ACO-Based Bayesian Network Ensembles for the Hierarchical Classification of Ageing-Related Proteins

Khalid M. Salama and Alex A. Freitas

School of Computing, University of Kent, UK
{kms39,A.A.Freitas}@kent.ac.uk

Abstract. The task of predicting protein functions using computational techniques is a major research area in the field of bioinformatics. Casting the task into a classification problem makes it challenging, since the classes (functions) to be predicted are hierarchically related, and a protein can have more than one function. One approach is to produce a set of local classifiers; each is responsible for discriminating between a subset of the classes in a certain level of the hierarchy. In this paper we tackle the hierarchical classification problem in a local fashion, by learning an ensemble of Bayesian network classifiers for each class in the hierarchy and combining their outputs with four alternative methods: a) selecting the best classifier, b) majority voting, c) weighted voting, and d) constructing a meta-classifier. The ensemble is built using ABC-Miner, our recently introduced Ant-based Bayesian Classification algorithm. We use different types of protein representations to learn different classification models. We empirically evaluate our proposed methods on an ageing-related protein dataset created for this research.

Keywords: Ant Colony Optimization, Data Mining, Bayesian Networks, Hierarchical Classification, Protein Function Prediction.

1 Introduction

Data mining, a research focus in the fields of artificial intelligence, machine learning and statistics, is the process of discovering accurate, comprehensible and useful patterns in real-world datasets. Classification is one of the widely studied data mining tasks, in which the aim is to learn a model used to predict the class of unlabelled cases [22]. Many real-world classification problems have their classes organized into a hierarchy – typically a tree or a Directed Acyclic Graph (DAG), which makes the hierarchical classification problem a challenging research topic. A commonly used approach to tackle such a problem is the local approach, where the class hierarchy is processed in a top-down fashion, producing one or more local classifiers for each class level and combining their outputs. Each classifier is trained with a flat classification algorithm using a local data subset, and it discriminates among a subset of classes in the hierarchy.

L. Vanneschi, W.S. Bush, and M. Giacobini (Eds.): EvoBIO 2013, LNCS 7833, pp. 80–91, 2013.

Protein function prediction is an important application of hierarchical classification, and is considered as one of the major types of bioinformatics problems [7]. Determining protein functions is crucial for improving biological knowledge, diagnosis and treatment of diseases. In this work, we focus on human ageing-related proteins, predicting their biological processes according to the Gene Ontology. Research on ageing is important because ageing is the greatest risk factor for a number of diseases.

ABC-Miner [14], recently introduced by the authors, is a flat classification algorithm that learns the structure of a Bayesian Augmented Naïve-Bayes (BAN) network using Ant Colony Optimization (ACO)– a meta-heuristic global search for solving combinatorial optimization problems [6].

In this paper we approach the hierarchical protein function prediction in a local fashion, using the ABC-Miner algorithm. We build an ensemble of classifiers for each node in the class hierarchy using the same algorithm but with four different types of protein representations, which were the representation types used in [18]. Each representation consists of a predefined set of protein features of the same type. There are many types of protein representations, and the choice of the feature representation might be as important as the choice of the classification algorithm. The issue of using different representations is related to the well-known issue of feature selection in data mining, where a subset of the features is selected during the run of the algorithm to build a classifier. However, we favour the former, mainly because it significantly reduces computational time. This is important in our local hierarchical classification problem, where we have to build a classifier at each of the large number of class hierarchy nodes.

The combination of an ensemble's outputs at each class node is performed by four alternative methods: a) selecting the best classifier, b) majority voting, c) weighted voting, and d) constructing a meta-classifier to perform the final prediction. From one perspective, we compare the use of each protein representation individually to the use of the various proposed ensemble methods to combine representations. From another perspective, we compare the use of our ant-based algorithm – on each protein representation individually and with the various ensemble methods – to other well-known Bayesian classification algorithms, namely: Naïve-Bayes, TAN, and GBN. We evaluate the classification performance of our ensemble settings on a new dataset of ageing-related proteins, which was created for this research – due to the lack of ageing datasets for hierarchical classification in the literature.

The rest of the paper is organized as follows. Section 2 gives a brief overview on Bayesian network classifiers. Section 3 describes our Ant-based Bayesian Classification Algorithm. Section 4 discusses hierarchical classification approaches. Section 5 describes our proposed ensemble-of-classifier methods based on different protein representations for hierarchical classification. Section 6 describes the creation of our ageing-related dataset. Experimental results are shown in Section 7, followed by the conclusion and future research directions in Section 8.

2 Bayesian Network Classifiers

In the context of reasoning with uncertainty, Bayesian networks (BN) is one of the most powerful tools that model (in)dependence relationships between variables [5]. A directed acyclic graph (DAG) is used to represent the variables as nodes and statistical dependencies between the variables as edges between the nodes. In addition, a set of conditional probability tables (network parameters), one for each variable, is obtained by computing the probability distribution of the variable given its parents. Note that a Bayesian network should be able to answer probabilistic queries about any node(s) in the network.

Bayesian network classifiers are a special kind of probabilistic networks, where the concern is to answer queries about the probability of a specific node: the class attribute. Thus, the class node is treated as a special variable in the network; it is set as the parent of all other variables. The purpose is to compute the probability of each value c in the class variable C given a case $\mathbf{x}$ (an instance of the input attributes $\mathbf{X} = \{X_1, X_2, ..., X_n\}$), then label the case with the class having the highest probability, as in the following formulas:

$$C(\mathbf{x}) = \arg\max_{\forall c \in C} P(C = c | \mathbf{x} = x_1, x_2, ..., x_n), \tag{1}$$

letting $\mathbf{Pa}(X_i)$ be the set of parent predictor variables of X_i in the network, according to the Bayes' Theorem:

$$\overbrace{P(C = c | \mathbf{x} = x_1, x_2, ..., x_n)}^{\text{posterior probability}} \alpha \overbrace{P(C = c)}^{\text{prior probability}} \prod_{i=1}^{n} \overbrace{P(x_i | \mathbf{Pa}(X_i), C = c)}^{\text{likelihood}} \tag{2}$$

There are several types of Bayesian network classifiers studied in the literature: Naïve-Bayes, Tree Augmented Naïve-Bayes (TAN), Bayesian network Augmented Naïve-Bayes (BAN), and General Bayesian networks (GBN). Naïve-Bayes is the simplest Bayesian classifier in the literature, and it assumes the attributes are independent given the class label [3] – i.e., each feature node has just one parent node in the network: the class variable.

Since the independency assumption in general is not realistic, extended versions were developed to improve the performance of Naïve-Bayes. TAN allows a node in a BN to have more than one parent, besides the class variable. This produces a tree-like BN structure. In BAN classifiers, no restrictions (or at most k-dependencies) are enforced on the number of parents that a node in the network can depend on. Unlike the other BN classifier learners, a GBN learner treats the class variable node as an ordinary node during network construction. The idea is to build a general purpose BN, extract the *Markov blanket* of the class node and use the resulting network as a Bayesian classifier. For a review and comparison of various BN classifiers, see [3,9].

Learning a BN (classifier) from a dataset $\mathbf{D}$ is decomposed into two phases; learning the network structure and learning the network parameters. Parameter learning can be done in a relatively straightforward way by computating a

conditional probability table (CPT) for each variable with respect to its parent variables. The CPT of variable X_i encodes the likelihood of each value of this variable given each combination of values of $\mathbf{Pa}(X_i)$ in the network structure G, and the likelihood of the dataset $\mathbf{D}$ given a network G is denoted by $P(\mathbf{D}|G)$. Typically, the purpose is to find G that maximizes $P(\mathbf{D}|G)$ for a given $\mathbf{D}$, which is the role of BN structure learning. A common approach to this problem is to introduce a scoring function, f, that evaluates each G with respect to $\mathbf{D}$, searching for the best network structure according to this score [3].

Most algorithms used in the literature for building such BN classifiers are deterministic and greedy, and so are likely to get trapped into local optima in the search space. Since learning the optimal BN structure from a dataset is $\mathcal{NP}$-complete, stochastic meta-heuristic global search methods like ACO, which are less likely to get trapped into local optima, can be applied to build high-quality BN classifiers in an acceptable computational time.

3 The ABC-Miner Algorithm

Ant Colony Optimization (ACO) [6] is a meta-heuristic for solving combinatorial optimization problems, inspired by observations of the behavior of ant colonies in nature. The main idea is to utilize a swarm of simple individuals that use collective behaviour to achieve a certain goal. ACO algorithms have been successful in solving several combinatorial optimization problems, including classification rules discovery [12,11] and general purpose BN construction [2,13,23]. However, ABC-Miner [14], recently introduced by the authors, is the first ACO algorithm to learn BN classifiers.

In ABC-Miner the decision components in the construction graph (which define the search space that an ant uses to construct a candidate solution) are all the edges of the form $X \rightarrow Y$ where $X \neq Y$ and X, Y belong to the set of predictor attributes. These edges represent the attribute dependencies in a constructed BN classifier. Each ant in the colony creates a candidate solution (BN classifier). Then the quality of that solution is evaluated. The best solution produced in the colony at the current iteration is selected to undergo local search before the ant updates the pheromone trail on the construction graph according to the quality of its solution. The pheromone amounts deposited on the decision components guide the subsequent ants towards new better candidate solutions. After that, it compares the current iteration's best solution with the global best solution to keep track of the best solution found along the entire search so far. This set of steps is repeated until the algorithm converges on a solution or the maximum number of iterations is reached.

In order to build the structure of a BN classifier, the maximum number of parents for a node is typically specified by the user. However, the selection of the optimal number of dependencies that a variable in the network can have is automatically carried out in ABC-Miner [14]. To create a candidate solution, an ant starts with the network structure of Naïve-Bayes, i.e. a BN in which all the variables have only the class variable as the parent. Then it expands that

structure into a Bayesian Augmented Naïve-Bayes (BAN) structure by adding edges to the network. The selection of the edges is performed according to a probabilistic state transition formula that involves pheromone amount and the heuristic information – using conditional mutual information [14] associated with the edges. An edge is valid to be added to the BN classifier being constructed if its inclusion does not create a directed cycle and does not exceed the limit of k parents (chosen by the current ant). After the ant adds a valid edge, all the invalid edges are eliminated from the construction graph. The ant keeps adding edges to the current solution until no valid edges are available. When the structure is finished, the CPT is computed for each variable, producing a complete BN classifier. Then the quality of the solution is evaluated and all the edges become available for constructing further candidate solutions.

The ABC-Miner algorithm evaluates the quality of the BN classifier using *accuracy* [14], a conventional measure of predictive performance, since the goal is to build a BN only for predicting the value of a specific class attribute, unlike conventional BN learning algorithms whose scoring function do not distinguish between the predictor and the class attributes. Experimental evaluations showed the predictive effectiveness of ABC-Miner in flat classification comparing to other Bayesian classification algorithms, namely: Naïve-Bayes, TAN and GBN [14]. Thus, we carry on using it for hierarchical classification.

4 Hierarchical Classification

Hierarchical classification refers to the task of predicting the class value(s) of a given case in a domain where the class values are arranged into a hierarchy. Many real-world classification problems have hierarchical classes, were a case belongs to a series of classes related in a general-to-specific structure. Such class structure is found, e.g., in document topics, music genres and protein functions, which makes the classification task more complex and challenging [19].

Figure 1 shows examples of hierarchical classification problems. A hierarchical classification problem can be characterized by three properties:

Graph Structure - This specifies whether the type of graph representing the hierarchical classes is tree or DAG. A node in the tree structure has only one parent, while it can have multiple parents in the DAG.
Labelling Type - This indicates whether a case is allowed to have class labels associated with a single or multiple paths in the class hierarchy. The hierarchy can be a tree, yet a case can be labelled with classes in different paths.
Labelling Depth - In full depth labelling, every case is labelled with classes at all levels, from the root to the leaf level. Partial depth labelling indicates that for some cases the value of the class label at the leaf level is not specified.

There are three different broad approaches to tackle hierarchical classification problems [19]. The first (and the simplest) approach is to completely ignore the class hierarchy and convert the problem into flat classification by predicting only classes at the leaf nodes. The second approach is to produce one or

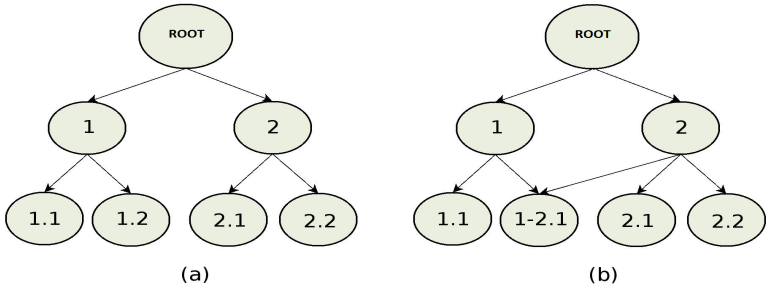

Fig. 1. Examples of hierarchies of classes: (a) Tree and (b) DAG. If each case in (a) is labelled with exactly one of the classes: 1.1, 1.2, 2.1, 2.2, then (a) is a single-path and full-depth classification problem. If a case in (b) is labelled with only class 2, and another case is labelled with both 1.1 and 1-2.1, then (b) is a multiple-path and partial-depth problem.

more local classifiers for each class level and combine their outputs. Each classifier is trained with a flat classification algorithm using a local data subset, discriminating among a subset of classes in the hierarchy. The third (and most complex) approach is to design a specific classification algorithm that can process the whole hierarchy in a global way. The local approach is widely used since it utilizes existing flat classification algorithms and the divide-and-conquer principle (see below) to solve more easily manageable classification problems, by comparison with the first and third approaches where a single classifier has to discriminate among a very large set of all leaf classes or all classes in the hierarchy, respectively.

The local approach for hierarchical classification uses the principle of "divide and conquer" to process the class hierarchy in a top-down fashion. More precisely, we use the "one-classifier-per-node" version of the local approach, which is suitable for our dataset, where the graph type is DAG, labelling type is multiple paths, and labelling depth is partial depth (see Section 6). In this local approach version, a binary flat classifier is built for each class in the hierarchy (except the root node). The flat classification algorithm at a given node uses the cases belonging to this class node as positive examples, and the cases belonging to the siblings of this class node (but not to the current class node) as negative examples in the training phase. To illustrate how to classify a case in the testing phase, we use example (b) in Figure 1. The local classifier at node 1 decides whether a given case belongs to this class or not, the same for node 2. If the case is classified as belonging to class 2, it is passed to the classifiers of the class nodes 2.1 and 2.2 (the case is also passed to the classifier at node 1-2.1 if it is classified as belonging to class 1 and 2) at the next level. If the case does not belong to class 2, no further classifications are performed from node 2. In this example, a case can be labelled with, say, classes 1, 2, and 2.2, which is appropriate for the aforementioned characteristics of our dataset.

5 Proposed Methods for an Ensemble of Classifiers

An ensemble of classifiers is often used to combine the predictions from separate classifiers in order to increase predictive accuracy. The idea is to construct an ensemble of classifiers having different inductive biases, so they make different errors. Hence, combining their classification outputs will make the overall prediction of the ensemble more accurate [22].

There are two main issues in ensembles of classifiers. First, the type of diversity in the classifiers: different algorithms with the same dataset, different attributes (features) from the same dataset, different data representations with the same algorithm, and different training case subsets with the same algorithm (e.g., bagging). The second issue is how the classes predicted by the classifiers are combined. Table 1 shows how our work fits into the context of related work on hierarchical classification according to these two issues.

Table 1. Related Work on Ensembles in Hierarchical Protein Function Classification

		Approaches to Combine Classifier Ensemble' Predictions			
		Select the best	Majority Vote	Weighted Vote	Stacking
Diversity in the Ensemble	Different Algorithms	Secker et al. [17], Silla et al. [18]			Costa et al. [4]
	Different Data Representations	**This Work**, Silla et al. [18]	**This Work**	**This Work**	**This Work**
	Different Feature Subsets	Secker et al. [16]			
	Different Data subsets			Schietgat et al. [15]	

In addition to the use of our ABC-Miner for hierarchical protein function prediction in a new ageing-related dataset, what is novel in this work are the various methods we employ to combine the outputs of an ensemble's classifiers built with different protein representations in hierarchical classification. As shown in Table 1, three of the four cells marked with the keyword "this work" involve a new combination of the technique of building classifiers with different protein representations (proposed in [18]) with a technique for combining the classifiers' predictions.

In essence, ABC-Miner builds an ensemble of four binary classifiers (one for each protein representation, i.e. a pre-defined feature set) for each class node in the hierarchy. The selective method chooses the best classifier – with the highest accuracy in a validation set (a subset of the training set) [17] – in the ensemble to predict whether or not a given case belongs to the current local class node. The majority voting method uses the majority decision of the ensemble to make a local class prediction. For example, if three out of four classifiers predicted

that a given case does not belong to the current class, this case is not labelled by this class. The weighted voting method weights the vote of each classifier by its accuracy on a validation set (again, a subset of the training set), and predicts the class with the the largest sum of weighted votes.

The stacking method is a meta-classifier built upon the predicted classes of the ensemble's classifiers to perform the final classification of a case. For a given training case, a meta-case is constructed with four attributes: each represents the class predicted by one of the ensemble's four classifiers. The class of the meta-case is the same as the one in the training case. At each node in the hierarchy the algorithm uses an ensemble's classifiers to classify the n cases of its local training subset and then n meta-cases are generated upon the output of the classifiers. A meta-classifier is built to learn the relationships between the predictions of the various classifiers and the actual classes. In the testing phase, the meta-classifier outputs the final classification for a test case upon the predictions of the ensemble's classifiers.

The accuracy of a classifier in an ensemble is measured by the hierarchical F-measure, which combines hierarchical precision (hP) and hierarchical recall (hR), as shown in the following formula, where A_i is the set of actual (true) class labels and P_i is the set of predicted class labels, respectively, for the ith case, and n is the total number of validation cases.

$$hP = \frac{\sum_{i=1}^{n} |A_i \cup P_i|}{\sum_{i=1}^{n} |P_i|} \quad hR = \frac{\sum_{i=1}^{n} |A_i \cup P_i|}{\sum_{i=1}^{n} |A_i|} \quad hF = \frac{2.hP.hR}{hP + hR}\,, \tag{3}$$

6 The Ageing-Related Protein Dataset

A new dataset of human ageing-related protein was created specifically to be used in our experiments. A case (protein) has four different sets of predictor attributes (four protein representations), extracted from its amino acid sequence as described later, and a set of class labels (functions) to be predicted. These functions are hierarchically-related biological processes in the Gene Ontology (GO) [20]. To create the dataset, first a set of ageing genes were obtained from the GenAge database [10], which contained 260 human ageing-related genes (species "Homo sapiens"). Note that we obtained only the gene names from the GenAge dataset, the proteins target classes and features were obtained as follows.

Next, we used the Swiss-Prot database [21] from the UniProt knowledge base to obtain the sequences of the proteins corresponding to those genes (from which the predictor attributes are extracted) and the biological process GO terms (to be predicted) for each protein. The total number of GO terms included in that set of proteins was 2889. Considering each of those terms as a class to be predicted is not desirable, for two reasons. First, the number of classes in the hierarchy would be very large, with many classes having very few cases (making their prediction unreliable). Second, a lot of those GO terms are irrelevant to the process of ageing, since an ageing-related protein often performs other biological processes unrelated to ageing. Therefore, it is desirable to have a subset of these terms containing only GO terms related to the ageing process.

In order to mitigate these problems, we used the DAVID [8] bioinformatics tool to identify *enriched* GO terms in our set of ageing-related proteins. DAVID performs statistical tests to identify the GO terms whose annotations are most correlated with a specific set of proteins. We set the minimum number of proteins per GO term to 15, and set the EASE parameter that represents the maximum p-value of the enrichment significance to 1E-10. We obtained 102 biological process GO terms, which were used as the class hierarchy in our dataset. The dataset was reduced to 247 proteins, as 16 were lost due to the GO terms reduction. Table 2 shows the top 100 GO terms (GO:Term Proteins_Count) in our dataset sorted by the protein count. The top left term is the root node that has the 247 proteins. To see the names and the hierarchical organization of the terms, please refer to QuickGO [1], where only the "IS-A" relationship is considered in our construction of the class hierarchy.

Table 2. Top 100 GO Terms in the Dataset and Their Protein Counts

GO:0008150 247	GO:0034641 119	GO:0031325 91	GO:0043085 62	GO:0048856 42
GO:0009987 241	GO:0007165 119	GO:0010604 89	GO:0050793 61	GO:0009725 40
GO:0065007 207	GO:0009889 119	GO:0032502 85	GO:0009892 60	GO:0010942 40
GO:0050789 204	GO:0051171 118	GO:0007166 85	GO:0010557 60	GO:0043068 40
GO:0050794 201	GO:0031326 116	GO:0065008 84	GO:0006259 59	GO:0043065 40
GO:0008152 196	GO:0019219 116	GO:0033554 78	GO:0031399 59	GO:0045944 40
GO:0044237 188	GO:0048519 115	GO:0050790 78	GO:0051726 56	GO:0051094 38
GO:0050896 184	GO:0006950 111	GO:0010033 76	GO:0006974 55	GO:0009628 37
GO:0044238 181	GO:0042221 111	GO:0010941 76	GO:0010605 55	GO:0006281 37
GO:0051716 163	GO:0006139 110	GO:0043067 76	GO:0006357 54	GO:0008284 37
GO:0019222 158	GO:0048523 110	GO:0042981 76	GO:0007167 53	GO:0012502 31
GO:0043170 156	GO:0010556 107	GO:0035556 75	GO:0051254 53	GO:0006917 31
GO:0044260 150	GO:0090304 104	GO:0044093 74	GO:0010628 52	GO:0009314 29
GO:0031323 148	GO:2000112 104	GO:0051246 72	GO:0007169 51	GO:0040008 28
GO:0080090 145	GO:0010468 103	GO:0009891 69	GO:0060548 49	GO:0043434 27
GO:0048518 144	GO:0065009 102	GO:0032268 69	GO:0043069 49	GO:0051052 27
GO:0060255 143	GO:0009893 97	GO:0051173 67	GO:0045893 49	GO:0006979 26
GO:0048522 135	GO:0051252 96	GO:0031328 67	GO:0043066 49	GO:0006916 26
GO:0006807 120	GO:2001141 96	GO:0045935 64	GO:0009719 47	GO:0048513 19
GO:0034641 119	GO:0006355 94	GO:0042127 62	GO:0042592 44	GO:0009416 19

We extracted 4 sets of protein features (used in [18]), each set constituting a different protein representation, from the proteins' amino acid sequences. In essence, the first feature set contains 5 z-values, representing hydrophobicity/hydrophilicity (z1), steric/bulk properties and polarizability (z2), polarity (z3), and electronic effects (z4 and z5) of the amino acids in the whole sequence. The second set contains 15 z-values; 5-values are averaged over the whole sequence, other 5 z-values are averaged over the N-terminus (the first 150 amino acids) of the protein, and further 5 z-values are computed from the C-terminus (the last 150 amino acids) of the sequence.

The third feature set contains amino acid compositions, which are the percentages of occurrence of each amino acid within a protein sequence. This produces a set of 20 features, each of them with the percentage of how many times a given amino acid occurs within the protein sequence. The fourth set contains local descriptors, consisting of 21 features (3 Composition, 3 Transition and 15 Distribution features), which are computed based on the variation of occurrence of functional groups of amino acids within the protein sequence. The functional groups used were: hydrophobic (amino acids CVLIMFW), neutral (amino acids GASTPHY), and polar (amino acids RKEDQN). We also added the sequence length and molecular weight of the protein as two additional features to each of the feature sets, since both are easy to obtain and may be somewhat relevant to protein functional prediction.

7 Experimental Results

The experiments were carried out using a well-known 5-fold cross validation procedure [22]. The results (average F-measure on the test set) are reported in Table 3. From one perspective, we compare the performance of the various ensemble-of-classifiers methods to the use of each protein representation separately. From another perspective, we compare the use of ABC-Miner as the base classifier in an ensemble to the use of other BN classification algorithms: Naïve-Bayes, GBN and TAN. Parameter settings are the same as in [14].

In Table 3, the top four rows refer to the use of each protein representation separately, so that no ensemble of classifiers is built. The bottom four rows refer to experiments with ensembles, using the various methods of prediction combination (discussed in Section 5) of the ensemble of classifiers built with different representations. An entry in bold face indicates that it is the highest F-measure value obtained across the four algorithms for the same experiment variation. An underlined entry indicates that it is the highest F-measure value obtained in all of the experiment variations for the same algorithm.

Table 3. Predictive Performance (*mean* ± *standard error*) Results - F-measure

Experiment Variation	Naïve-Bayes	TAN	GBN	ABC-Miner
5 Z-values	0.196 ± 0.02	0.285 ± 0.01	0.218 ± 0.02	**0.342 ± 0.02**
15 Z-values	0.307 ± 0.01	**0.321 ± 0.01**	0.242 ± 0.01	0.298 ± 0.01
AA Composition	0.279 ± 0.01	0.283 ± 0.02	0.271 ± 0.01	**0.363 ± 0.02**
Local Descriptors	0.345 ± 0.00	0.340 ± 0.00	0.342 ± 0.00	**0.347 ± 0.00**
Selective Classifier	0.332 ± 0.01	0.332 ± 0.01	0.287 ± 0.01	**0.351 ± 0.02**
Majority Voting	0.269 ± 0.01	**0.372 ± 0.02**	0.251 ± 0.02	0.366 ± 0.01
Weighted Voting	0.376 ± 0.01	0.378 ± 0.02	0.376 ± 0.02	**0.481 ± 0.01**
Meta-Classifier	0.308 ± 0.02	0.381 ± 0.01	0.328 ± 0.02	**0.397 ± 0.02**

As shown in Table 3, ABC-Miner obtained the highest performance in 6 out of 8 experiments variations, where in 5 variations it was significantly better than the second best algorithm according to a two-tailed Student's t-test with significance level of 5%. TAN performed the second best, since it came in the first place in 2 variations and in the second place in 5 variations. In addition, the different approaches for combining an ensemble's classifiers' predictions have generally performed better than the use of each protein representation separately. Specifically, the weighted voting approach obtained the best results with 3 out of 4 algorithms. The meta-classifier came in the first place for 1 algorithm and in the second place for 2 algorithms. Unexpectedly, the selective approach has outperformed the majority vote for 2 algorithms. For each of the four used algorithms, the best ensemble method outperformed the best single protein representation at the statistical significance level of 5%, showing the effectiveness of the proposed ensemble methods over other approaches.

8 Concluding Remarks

In this paper we have used our recently introduced ant-based Bayesian classification algorithm in several ensembles of classifiers to tackle the hierarchical protein function prediction problem in a local approach. We created a new ageing-related protein dataset to carry out our experiments. Results have showed the effectiveness of the proposed ensemble methods in hierarchical classification, especially when ABC-Miner is used. However, lack of prediction model comprehensibility is an issue; a large number of classifiers (four for each class in the hierarchy) are built, reducing the interpretability of the models by users. This is an inherited drawback of the local approach. Therefore, an important research direction is to extend ABC-Miner to tackle hierarchical classification in a global fashion, where only one classification model is produced for the entire class hierarchy.

Acknowledgements. The authors thank Dr. Carlos Silla for extracting the feature sets used in our experiments and Dr. Joao Pedro de Magalhaes for his valuable advice about the creation of the ageing-related protein's dataset.

References

1. Binns, D., Dimmer, E., Huntley, R., Barrell, D., O'Donovan, C., Apweiler, R.: QuickGO: a web-based tool for Gene Ontology searching. Bioinformatics 25, 3045–3046 (2009)
2. de Campos, L.M., Fernandez-Luna, J.M., Gamez, J.A., Puerta, J.M.: Ant colony optimization for learning Bayesian networks. International Journal of Approximate Reasoning 31(3), 291–311 (2002)
3. Cheng, J., Greiner, R.: Learning Bayesian Belief Network Classifiers: Algorithms and System. In: Stroulia, E., Matwin, S. (eds.) AI 2001. LNCS (LNAI), vol. 2056, pp. 141–151. Springer, Heidelberg (2001)

4. Costa, E.P., Lorena, A.C., Carvalho, A.C.P.L.F., Freitas, A.A.: Top-Down Hierarchical Ensembles of Classifiers for Predicting G-Protein-Coupled-Receptor Functions. In: Bazzan, A.L.C., Craven, M., Martins, N.F. (eds.) BSB 2008. LNCS (LNBI), vol. 5167, pp. 35–46. Springer, Heidelberg (2008)
5. Daly, R., Shen, Q., Aitken, S.: Learning bayesian networks: Approaches and issues. Knowledge Engineering Reviews 26(2), 99–157 (2011)
6. Dorigo, M., Stützle, T.: Ant Colony Optimization. The MIT Press (2004)
7. Freitas, A.A., de Carvalho, A.C.P.F.L.: A tutorial on hierarchical classification with applications in bioinformatics. In: Research and Trends in Data Mining Technologies and Applications, pp. 175–208 (2007)
8. Huang, D., Sherman, B., Lempicki, R.: Systematic and integrative analysis of large gene lists using DAVID Bioinformatics Resources. Nature Protocol 4, 44–57 (2009)
9. Jiang, L., Wang, D., Cai, Z., Yan, X.: Survey of Improving Naive Bayes for Classification. In: Alhajj, R., Gao, H., Li, X., Li, J., Zaïane, O.R. (eds.) ADMA 2007. LNCS (LNAI), vol. 4632, pp. 134–145. Springer, Heidelberg (2007)
10. de Magalhaes, J., Budovsky, A., Lehmann, G., Costa, J., Li, Y., Church, V.F.G.: The Human Ageing Genomic Resources: online databases and tools for biogerontologists. Aging Cell, 65–72 (2009)
11. Martens, D., Baesens, B., Fawcett, T.: Editorial survey: swarm intelligence for data mining. Machine Learning 82(1), 1–42 (2011)
12. Parpinelli, R.S., Lopes, H.S., Freitas, A.A.: Data mining with an ant colony optimization algorithm. IEEE TEC 6, 321–332 (2002)
13. Pinto, P.C., Nägele, A., Dejori, M., Runkler, T.A., Sousa, J.M.C.: Using a local discovery ant algorithm for Bayesian network structure learning. IEEE Transactions on Evolutionary Computation 13(4), 767–779 (2009)
14. Salama, K.M., Freitas, A.A.: ABC-Miner: An Ant-Based Bayesian Classification Algorithm. In: Dorigo, M., Birattari, M., Blum, C., Christensen, A.L., Engelbrecht, A.P., Groß, R., Stützle, T. (eds.) ANTS 2012. LNCS, vol. 7461, pp. 13–24. Springer, Heidelberg (2012)
15. Schietgat, L., Vens, C., Struyf, J., Blockeel, H., Kocev, D., Dzeroski, S.: Predicting gene function using hierarchical multi-label decision tree ensembles. BMC Bioinformatics 11(1) (2010)
16. Secker, A., Davies, M.N., Freitas, A.A., Clark, E., Timmis, J., Flower, D.R.: Hierarchical classification of GPCR with data-driven selection of attributes and classifiers. International Journal of Data Mining and Bioinformatics 4(2), 191–210 (2010)
17. Secker, A., Davies, M.N., Freitas, A.A., Timmis, J., Mendao, M., Flower, D.R.: An experimental comparison of classification algorithms for the hierarchical prediction of protein function. Expert Update (BCS-SGAI Magazine) 9, 17–22 (2007)
18. Silla, C.N., Freitas, A.A.: Selecting different protein representations and classification algorithms in hierarchical protein function prediction. Intelligent Data Analysis 15(6), 979–999 (2011)
19. Silla, C.N., Freitas, A.A.: A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery 22(1-2), 31–72 (2011)
20. The Gene Ontology Consortium: Gene ontology: tool for the unification of biology. Nature Genetics 25(1), 25–29 (2000)
21. The UniProt Consortium: The Universal Protein Resource (Uniprot). Nucleic Acids Research 38, D142–D148 (2010)
22. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn. Morgan Kaufmann, San Francisco (2010)
23. Wu, Y., McCall, J., Corne, D.: Two novel Ant Colony Optimization approaches for Bayesian network structure learning. In: International Conference on Evolutionary Computation (CEC), pp. 1–7 (2010)