Multi-label Hierarchical Classification of Protein Functions with Artificial Immune Systems

Roberto T. Alves[1], Myriam R. Delgado[1], and Alex A. Freitas[2]

[1] Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, UFTPR
Av. Sete de Setembro, 3165, CEP: 80230-901, Curitiba – PR – Brazil
roberto.t.alves@gmail.com, myriamdelg@utfpr.edu.br
[2] Computing Laboratory and Centre for BioMedical Informatics, University of Kent,
CT2 7NF, Canterbury, U.K.
A.A.Freitas@kent.ac.uk

Abstract. This work proposes two versions of an Artificial Immune System (AIS) - a relatively recent computational intelligence paradigm – for predicting protein functions described in the Gene Ontology (GO). The GO has functional classes (GO terms) specified in the form of a directed acyclic graph, which leads to a very challenging multi-label hierarchical classification problem where a protein can be assigned multiple classes (functions, GO terms) across several levels of the GO's term hierarchy. Hence, the proposed approach, called MHC-AIS (Multi-label Hierarchical Classification with an Artificial Immune System), is a sophisticated classification algorithm tailored to both multi-label and hierarchical classification. The first version of the MHC-AIS builds a global classifier to predict all classes in the application domain, whilst the second version builds a local classifier to predict each class. In both versions of the MHC-AIS the classifier is expressed as a set of IF-THEN classification rules, which have the advantage of representing comprehensible knowledge to biologist users. The two MHC-AIS versions are evaluated on a dataset of DNA-binding and ATPase proteins.

Keywords: Artificial Immune System, Hierarchical and Multi-label Classification, Prediction of Protein Function.

1 Introduction

Artificial Immune Systems (AIS) are one of the most recent natural computing approaches to emerge from computer science. The immune system is a distributed system, capable of constructing and maintaining a dynamical and structural identity, learning to identify previously unseen invaders and remembering what it has learnt. These computational techniques have many potential applications, such as in distributed and adaptive control, machine learning, pattern recognition, fault and anomaly detection, computer security, optimization, and distributed system design [1].

In data mining, ideally the discovered knowledge should be not only accurate, but also comprehensible to the user [2]. This work addresses the multi-label hierarchical classification task of data mining, where the goal is to discover a classification model

A.L.C. Bazzan, M. Craven, and N.F. Martins (Eds.): BSB 2008, LNBI 5167, pp. 1–12, 2008.

that predicts more than one class for an example (data instance) across several levels of a class hierarchy, based on the values of the predictor attributes for that example.

Bioinformatics is an inter-disciplinary field, involving the areas of computer science, mathematics, biology, etc. [3]. Among many bioinformatics problems, this paper focuses on the prediction of protein functions from information associated with the protein's primary sequence. As proteins often have multiple functions which are described hierarchically, the use of multi-label hierarchical techniques for the induction of classification models in Bioinformatics is a promising research area. At present, the biological functions that can be performed by proteins are defined in a structured, standardized dictionary of terms called the Gene Ontology (GO) [4].

The AIS algorithms proposed in this paper combine the adaptive global search of the AIS paradigm with advanced concepts and methods of data mining (hierarchical and multi-label classification), in order to solve a challenging bioinformatics problem (protein function prediction – assigns GO terms (classes) to proteins). The AIS presented in this paper discovers knowledge interpretable by the user, in the form of IF-THEN classification rules, unlike other methods proposed in the literature, whose classification model is typically a "black box" which normally does not provide any insight to the user about interesting hidden relationships in the data [5].

2 Multi-label Hierarchical Classification

The classification task of data mining [2] consists of building, in a training phase, a classification model that maps each example t_i to a class $c \in \mathcal{C}$ of the target application domain, with $i = 1, 2, ..., n$, where n represents the number of examples in the training set.

The majority of classification algorithms cope with problems where each example t_i is associated with a single class $c \in \mathcal{C}$. These algorithms are called single label. However, some classification problems are considerably more complex because each example t_i is associated with a subset of classes $C \in \mathcal{C}$ of the application domain. Protein function prediction is a typical case of this type of problem, since a protein can perform several biological functions. Algorithms for coping with this kind of problem are called multi-label [6].

There has been a very large amount of research on conventional “flat” (non-hierarchical) classification problems, where the classes to be predicted are not hierarchically organized. However, in some problems the classes are hierarchically organized, which makes the classification problem much more challenging. Problems of this type are known as hierarchical classification problems [7].

In hierarchical classification problems, typically the classes are hierarchically organized in one of the following two forms: as a tree (where each class has at most one parent class) or as a direct acyclic graph (DAG), where each class can have more than one parent. In bioinformatics, two of the most well-known hierarchical structures for classifying protein functions are the enzyme commission hierarchy [8] – organized in the form of a tree and GO [4] – organized in the form of a DAG. The GO consists of a dictionary that defines gene products independent from species. GO actually consists of 3 separate "domains" (very different types of GO terms): molecular function, biological process and cellular component. The GO is structurally organized

in the form of a direct acyclic graph (DAG), where each GO term represents a node of the hierarchical structure.

In hierarchical classification, there are basically two types of classifiers that can be built to cope with the full set of classes to be predicted: local or global classifiers. In local classifiers, for each class $c \in \mathcal{C}$ a (local) classifier is built to predict whether or not each class c is associated with an example t_i. After all classifiers are built, an example t_i is submitted to all those classifiers (one for each class) in order to determine which classes are predicted for that example. In global classifiers, a single (global) classifier is built to discriminate among all classes of the application domain and so t_i is submitted to a single (potentially very complex) classifier [7].

3 Multi-label Hierarchical Classification with an Artificial Immune System

The immune system as a biological complex adaptive system has provided inspiration for a range of innovative problem solving techniques, including classification tasks [9] In this paper, the construction of a immune-based learning algorithm is explored whose recognition, distributed, and adaptive nature offer many potential advantages over more traditional models. The AIS algorithm used in this paper is called MHC-AIS (Multi-label Hierarchical Classification with an Artificial Immune System). MHC-AIS is based on the following natural immunology principles: clonal selection, immune network and somatic hypermutation [10,11]. In AIS, antibodies (*ab*) represent candidate solutions to the target problem, whilst antigens (*ag*) represent specific instances of the problem. In the context of this work, *ab´*s represent IF-THEN classification rules and *ag´*s represent proteins in the training set whose classes have to be predicted by the AIS.

In essence, in the clonal selection theory antibodies are cloned in proportion to their degree of matching ("affinity") to antigens, so that the antibodies which are better in recognizing antigens produce more clones of themselves. The just-generated clones are then subject to a process of somatic hypermutation, where the rate of mutation applied to a clone is inversely proportional to its affinity with the antigens. In computer science terms, the best antibodies are cloned more often and undergo a smaller rate of mutation (have fewer parts of their candidate solution modified) than the worst antibodies. With time this process of clonal selection and hypersomatic mutation leads to better and better candidate solutions to the target problem.

In essence, the theoretical immunology principle of immune networks states that antibodies can recognize not only antigens but also other antibodies. The first kind of recognition stimulates antibody production, but the latter suppresses antibodies, which in computer science terms mean a candidate solution tends to suppress other similar candidate solutions, which has the effect of improving the diversity of the search for a (near-)optimal candidate solution.

The training phase MHC-AIS is performed by two major procedures, called Sequential Covering (SC) and Rule Evolution (RE) procedures. The SC procedure iteratively calls the RE procedure until (almost) all “antigens” (proteins, examples) are covered by the discovered rules. The RE procedure essentially evolves artificial “antibodies” (IF-THEN classification rules) that are used to classify antigens. Then,

the best evolved antibody is added to discovered rule set. Each antibody (candidate classification rule) consists of two parts: the rule antecedent (IF part), represented by a vector of conditions (attribute-value pairs), and the rule consequent (THEN part) that represents the classes predicted by the rule. In this work the classes correspond to GO terms denoting protein functions. This work proposes two versions of the MHC-AIS, viz.: local and global versions (more details in the following subsections).

3.1 Global Version

In biological databases a protein is annotated only with its most specific GO term. Given the semantics of the GO's functional hierarchy, this implicitly means the protein also contains all the functional classes of its ancestral GO terms in the GO's DAG. Hence, in a data preprocessing step, MHC-AIS explicitly assigns to each antigen (protein) both its most specific class(es) (GO term(s)) and all its ancestral classes. MHC-AIS also considers the semantics of the GO's functional hierarchy when creating classification rules – i.e., it guarantees that, if a rule predicts a given GO term, all its ancestral GO terms are also predicted by the rule.

Fig. 1 shows the high-level pseudocode of the SC procedure.

```
Input: full protein training set;
Output: set of discovered rules;
DiscoveredRuleSet = ∅;
TrainSet = {set of all protein training examples};
Re-label TrainSet regarding GO's functional class hierarchy;
WHILE |TrainSet| > MaxUncovExamp
       BestRule = RULE-EVOLUTION(TrainSet); //based on AIS
       DiscoveredRuleSet = DiscoveredRuleSet ∪ BestRule;
       updateCoveredClasses(TrainSet, BestRule)
       removeExamplesWithAllClassesCovered(TrainSet);
END WHLE
```

Fig. 1. Sequential Covering (SC) procedure

First, it initializes the set of discovered rules with the empty set and initializes the training set with the set of all original training examples. Next, each example in the training set is extended to contain both the original class and all its ancestral classes in the GO hierarchy. Thereafter, the algorithm starts a WHILE loop which, at each iteration, calls the RE procedure. The latter receives, as parameters, the current training set and use AIS algorithm to discover classification rules. The RE procedure returns the best classification rule discovered by the AIS for the current training set. Then the SC procedure adds that rule to the discovered rule set and removes the training examples covered by that rule, as follows. In conventional rule induction algorithms for single-label classification, examples correctly covered by the just discovered rule are removed from the training set. However, in multi-label classification this process is more complex, since different rules and different training examples have different numbers of classes. In the global version of the AIS, the process of example removal works as follows. First, the training examples covered by the just-discovered rule (i.e. examples satisfying the rule's antecedent) are identified.

For each of those examples, its annotated (true) classes which are predicted by the just-discovered rules are marked as covered. As more and more rules are discovered, more and more of the annotated classes of each example will be covered. Only when all the classes of an example are covered that example is removed from the training set. The process of rule discovery terminates when the number of examples in the current training set becomes smaller than a user-defined parameter called *MaxUncovExamp*. Such procedure avoids the discovery of rules covering too few examples, unlikely to generalize well to the test set [12].

Fig. 2 shows the high-level code of the RE procedure, where rules are obtained by the proposed MHC-AIS. First, the initial population of antibodies $AB_{t=0}$ is created, where the consequent of each rule contains (initially) all GO classes in the data being mined. At the end of the evolutionary process, the AIS updates the consequent of the discovered rule (to be returned by the RE procedure) to contain only a subset of classes, representing the classes predicted by that rule, as will be explained later.

```
Input: current TrainSet;   Output: the best evolved rule;
AB_{t=0} = Create initial population of antibodies at random;
Computefitness(AB_{t=0},TrainSet);
FOR t = 1 to Number of Generations
        CL = ProduceClones(AB_{t-1});
        CL* = MutateClones(CL);
        AB_t = AB_{t-1} ∪ CL*;
        Computefitness(AB_t,TrainSet);
        Suppresion(AB_t);
        Elitism(AB_{t+1});
END FOR t;
Determine the final subset of classes of the best antibody found so far;
return(best antibody);
```

Fig. 2. Rule Evolution (RE) procedure

After its creation, the fitness (quality measure) of each antibody $ab_i^{t=0}$ of the initial population is calculated on the training set, where each example represents an antigen ag_j. The fitness of each ab_i is computed in two stages. First, a fitness value is associated with each kth-class c_k^i contained in the consequent of rule (antibody) ab_i. The value of this fitness is computed according to the following equation:

$$fit\left(c_k^i\right) = \frac{TP_{c_k^i}}{TP_{c_k^i} + FN_{c_k^i}} \times \frac{TN_{c_k^i}}{TN_{c_k^i} + FP_{c_k^i}} \tag{1}$$

where:

- TP (true positives) = number of training examples satisfying affinity(ab_i,ag_j) $\geq \delta_{AF}$ and having the annotated class c_k^i.
- TN (true negatives) = number of training examples satisfying affinity(ab_i,ag_j) $< \delta_{AF}$ and not having the annotated class c_k^i.
- FP (false positives) = number of training examples satisfying affinity (ab_i,ag_j) $\geq \delta_{AF}$ and not having the annotated class c_k^i.
- FN (false negatives) = number of training examples satisfying affinity (ab_i,ag_j) $< \delta_{AF}$ and having the annotated class c_k^i.

The function affinity (ab_i,ag_j) returns the degree of matching between the rule ab_i and the training example ag_j. The value of the parameter δ_{AF} represents the minimum degree of matching required for the antigen ag_j to be deemed as classified by the rule ab_i. It is important to note that δ_{AF} is a user-specified parameter, which gives more flexibility to the use of the algorithm, allowing the use of a partial or total degree of matching (δ_{AF} = 1.0) in the classification process. MHC-AIS is a hierarchical classification algorithm, and so it must consider the hierarchical structure of classes in the classification process, to reduce classification errors. A common hierarchical classification error occurs when a classifier correctly predicts a given class c for an example but does not predict an ancestral class of c. Recall that all the ancestral classes of a given predicted class must also be predicted by the trained classifier, due to the semantics of the class hierarchy in the GO. Some hierarchical classification algorithms try to correct hierarchical classification errors after the classifier has been built, in a post-processing phase. By contrast, MHC-AIS maintains a set of consistent hierarchical classifications during the construction of the global classifier. This kind of consistency is given by equation (2):

$$fit\left(c_{k*}^{i}\right) = \max\left[\, fit\left(c_{k*}^{i}\right), fit\left(c_{k}^{i}\right)\right], \quad c_{k*}^{i} \in Ancestors(c_{k}^{i}) \tag{2}$$

Hence, if the fitness of some ancestral class $c_{k*}^{\ i}$ is smaller than the fitness of its descendant class $c_k^{\ i}$, then the fitness of $c_k^{\ i}$ is assigned to its ancestral class, therefore maintaining the consistency of hierarchical classifications during training.

The fitness of an entire rule (computed as an aggregated value of the fitness of all the classes predicted by the rule) is calculated by equation (3):

$$fitness\left(ab_i\right) = \frac{1}{n} fit\left(c_{k}^{i}\right), \quad fit\left(c_{k}^{i}\right) > \delta_{FT} \tag{3}$$

where n indicates the number of classes $c_i^{\ k}$ with fitness greater than the value of the parameter δ_{FT}.

Next, the AIS starts to evolve the population of antibodies. Once the global fitness of the entire rule has been calculated for each ab_i, the algorithm executes the clonal expansion process, typical in AIS [1]. Each ab_i produces *NumCl* clones of itself, where *NumCl* is proportional to the fitness of ab_i. The number of clones to be produced for each ab_i is determined by equation (4):

$$NumCl_i = \text{int}\left(fitness\left(ab_i\right) \times NumMaxCl \times ClRate\right) \tag{4}$$

where the value of $NumCl \in [1, NumMaxCl]$. The parameter *NumMaxCl* represents the maximum number of clones that can be generated for a given *ab*. The function *int* truncates the fractional part of its parameter. The *ClRate* is calculated in every iteration with the goal of controlling the size of the antibody population, stimulating or inhibiting the production of clones. The value of *ClRate* is given by equation (5):

$$ClRate = \begin{cases} HyperClRate & \text{if } |AB| < nIP \\ 0 & \text{if } |AB| > nMaxP \\ 1 - \left(\dfrac{|AB| - nIP}{nMaxP - nIP} \right) & \text{otherwise} \end{cases} \quad (5)$$

where *HyperClRate*, *nIP* and *nMaxP* are specified in the beginning of the execution of the algorithm and indicate, respectively, clonal hyper-expansion rate, initial antibody population size and maximum antibody population size. It is important to emphasize that the parameter *nMaxP* does not represent the maximum size that the antibody population *AB* can take during the evolution. Rather, it indicates that, if the size of *AB* is greater than the value of that parameter, the generation of clones proportional to antibody fitness is turned off. Next, the population *CL* of clones undergoes a process of somatic hypermutation just on the IF part of the rule. A mutation rate applied to each clone *cl* is inversely proportional to the fitness of the antibody *ab* from which the clone was produced. The mutation rate is determined by equation (6):

$$MutRate_{cl} = mutMin + (mutMax - mutMin) \times (1 - fitness(cl)) \quad (6)$$

where *MutMin* and *MutMax* indicate, respectively, the minimum and maximum mutation rates to be applied to a clone *cl*; and the function fitness(*cl*) is presented in equation (3). The *MutRate* represents the probability that each gene (rule condition – IF antecedent) will undergo mutation. The population CL^*, which is formed only by clones that underwent some mutation, is then inserted in *AB*. Other procedures are also applied to *AB* during the rule evolution procedure: suppression of antibodies and elitism. The suppression procedure, characteristic of AIs based on the immune network theory, removes from AB_t similar antibodies. More precisely, if two antibodies ab_i and ab_i^* have a similarity degree greater than or equal to the value of δ_{SIM}, then, out of those two antibodies, the one with the smallest fitness is removed. The degree of similarity between two antibodies is computed as the number of conditions (attribute-value pairs) in the rule antecedents of both antibodies divided by the number of conditions in the rule antecedent of the antibody with the greatest number of conditions – which produces a measure of antibody similarity normalized in the range from 0 (no rule conditions in common) to 1 (identical rule antecedents). Elitism, a mechanism quite common in evolutionary algorithms [13], selects the antibody with the best fitness to be included in the next-iteration population AB_{t+1}.

During the rule evolution procedure all the classes occurring in the data being mined are represented in the consequent. The choice of the final subset of classes to be assigned to the consequent of the best discovered rule is given by equation (7):

$$PC = \bigcup c_k \in C \mid fit(c_k) > \delta_{FT} \quad (7)$$

where PC represents the set of classes predicted by the best discovered rule whose fitness value is greater than δ_{FT}.

3.2 Local Version

Like the global MHC-AIS, the local MHC-AIS consists of the SC (Fig. 3) and RE procedures, but with some differences. In the local version, the SC procedure labels the training examples as positive or negative. Positive examples represent examples associated with the class of the current node of the GO's DAG (a classifier is trained for each node of the GO's DAG), denoted class Y, whilst examples that do not have the class Y are labeled as negative examples. MHC-AIS is an algorithm for constructing hierarchical classifiers, and therefore the hierarchical structure has to be coped with like in the global version. Hence, all training examples labeled with any descendant class or ancestor class of the current class Y are labeled as positive class. Concerning the latter type of positive examples, it is often the case that, when a hierarchical classifier is being built, examples annotated with an ancestor class of the current class Y are removed, since they are considered as ambiguous – they do not have an annotation suggesting that they have class Y, but maybe they actually have class Y, which was not annotated yet simply due to the lack of evidence for its presence (note that "absence of evidence is different from evidence of absence"). However, in this work we use examples with an annotated class that is an ancestral of the current class Y in order to increase the number of positive examples and so hopefully increase the predictive accuracy of the algorithm.

```
Input: full training set; Output: set of discovered rules;
DiscoveredRuleSet = ∅;
FOR EACH class c
       TrainSet = {set of all training examples};
       WHILE |TrainSet| > MaxUncovExamp
            BestRule = RULE-EVOLUTION(TrainSet, class c);//based on AIS
           DiscoveredRuleSet=DiscoveredRuleSet ∪ BestRule;
          TrainSet = TrainSet – {examp. correctly covered by BestRule};
       END WHILE;
END FOR EACH class;
```

Fig. 3. Sequential Covering (SC) procedure for Local Version

In this local version, MHC-AIS first discovers as many classification rules as necessary in order to cover the positive examples. Next, the algorithm discovers as many rules as necessary to cover the negative examples. Every time that a given rule is discovered, all the examples correctly covered by that rule (i.e. examples satisfying the conditions in the rule antecedent and having the class predicted by the rule consequent) are removed from the current training set, as usual in rule induction algorithms. This iterative process of rule discovery and removal of training examples is repeated until the number of examples in the current training set becomes smaller than a user-defined threshold *MaxUncovExamp*.

The other procedures of the local MHC-AIS are the same as in the global version of the algorithm, described in the previous subsection.

4 Computational Results

The two versions of the MHC-AIS were evaluated on a dataset of proteins created from information extracted from the well-known UNIPROT database [14]. This dataset contains two protein families: DNA-binding and ATPase [15]. These two protein families were chosen for our experiments because there are many proteins that belong to both families, increasing the difficult of the problem of building a multi-label classifier. The dataset used in the experiments contains 7877 proteins, where each protein (example) is described by 40 predictor attributes, 38 of which are PROSITE[1] patterns and 2 of which are continuous attributes (molecular weight and the number of amino acids in the primary sequence). In total, the dataset contains 214 classes (GO terms) to be predicted.

As previously discussed, in data mining the discovered knowledge should be not only accurate, but also comprehensible to the user [2,5]. In this spirit, the results can be evaluated according to two criteria, viz. the predictive accuracy and simplicity of the discovered rule set. In this paper, the predictive accuracy is evaluated by the F-measure (adapted to the scenario of multi-label hierarchical classification), which involves computing the precision and recall of the discovered rule set on the test set (unseen during training). Interpretability will be measured in terms of the size of the discovered rule set, an approach which is not ideal but is still used in the literature.

In the global version, the set of GO terms predicted for a test example t, denoted *PredGO(t)*, consists of the union of all GO terms in the consequent of all rules covering t – i.e. all rules whose conditions are satisfied by t's attribute values.

In the local version of MHC-AIS, each test example t is submitted to the n trained classifiers. Each classifier consists of a set of discovered rules. The class predicted by each classifier is the class represented in the consequent of the rule with the greatest fitness value (computed during training) out of all rules discovered by that classifier that cover the example t. If no discovered rule covers the example t, the latter is classified by the default rule, which predicts the majority class in the training set. Hence, *PredGO(t)* consists of all GO terms whose trained classifiers predicted their corresponding positive class for the example t.

MHC-AIS computes the Precision and Recall for a test example t – denoted *P(t)* and *R(t)*, respectively – as per equations (8) and (9), where *TrueGO(t)* is the set of true GO terms for test example t.

$$P(t) = |PredGO(t) \cap TrueGO(t)| \,/\, PredGO(t) \tag{8}$$

$$R(t) = |PredGO(t) \cap TrueGO(t)| \,/\, TrueGO(t) \tag{9}$$

Thus, precision is the proportion of true classes among all predicted classes, whilst recall is the proportion of predicted classes among all true classes. The F-measure for a test example t is given by equation (10), the harmonic mean of P and R.

$$F(t) = (2 \times P(t) \times R(t)) \,/\, (1 + P(t) + R(t)) \tag{10}$$

[1] PROSITE patterns are motifs well-known in bioinformatics [16] and they are represented as binary attributes – i.e., each attribute indicates whether or not the corresponding PROSITE pattern occurs in the sequence of amino acids of a protein.

Finally, once *P(t)* and *R(t)* have been computed for each test example *t*, the system computes the overall F-measure over the entire test set **T** by equation (11), where |**T**| denotes the cardinality of the test set **T**.

$$\text{Predictive Accuracy} = F(\mathbf{T}) = (\Sigma_{t \in \mathbf{T}} F(t)) / |\mathbf{T}| \quad (11)$$

Table 1 shows the predictive accuracy for precision, recall and F-measure for global and local version. The numbers after the "±" symbol represent the standard deviations associated with a well-known 10-fold cross-validation procedure [2]. In the columns F-measure, the best result (out of both version of MHC-AIS) is shown in bold. The results presented in Table 1 consider different affinity (matching) thresholds for both versions of MHC-AIS, to evaluate the predictive performance of the algorithms using partial matching ($\delta_{AF} < 1.0$) or total matching ($\delta_{AF} = 1.0$).

Table 1. Predictive accuracy (%) of MHC-AIS versions on the used protein data set

Affinity	Global Version			Local Version		
Threshold	Precision	Recall	F-Measure	Precision	Recall	F-Measure
0.8	45.93±2.71	98.23±0.61	**58.35±2.23**	80.58±1.01	44.65±1.59	55.65±1.45
0.9	50.79±3.18	92.86±3.76	58.34±2.86	75.61±1.12	52.57±2.35	**59.75±1.77**
1.0	28.91±1.31	99.50±0.12	42.84±1.37	58.56±1.01	69.91±1.13	**61.37±0.82**

Table 1 shows that the global MHC-AIS performed worst (according to the F-measure) when using total matching. Note that the global MHC-AIS obtained the worst results for the precision measure with all affinity threshold values. By contrast, the global MHC-AIS obtained very good recall values with all affinity thresholds. This performance behavior of global MHC-AIS indicates that the trained global classifier has a bias favoring the prediction of a large number of classes, mainly because the set of classes predicted for a test example consists of the union of all classes in the consequents of all rules covering that example - regardless of the fitness of the individual rules in question and the fact that the predictions of some of those rules might be inconsistent with each other. This tends to predict more classes than the actual number of true classes for a given test example, which tends to increase recall but reduce precision (given the definition of these terms).

In both cases of MHC-AIS, as the value of the affinity threshold δ_{AF} increases the value of precision is reduced, showing a disadvantage in the use of total matching. As expected, due to the trade-off between precision and recall, the local version of the algorithm had the opposite performance behavior in the case of recall, where the largest value was obtained with total matching.

Table 2. Simplicity of the discovered rule set of MHC-AIS versions

Threshold	Global Version		Local Version	
Affinity	#rules	#Conditions	#rules	#Conditions
0.8	**63.90±1,59**	**1164,30±28.20**	788.00±3.68	2901.30±42.83
0.9	**58.09±3.08**	**1066.60±53.39**	1016.80 ±8.09	4829.80±67.44
1.0	**79.90±1.83**	**1361.00±41.16**	1232.90±16.07	7069.53±18298

Table 2 shows the results of both local and global versions of MHC-AIS with respect to the simplicity (interpretability) of the discovered rule set. This simplicity was measured by the number of discovered rules and total number of rule conditions (in all rules). The averages were computed over 10-fold cross-validation.

Note that, as shown in Table 2, the global MHC-AIS obtained much better results concerning rule set simplicity than the local MHC-AIS, in all experiments. This advantage of the global MHC-AIS is probably due to the fact that, by building a single set of rules predicting all classes in a single run of the algorithm, the algorithm can avoid the need for discovering redundant rules covering the same set of true classes for some examples. In particular, when the local version discovers rules predicting the "negative" class at each node of the GO's DAG, it should be noted that those rules predicting the negative class tend to be redundant with respect to rules predicting positive classes in other nodes of the GO's DAG, since some of the negative class examples for a given GO node will inevitably be positive class examples in another GO node. An example of a rule discovered rule by global MHC-AIS in the used data set is presented below:

IF (PS00676 == 1) and (PS00390 == 1) and (MOLECULAR_WEIGHT < 29353)
then (5488, 5515, 51087)

The biological interpretation of this rule is: if a protein presents "Sigma-54 interaction domain signatures and profile" and "Sodium and potassium ATPases beta subunits signatures" signatures and "molecular weight is less than 29353" then the predicted classes (biological functions) are: "binding" (5488) and "protein binding" (5515) and "chaperone binding" (51087). Note that the GO hierarchy was considered, i.e. the true hierarchical path is 5488 → 5515 → 51087 (from shallower to deeper nodes).

5 Conclusion and Future Work

This work described an artificial immune system (AIS)-based rule induction algorithm to the prediction of protein function. The paper proposed two versions of the AIS algorithm, a global version, where a single global classifier is built predicting all classes of the application domain; and a local version, where a local classifier is built for each node of the GO class hierarchy. Both versions have the advantage of discovering IF-THEN classification rules, constituting a type of knowledge representation that can, in principle, be easily interpretable by biologist users. The global and local versions of the AIS have different (roughly dual) advantages and disadvantages with respect to predictive accuracy, but the global version at least has the advantage of discovering much simpler (smaller) rule sets.

Future work involves: (a) comparing the predictive performance of both versions of the AIS with other classification algorithms designed for hierarchical classification (e.g. [17]); (b) investigating new criteria for selecting, out of all classes in the consequent of the rules covering a test example in the global approach, which classes should be actually predicted for the test example; (c) incorporating an explicit mechanism during the training phase to improve the rules´ interpretability (d) analyzing the biological relevance of the discovered rules; and (e) evaluating the proposed AIS in datasets of other protein families and other types of predictor attributes.

References

1. De Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer, Berlin (2002)
2. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Mateo (2005)
3. Fogel, G.B., Corne, D.W.: Evolutionary Computation in Bioinformatics. Morgan Kaufmann Publishers, San Franciso (2003)
4. The Gene Ontology Consortium. The Gene Ontology (GO) Database and Informatics Resource. Nucleic Acids Research 32(1), 258–261 (2004)
5. Freitas, A.A.: Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer, Berlin (2002)
6. Tsoumakas, G., Katakis, I.: Multi-Label Classification: An Overview. International Journal of Data Warehousing and Mining 3(3), 1–13 (2007)
7. Sun, A., Lim, E.-P., Ng, W.-K.: Performance Measurement Framework for Hierarchical Text Classification. Journal of the American Society for Information Science and Technology 54(11), 1014–1028 (2003)
8. E. Nomenclature, of the IUPAC-IUB. American Elsevier Pub. Co., New York, NY 104 (1972)
9. Freitas, A.A., Timmis, T.: Revisiting the foundations of artificial immune systems for data mining. IEEE Trans. on Evolutionary Computation 11(4), 521–540 (2007)
10. Ada, G.L., Nossal, G.V.: The Clonal Selection Theory. Scientific American 257, 50–57 (1987)
11. Jerne, N.K.: Towards a Network Theory of Immune System. Ann. Immunol (Inst. Pasteur) 125C, 373–389 (1974)
12. Alves, R.T., Delgado, M.R., Lopes, H.S., Freitas, A.A.: An artificial immune system for fuzzy-rule induction in data mining. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 1011–1020. Springer, Heidelberg (2004)
13. Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley, Reading (1989)
14. The UniProt Consortium. The Universal Protein Resource (UniProt). Nucleic Acids Res. 35, D193–D197 (2007)
15. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Water, P.: Molecular Biology of the Cell, 4th edn. Garland Science, New York (2002)
16. Hulo, N., Bairoch, A., Bulliard, V., Cerutti, L., De Castro, E., Langendijk-Genevaux, P.S., Pagni, M., Sigrist, C.J.A.: The PROSITE Database. Nucleic Acids Res. 34, D227–D230 (2006)
17. Wolstencroft, K., Lord, P.W., Tabernero, P., Brass, P., Stevens, R.: Protein classification using ontology classification. Bioinformatics 22, 530–538 (2006)