**	0	0	1	**1**	0	0	0	1	0	1	
a9	0	1	0	0	1	0	0	1	0	0	1	0	1	0	0	
a10	**1**	0	0	0	0	**1**	1	1	0	0	**1**	0	0	1	0	
a11	0	**1**	**1**	0	0	0	0	0	0	1	0	0	1	0	0	
a12	0	**1**	0	0	0	0	1	**1**	0	0	1	0	0	0	0	
a13	0	**1**	1	0	0	1	**1**	**1**	0	0	1	0	1	0	0	
a14	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	
a15	**1**	1	1	1	0	1	1	1	0	**1**	1	**1**	1	1	1	
a16	1	1	0	1	0	0	0	**1**	0	1	0	0	0	0	0	
a17	1	1	0	0	0	0	1	1	0	1	1	1	1	0	0	
a18	1	1	0	1	0	0	1	1	0	0	1	1	0	1	1	
a19	1	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0 a lot of positive intersections but none predicted
a20	1	1	0	**1**	0	0	0	**1**	0	0	1	0	0	1	0	5, 8, 9, 13, 25 low number of positive but well predicted
a21	0	1	0	0	0	**1**	0	0	0	0	**1**	0	0	1	0	many positive values but no predictions
a22	1	1	0	1	0	1	1	**1**	0	0	1	0	0	1	0	all negative

Positive intersections	**[[a14(_),a23(_)],[a13(_),a23(_),a26(_)],[a13(_),**
Negative intersections	[[a1(_),a6(_),a24(_)],[a1(_),a5(_),a6(_),a9(_),a2
Unknown hypothesis	[[a1(o22),a2(o22),a4(o22),a6(o22),a7(o22),a8(o
Positive hypotheses	**[[a14(_),a23(_)],[a13(_),a23(_),a26(_)],[a13(_),**
Negative hypotheses	[[a1(_),a6(_),a24(_)],[a1(_),a5(_),a6(_),a9(_),a2
Inconsistent hypotheses	[[a23(_2404)]]
Background for positive h	**[[a13(o7),a14(o7),a23(o7),a26(o7)],[a1(o10),a6**
Background for negative hy	[[a1(o1),a5(o1),a6(o1),a9(o1),a20(o1),a24(o1),a
Background for inconsistent	[]
Positive prediction	**[]**
Negative prediction	[[a1(o22),a2(o22),a4(o22),a6(o22),a7(o22),a8(o
Inconsistent to predict	[]

Fig. 6. The fragment of the data set for analysis is: rows are species a1–a22 (features) and columns are COGs o1–o15 (objects). Note that a subset of species and a subset of COGs are currently selected. The cell values indicate the existence of a given COG for a given animal. The user interface for *Jasmine* is designed in a way that any cell value can be set as "unknown" and its prediction can be attempted. The right pane contains the reasoning protocol in the same format to the one presented in Fig. 4.

the obtained predictions of S_1 given S_2, or S_2 given S_1, the stronger the correlation.

Fig. 6 gives an example of interactive exploration of correlation between phylogenetic profiles using causal links. For each species, a user attempts to verify whether the occurrences of COGs can be predicted, given the selected subset of the matrix.

The question mark indicates the current cell to be predicted; only one cell can be predicted at a time. In this example, the prediction of a2(COG8) is attempted, since the question mark is specified in the respective cell. The prediction is 0 (no COG8 in species a2), and the "Negative prediction" line on the right of Fig. 6 enumerates the source of prediction: species a1 at COG o22, species a2 at COG22, etc. Hence, the prediction protocol on the right corresponds to the Yes/No prediction for a given cell. In the given setting, for each species, the user may specify peculiarities of the causes for the COGs occurring in this species in the fields on the right of the grid.

The prediction grid (area on the left) shows the results of previous explorations. The prediction is successful for the species a1, a3, a5, a8 ... shown in bold in the leftmost grayed column: cells with ones (shown in bold as well) are predicted properly for COGs 5 and 6. Consequently, there are causes for the existence of COGs for these species (for a number of reasons which may have a domain-specific biological interpretation). These causes are read from the prediction protocol areas on the right in the lines "Positive prediction", "Negative prediction", and "Inconsistent prediction" (three lines at the bottom). For our further analysis, we collect all hypotheses for each species which provided the prediction (Table 3).

Table 3 presents the features which have lead to the given goal concept (conditions for occurrences of COGs in species). For each species (the left column), we perform the prediction for the existence of each COG and collect the hypotheses that deliver the correct positive prediction. If the prediction is wrong, or there is a negative case, the hypotheses are not taken into account. The right column enumerates the COGs which substantiate the conditions delivering positive correct prediction.

As an example of our analysis, we suggest the reader observe that the species a3 and a5 imply the occurrence of COGs in species a1. In the Fig. 6 there is the bold **1** in the fifth column for data (COG 5, or o5). The occurrence of COG 5 for species a1 is implied by the species a16 (refer to Table 3, first row below the header, the right column). The fact that the COG o9 occurs in a3 (in particular) is written as [**a3(o9)**,a5(o9),a6(o9),a8(o9)]; therefore, COG o9 contributes to the causal link a3(o9) → a1(o5). Finally, notice the bi-directional node a3 ↔ a1 (we have just verified this node in one direction).

Finally, the links between the species, showing how predictions have been made, are visualized as a graph (Fig. 7).

Each interrelationship between a pair of species is obtained as a link between the occurrence of a particular COG for the first species (and some other species) and the occurrence of this COG in the second species. Each edge can be one-directional or bi-directional. Nodes are subscribed by the species numbers; they are in the one-to-one correspondence to the evolutionary tree below (Fig. 8).

As a result, our Jasmine-assisted analysis suggested a novel approach to represent evolutionary dependence between biological species. In the traditional approach, the evolutionary relations are inferred, based on the mea-

Table 3
Enumeration of all features which have lead to the given goal concept (prediction of the existence of a COG for a given species)

species, a1-a8	Lists of conditions (sets of species) to infer that the given species occurs in a current COG. Only those conditions are selected which lead to proper predictions.	COGs that are present in the conditions (on the left). Occurrence of these cogs in the conditions (presented on the left) implies the occurrence of the selected (other) COGs for the species under investigation (leftmost columns).
a1	[[a3(_),a6(_)],[a3(_),a5(_),a6(_)],[a3(_),a5(_),a6(_),a8(_)],[a3(_),a5(_),a8(_)]]	[a3(o9),a5(o9),a6(o9),a8(o9)], [a3(o20),a5(o20),a6(o20),a8(o20),a9(o20)], [a3(o25),a5(o25),a8(o25),a14(o25),a15(o25),a21(o25)]
a2		
a3	[[a1(_),a5(_),a6(_)],[a1(_),a5(_),a15(_),a21(_)]]	[a1(o9),a5(o9),a6(o9),a8(o9)], [a1(o20),a5(o20),a6(o20),a8(o20),a9(o20)].
a4		
a5	[[a1(_),a3(_),a6(_)],[a3(_)],[a3(_),a6(_),a8(_)],[a1(_),a3(_),a6(_),a8(_)],[a1(_),a3(_),a8(_)],[a3(_),a9(_)],[a3(_),a6(_),a8(_),a9(_)]] [[a1(_),a3(_),a6(_)],[a1(_),a3(_),a15(_),a21(_)]]	[a1(o20),a3(o20),a6(o20),a8(o20),a9(o20)] [a1(o25),a3(o25),a14(o25),a15(o25),a21(o25)]
a6		
a7		
a8	[[a9(_)],[a3(_),a5(_),a6(_)],[a1(_),a3(_),a5(_),a6(_)],[a1(_),a3(_),a5(_)],[a3(_),a5(_),a6(_),a9(_)]] [[a1(_),a3(_),a5(_)]]	a1(o20),a3(o20),a5(o20),a6(o20),a9(o20)]

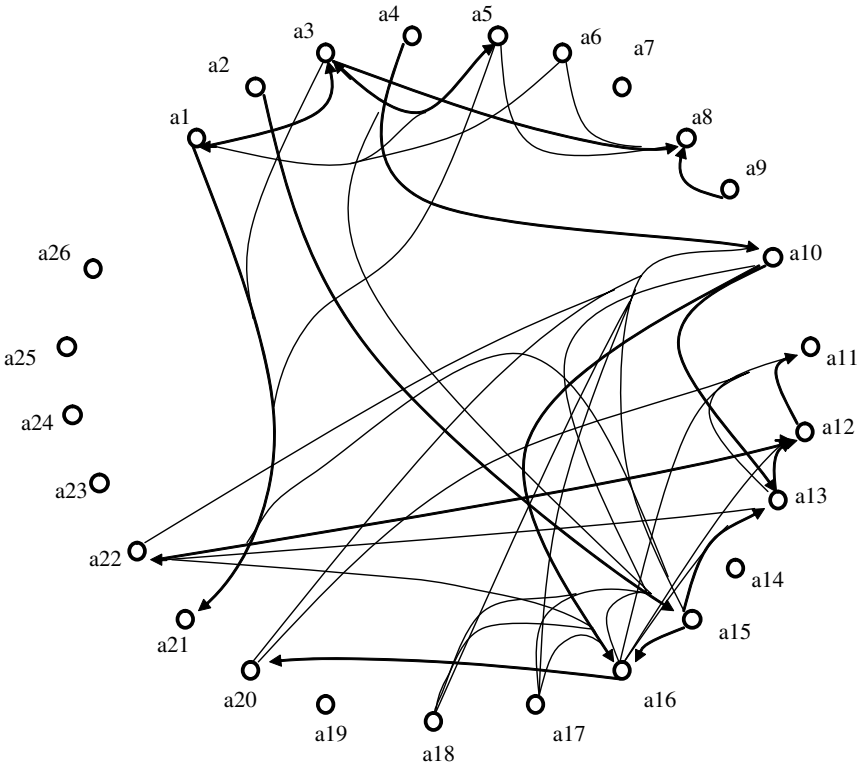

Fig. 7. The graph of causal links between species may serve as an alternative approach to building an evolutionary tree for species a1–a22 (corresponds to the selection of species and of Fig. 6).

surements of distance (difference) between the species (expressed, for example, by distances between respective protein sequences). Unlike the traditional approach, we introduce the one where evolutionary relationships are of the *causal* nature. These evolutionary relationships are based on the revealed correlation that certain features of species *determine* the goal concepts of other species. We believe the proposed evolutionary approach better fits the

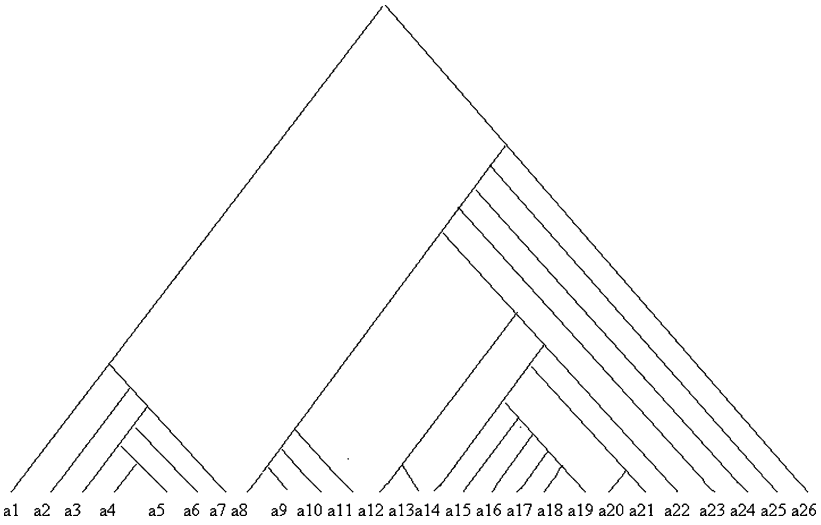

Fig. 8. The fragment of traditional evolutionary tree for 26 species.

biological intuition behind the evolution process than a distance-based one (see e.g. [49]).

7. Related work

In this paper we presented the reasoning model and implementation of a deterministic machine learning system, *Jasmine*, and illustrated its capability by two problems in bioinformatics. In this section, we will draw the comparison of the *Jasmine* reasoning strategy with other inductive machine learning approaches, including analogical reasoning. In addition, we will consider both *Jasmine* limitations and advantages over statistical approaches, and outline the bioinformatics-specific features of machine learning.

Originally, the JSM-method was formulated in early 1980s in terms of a predicate logic, and is an extension of the First-Order Predicate Logic with quantifiers over tuples of variable length [5]. *Jasmine* complies with the common paradigm of learning from positive and negative examples [18]: given descriptions of positive and negative examples relatively to a goal concept, positive hypotheses are ''generalized descriptions'' of a subset of positive examples that do not ''cover'' any negative example. Generalization of negative hypotheses is defined similarly. *Jasmine* is designed as a logic program for convenient integration with logical as well as knowledge representation formalisms realized using logic programming [26]. A procedural reasoning system of the JSM type has been proved efficient in bioinformatics [2], and *Jasmine* further leverages not only JSM reasoning capabilities, but also the automated adjustment to new domain and integration with other reasoning components inherent to logic programming.

As to the comparison with competitive approaches to logic-based implementation of machine learning, it is worth describing Inductive Logic Programming (ILP) mentioned in the Section 1. In ILP (see, e.g., [20]) the idea of a version space [18,31] is implemented by the use of the generality relation (''to be more general than, or to be equal to'') which operates on descriptions. Classifiers are considered the inference relations in logic and the set of classifiers is thought of as a subset of the set of logic programming formulas. The major drawback of the version spaces, where classifiers are defined syntactically, is that in the case of a too restrictive choice involving purely conjunctive classifiers, there is no classifier that matches all positive examples. This situation which is referred to as ''collapse of the version space'' is quite frequent, in particular, when classifiers are conjunctions of attribute value assignments and ''wildcards'' are used for arbitrary values. If the expressive power is increased syntactically, e.g., by introducing disjunction, then the version space tends to become trivial, while the most specific generalization of positive examples becomes ''closer'' to or just coincide with the set of positive examples.

In contrast, the JSM-hypotheses, as has been proven in [29], offer the methodology of ''context-restricted'' disjunction: not all disjunctions are possible, but only those comprising minimal hypotheses. As the origin of the generality order can be arbitrary, in the JSM-method an analyst can use descriptions of: positive and negative examples given by formulas of description logic, by conceptual graphs, etc; and only an initial generality relation has to be specified.

Concurrently, the ILP methodology initially makes over-generalized conjectures, which, are first refuted, and then made more specific to cover only positive examples. Quite the reverse to this methodology, the JSM-method makes inductive ''leaps'' from the set of descriptions of examples to the set of descriptions of their generalization; the latter being ''far apart'' from the former.

Similarity in the JSM-method is not a relation, but an operation, which is idempotent, commutative and associative (i.e., it induces a semi-lattice on descriptions and their generalizations), a particular way of least general generalization of examples. JSM-method first constructs inductive

generalization of positive examples, which may then be disproved by negative examples. Symmetrically, negative examples and their generalizations may be disproved by positive examples.

Described in algebraic terms of Formal Concept Analysis (FCA), the JSM-method allows for simple and efficient implementation in procedural or declarative programming languages (using standard algorithms for computing closure systems and concept lattices [32]). The issue of duality between features and objects, illustrated in Section 5, is thoroughly addressed by the FCA approach [3]; specifically in *Jasmine* and how its (higher-order calculus) settings can be reduced to FCA (propositional) approach, if all predicates are unary and there are no explicitly specified links between the involved features (clauses).

Traditional techniques for performing analogical reasoning require *making a guess* about what information should be transferred from a previous analogous observation to a new situation. To avoid such guessing, the *justified* version of analogical reasoning attempts to map inference rules from analogs to target objects, and such inference rules might encode ''causal relations''. In this respect, the closest classical system to *Jasmine* is ANALOGY [45] of Winston and co-workers; it learns physical descriptions of objects when given their functional or attribute-based definitions as an input. ANALOGY transforms an attribute-based definition (e.g. amino-acid properties in our case) into a higher level physical and structural description (such as 3D structure in our case). There are multiple ways of formalizing causality, explanation and strength of evidence addressed in the computer science literature; in this study, we use the precise and very specific notion of causality relating it to *Jasmine*-specific inference.

Jasmine is not well-suited to handle large amounts of data, because the number of hypotheses grows exponentially with the number of examples. Focusing on flexibility, interactivity and expressiveness of a knowledge representation language, *Jasmine* requires conventional means to reduce the dimension of data such as clustering, filtering or other pre-processing approaches. Therefore, the methodology of mining large datasets for causal links with *Jasmine* is: to apply pre-processing first, and then operate with the reduced dataset, so that the results allow intuitive semantic interpretations and can be supported by explanations comprehensible by a human expert.

We proceed to the comparison of *Jasmine*-based analysis with a generic statistical analysis (see [19] for introduction to recent approaches). Mining the evolutionary and genomic data, the value of a statistical technique is that it may shed a light on the most common principles of data organization in biology [9]. Statistical approaches are usually less computationally intensive and less sensitive to unreliable data. However, averaging may conceal the correlation of higher consistency with a lower number of representatives. To overcome this problem, a deterministic approach like *Jasmine* can deliver simple rules with systematic exceptions instead.

Another disadvantage of using statistical approaches is that it is possible to loose important information at the initial step so that it cannot be restored in successive steps. Rather than considering evolutionary data as noisy, we hypothesize that the data set is reliable and build complex rules initially. Then, in contrast to the statistical approach, we simplify the rule by explicit enumeration of exceptions, which may form a basis of a more specific level of rules at successive steps of data mining.

When we apply *Jasmine* to a large dataset, a statistical approach may precede applying *Jasmine*. Hence, *Jasmine* would be applied to a dataset where some information is lost, and would reveal some correlations, and the selected statistical approach would be applied again taking these correlations into account. This process can be iterative. After that, the obtained rules can be applied to an original dataset, and some objects may be assigned the status of *exceptions*. However, since the loss of information occurs only for the sake of more efficient generation of the initial version of rules, the results of such training strategy are considered those obtained *without loss of information.* In terms of our example with the prediction of the fragments of immunoglobulin sequences, once we discover a correlation between these fragments, we can drill down to the level of individual amino acid and predict its occurrence with a higher accuracy.

We conclude this section by focusing on the bioinformatics-specific machine learning settings. There has always been a debate in the bioinformatics community about the role of approximation and explanation [41]. Some ''black box'' machine learning approaches such as neural network, genetic algorithms, and hidden Markov model only produce the result without any explanation from the learning process. Although they produce a better result compared to the rule-based inductive learning methods, their approaches are hard to understand and interpret into useful knowledge. As a result, interactive data exploration requires various kinds of rule generators such as decision trees to produce an explanation that is understandable by humans, and also accomplished the task. To solve a particular prediction task, knowledge at a particular level may remain implicit; however further domain exploration may require returning to this intermediate step and re-applying a rule-based method.

How can a bioinformatics domain be characterized in terms of deficiencies of an initial domain theory? As we have seen in our examples, it is:

- *incomplete*: domain theory does not necessarily relate each object to a class.
- *inconsistent* because some objects are related to both positive and negative classes.
- *incorrect* because not all predictions are biologically plausible.

- *intractable* because predictions with explanations for some objects cannot be obtained within specified time/space resources. Hence EBL has not found an extensive number of applications in bioinformatics (see e.g. [38]).
- *oversensitive* to single examples and the requirement of usable features (operation criterion) and goal concepts to be deductively linked.

The protein-related example in the given paper shows that an operation criterion cannot always be suggested, because associative features may occur at distinct physical levels (amino acids and 3D structure). Certain features of *Jasmine*, including iterations of the prediction procedures, integration with DeLP and a user interface which provides flexibility for setting goal concepts, help to overcome the problem of these deficiencies of an initial domain theory or a lack of it.

8. Conclusions: main features and advantages of Jasmine

The main features of *Jasmine* are as follows (not all of them have been demonstrated in the current paper due to its brevity):

- Knowledge base for machine learning combines data (facts) with rules (causal links between the data, clauses, productions, default rules etc.);
- Convenient component–based integration to other units implementing a specific approach to reasoning;
- Implements inductive, deductive, abductive and analogical reasoning [1,5,26];
- Implemented as a logic program with full-scale meta-programming extensions; meta-programming is intended to operate with domain-dependent predicates at a higher level of abstraction;
- Builds rules and meta-rules, and explains how they were built. Gives the cases (objects) which are similar to target case (to be predicted) and the cases which are different from the target [2]. Also, *Jasmine* enumerates the features of cases which separate similar and different cases;
- Distinguishes typical and atypical rules, rules can be defeated. Defeasible Logic Programming [42] is integrated into *Jasmine*;
- Front end of *Jasmine* and its data exchange is compatible with Microsoft Office.

In comparison with the implementations of JSM approach which inspired *Jasmine*, the latter has the following advantages:

1. Implementing JSM as a logic program, *Jasmine* can derive facts from clauses in addition to operating solely with facts. These clauses can bring in the representation means of particular approach to logical AI including situation calculus, default and defeasible reasoning, temporal and spatial logics, reasoning about mental attitudes and others. *Jasmine* uses facts dynamically obtained by external reasoning machinery, and at the same time *Jasmine* yields hypotheses as clauses which are processed by accompanying deductive systems. Therefore, deductive capabilities of *Jasmine* are beyond the ones of JSM; in particular, *Jasmine* is better suited for active learning (compare with [19]).
2. Current implementations of JSM are procedural and therefore strongly linked with domain specific features; *Jasmine* is fully domain-independent. Ideology of JSM-based analysis is strongly affected by its procedural implementation. Taking advantage of the capability of the extended logic program to update clauses on the fly, *Jasmine* can operate with hypotheses and the initial dataset with much higher flexibility than JSM. Using GUI of *Jasmine*, it is easy to tackle an inverse prediction problem.
3. The measure of likelihood of hypotheses generated by JSM are rather limited. A single fact may prevent a hypotheses to be generated which covers a large number of samples. In contrast to JSM, *Jasmine* is capable of optimizing the initial dataset, eliminating examples which deliver atypical solutions. Following the methodology of case-based reasoning, *Jasmine* refines the initial dataset of cases to adjust it to a given domain where prediction is to be performed.

For *Jasmine* as a logic programming system, its performance is a more critical issue than the size of a dataset, which is set by the limitations of particular implementation of Prolog (usually an order of a few gigabytes). When the number of examples reaches thousand, using *Jasmine* in the interactive mode becomes inconvenient.

In this paper we illustrated the data exploration strategies where the importance flexibility of building decision rules precedes the speed of computation. The dataset from the illustration examples are short not only due to space limitation, but also because we intend to demonstrate that the exploration initially starts with aggregated data of rather low dimension. When a structure of rules obtained from a reduced or clustered dataset becomes clear, the next step is to apply these rules to the whole dataset (which might not be the task of *Jasmine*). Hence the *Jasmine*'s performance is not a critical issue when *Jasmine* is used for initial high-level mining for new rules. It is worth mentioning that *Jasmine*'s functionality is not limited to biological domains [48].

In this paper we have demonstrated how a new method of data analysis allows posing new kind of problems for well-known data. Rather than evaluating *Jasmine* in standard settings, we show how the established domain in bioinformatics can leverage *Jasmine* capabilities. We also show how Jasmine assists in obtaining a clearer approach to hierarchical classification of proteins and a novel way to represent evolutionary relationships.

Jasmine's applicability to a wide spectrum of problems in bioinformatics provided further evidence that some aspects of scientific reasoning can be formalized and efficiently automated. This suggests that a system like *Jasmine* can be effective in a cost optimization of conducting experimental observations. Our presentation of *Jasmine* reasoning in this paper is such that the code from Section 2, Fig. 5 and a domain-specific expression for similarity to be written by the reader are sufficient for reasoning-intensive data mining. A supporting page for using *Jasmine* in bioinformatics is available [48].

References

[1] Anshakov OM, Finn VK, Skvortsov DP. On axiomatization of many-valued logics associated with formalization of plausible reasoning. Studia Logica 1989;42(4):423–47.

[2] Blinova VG, Dobrynin DA, Finn VK, Kuznetsov SO, Pankratova ES. Toxicology analysis by means of the JSM-method. Bionformatics 2003;19:1201–7.

[3] Davey BA, Priestley HA. Introduction to lattices and order. New York: Cambridge University Press; 2002.

[4] Dubchak I, Muchnik I, Holbrook SR, Kim S-H. Prediction of protein folding class using global description of amino acid sequences. Proc Natl Acad Sci USA 1995;92:8700–4.

[5] Finn VK. On the synthesis of cognitive procedures and the problem of induction. NTI Series 2, 1999;No. 1–2: p. 8–45.

[6] Furukawa K. From deduction to induction: logical perspective. In: Apt KR, Marek VW, Truszczynski M, Warren DS, editors. The logic programming paradigm. Springer: Berlin, Germany, Cambridge, MA, USA 1998.

[7] Galitsky BA, Gelfand IM, Kister AE. Predicting amino acid sequences of the antibody human VH chains from its first several residues. Proc Natl Acad Sci USA 1998;95:5193–8.

[8] Galitsky BA, Gelfand IM, Kister AE. Class-defining characteristics in the mouse heavy chains of variable domains. Protein Eng 1999;12:919–25.

[9] Galitsky BA. Bioinformatics data management and data mining. In: Rivero LC, Doorn JH, Ferraggine VE, editors. Encyclopedia of database technologies and applications. Hershey, PA, USA: Idea Group Reference; 2005.

[10] King RD. Applying inductive logic programming to predicting gene function. AI Mag 2004;25(1):57–68.

[11] King RD, Whelan KE, Jones FM, Reiser PKG, Bryant CH, Muggleton SH, Kell DB, Oliver SG. Functional genomic hypothesis generation and experimentation by a robot scientist. Nature 2004;427:247–52.

[12] Langley P. The computational support of scientific discovery. Int J Hum–Comput Stud 2000;53:393–410.

[13] Liao L, Noble WS. Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. J Comput Biol 2003;10:857–68.

[14] Marcotte EM, Xenarios I, van der Bliek AM, Eisenberg D. Localizing proteins in the cell from their phylogenetic profiles. Proc Natl Acad Sci USA V97 2000 vol. 22. p. 12115–20.

[15] McGuffin LJ, Jones TD. Benchmarking protein secondary structure prediction for protein fold recognition. Proteins 2003;52:166–75.

[16] Mill JS. A system of logic, racionative and inductive. London: Longmans, Green & Reader 1843.

[17] Mirkin B, Fenner T, Galperin M, Koonin E. Algorithms for computing parsimonious evolutionary scenarios for genome evolution, the last universal common ancestor and dominance of horizontal gene transfer in the evolution of prokaryotes. BMC Evol Biol 2003;3:2.

[18] Mitchell T. Machine learning. McGraw Hill; 1997.

[19] Ohno-Machado L. Research on machine learning issues in biomedical informatics modeling. J Biomed Inform 2004;37(4):221–303 (Editorial).

[20] Muggleton S, Firth J. CProgol4.4: theory and use. In: Dzeroski S, Lavrac N., editors. Inductive logic programming and knowledge discovery in databases; 1999.

[21] Plotkin GD. In: A note on inductive generalization. Machine Intelligence, vol. 5. Edinburgh University Press; 1970. p. 153–63.

[22] Popper K. The logic of scientific discovery. London: Hutchinson; 1972.

[23] Poupon A, Mornon J-P. Populations of hydrophobic amino acids within protein globular domains: identification of conserved "topo-hydrophobic" positions. Proteins 1998;33:329–42.

[24] Scott C, Nowak R. Dyadic classification trees via structural risk minimization, Neural Information Processing Systems 2002.

[25] Tatusov RL, Natale DA, Garkavtsev IV, Tatusova TA, Shankavaram UT, Rao BS, Kiryutin B, Galperin MY, Fedorova ND, Koonin EV. The COG database: new developments in phylogenetic classification of proteins from complete genomes. Nucleic Acids Res 2001;29(1):22–8.

[26] Vinogradov DV. Logic programs for quasi-axiomatic theories. NTI series vol. 2. N1-2 1999. p. 61–4.

[27] Ward JJ, Sodhi JS, Buxton BF, Jones DT. Predicting gene ontology annotations from sequence data using Kernel-based Machine Learning Algorithms. In: Proceedings of the 2004 IEEE computational systems bioinformatics conference (CSB 2004); 2004.

[28] Tramontano A, Morea V. Exploiting evolutionary relationships for predicting protein structures. Biotechnology and Bioengineering 2003;84(7):756–62.

[29] Ganter B, Kuznetsov SO. Hypotheses and version spaces. In: de Moor A, Lex W, Ganter B, editors. Proceedings 10th international conference on conceptual structures ICCS'03. Lecture notes in artificial intelligence, vol. 2746; 2003. p. 83–95.

[30] Ganter B, Wille R. Formal Concept Analysis: Mathematical Foundations. Springer; 1999.

[31] Mitchell T. Generalization as Search. Artificial Intelligence 1982;18(2).

[32] Kuznetsov SO, Obiedkov SA. Comparing performance of algorithms for generating concept lattices. J Exp Theor Artif Intell 2002;14(2-):189–216.

[33] Chandra A, Harel D. Horn clause queries and generalizations. J Logic Programming 1985;2(1):1–15.

[34] Smith DK, Xue H. Sequence profiles of immunoglobulin and immunoglobulin-like domains. J Mol Biol 1997;274:530–45.

[35] Delcher AL, Kasif S. Efficient parallel term matching and anti-unification. Logic programming. Cambridge, MA: MIT Press; 1990. p. 355–69.

[36] Arocha JF, Wang D, Patel VL. Identifying reasoning strategies in medical decision making: A methodological guide. J Biomed Inform 2005;38(2):154–71.

[37] Mitchell TM. The need for biases in learning generalizations. Tech. Rep CBM-TR-117. New Brunswick, NJ: Dept of Computer Science, Rutgers Univ; 1980.

[38] Hunter L. Classifying for Prediction: A Multistrategy Approach to Predicting Protein Structure. In: Michalski RS, Tecuci G, editors. Machine learning: a multi-strategy approach, Vol. 4. San Mateo, CA: Morgan Kaufman Publishers; 1993.

[39] Russell SJ. Preliminary steps toward the automation of induction. In: Proceedings of the 5th national conference on artificial intelligence. Los Altos, CA: Morgan Kaufmann; 1986. p. 477–84.

[40] Mitchell TM, Keller RM, Kedar-Cabelli ST. Explanation-based generalization: a unifying view. Machine Learning 1986;1:47–80.

[41] Tan AC, Gilbert D. Machine learning and its application to bioinformatics: an overview. <http://www.utdallas.edu/~yxl059100/MachineLearning-Bioinformatics.pdf/>. Last accessed Sept 17, 2005.

[42] García A, Simari G. Defeasible logic programming: an argumentative approach. Theory and Practice of Logic Programming 2004;4(1):95–138.

[43] Smith RG, Winston HA, Mitchell TM, Buchanan BG. Representation and use of explicit justification for knowledge base refinement. In: Proceedings of the ninth IJCAI. Los Altos: Morgan Kaufmann; 1985. p. 673–80.

[44] [ref de;eted] Kedar-Cabelli ST. Formulating concepts according to purpose. In: sixth AAAI-87, Morgan Kaufmann, Los Altos; 1987. p. 182–6.

[45] Winston PH, Binford TO, Katz B, Lowry M. Learning physical descriptions from functional definitions, examples and precedents. In: second AAAI-83, Morgan Kaufmann, Los Altos; 1983. p. 433–9.

[46] Koehl P, Levitt M. Sequence variations within protein families are linearly related to structural variations. J Mol Biol 2002;323:551–62.

[47] Jennings AJ, Edge CM, Sternberg MJ. Anapproach to improving multiple alignments of protein sequences using predicted secondary structure. Protein Eng 2001;14(4):227–31.

[48] Galitsky B. Biological Applications of Jasmine http://www.dcs.bbk.ac.uk/~galitsky/Jasmine/bio.html (Last downloaded March 2006).

[49] Gregory TR, Hebert PDN. The Modulation of DNA content: proximate causes and ultimate consequences genome research 1999;9(4):317–24.