Soft Comput (2005)
DOI 10.1007/s00500-005-0490-z

FOCUS

D. F. Tsunoda · H. S. Lopes

Automatic motif discovery in an enzyme database using a genetic algorithm-based approach

Published online: 10 May 2005

Abstract Proteins can be grouped into families according to some features such as hydrophobicity, composition or structure, aiming to establish the common biological functions. This paper presents a system that was conceived to discover features (particular sequences of amino acids, or motifs) that occur very often in proteins of a given family but rarely occur in proteins of other families. These features can be used for the classification of unknown proteins, that is, to predict their function by analyzing the primary structure. Runnings were done with the enzymes subset extracted from the Protein Data Bank. The heuristic method used was based on a genetic algorithm using specially tailored operators for the problem. Motifs found were used to build a decision tree using the C4.5 algorithm. The results were compared with motifs found by MEME, a freely available web tool. Another comparison was made with classification results of other two systems: a neural network-based tool and a hidden Markov model-based tool. The final performance was measured using sensitivity (Se) and specificity (Sp): similar results were obtained for the proposed tool (78.79 and 95.82) and the neural network-based tool (74.65 and 94.80, respectively), while MEME and HMMER resulted in an inferior performance. The proposed system has the advantage of giving comprehensible rules when compared with the other approaches. These results obtained for the enzyme dataset suggest that the evolutionary computation method proposed is very efficient to find patterns for protein classification.

1 Introduction

Since the beginning of the genome sequencing projects, biological databases have been overwhelmed by experimental data. Currently, most of the research results are freely available in the Internet and are organized in databases. After sequencing an organism, researchers turn to the laborious task of annotation. Afterwards, the proteome of the organism is seen as one of the main products of the whole process of genome sequencing.

Proteins are responsible for several functions such as: transport of small molecules (e.g., hemoglobin), regulation (e.g., insulin), sustentation (e.g., collagen), increase of reaction speed (e.g., enzymes) and others. Biological organisms have thousands of different types of proteins, which are constituted basically of amino acids linked in linear chains through peptide connections. Active intra-molecular forces cause the proteins to assume specific three-dimensional shapes that are directly related to their biological functions [7]. Proteins are grouped into super families, families and subfamilies according to their biological function. Therefore, the classification of proteins is an important task for the molecular biologist, and, ultimately, it is aimed to identify the function of the protein.

There are several protein databases available, for instance, Swiss-Prot and Protein Data Bank (PDB) [1]. In this work we used PDB that contains information about the primary, secondary and tertiary structures of thousands of proteins, besides many other data. The choice for PDB was due to the intention to improve this work in the future (not reported here) using information provided by the secondary and tertiary structures.

The protein-classification problem (PCP) is a very important research area in bioinformatics. As mentioned before, the many genome sequencing projects has unveiled a growing number of gene products whose function is unknown or barely estimated by homology techniques. The prediction of protein function has been done basically in two ways: prediction of the protein structure and then prediction of function from the structure, or else classifying proteins into functional families and supposing that similar sequences will have similar functions. Notwithstanding, most proteins share similar structures (in particular, considering the primary structure), since many of them have a common evolutionary origin [11].

D. F. Tsunoda · H. S. Lopes (✉)
Laboratório de Bioinformática / CPGEI,
CEFET-PR, Av. 7 de setembro,
3165 80230-901 Curitiba (PR), Brazil
E-mail: hslopes@cefetpr.br
Tel.: +55-41-3104694
Fax: +55-41-3104683

Common structures may be characteristic of a given family of proteins but, on the other hand, unrelated families can also share common structures. This two-fold characteristic makes protein classification a difficult problem.

Computer researches have been using many heuristics to present possible solutions for the PCP, for instance: artificial neural networks [16, 15], clustering [9], genetic algorithms [10] and other data mining algorithms [12, 14].

This paper reports the development and application of a computational tool, based on an evolutionary computation technique, a genetic algorithm, specially devised for the automatic discovery of motifs using as input the primary structure of proteins. The system finds sequences of amino acids (features or motifs) that occur very often in proteins of a given class (family) but rarely occur in proteins of other classes. Those discovered motifs can be further used for the characterization of families of proteins as well as for the automatic classification of unknown proteins.

Genetic algorithms were used mainly for its ability to perform adaptive, powerful and robust searches. Besides, as an evolutionary computation technique, they operate in parallel over a population of candidate solutions, allowing a simultaneous exploration of different regions of the search space in the solution domain.

2 Genetic algorithms

A genetic algorithm (GA) is a search and optimization methodology from the field of evolutionary computation that was invented by Holland [6]. A GA is based on the Darwin's natural selection principle of the survival of the fittest and is widely used for hard problems in engineering and computer science. A GA is a population-based method where each individual of the population represents a candidate solution for the target problem. This population of solutions is evolved throughout several generations, starting from a randomly generated one. During each generation of the evolutionary process, each individual of the population is evaluated by a fitness function, which measures how good the solution represented by the individual is for the target problem. From a given generation to another, some "parent" individuals (usually those having the highest fitness) produce "offsprings", i.e. new individuals that inherit some features from their parents, whereas others (with low fitness) are discarded, following Darwin's principle of natural selection. The selection of parents is based on a probabilistic process, biased by their fitness value. Following this procedure, it is expected that, on average, the fitness of the population will not decrease every consecutive generation. The generation of new offsprings, from the selected parents of the current generation, is accomplished by means of genetic operators. This process is iteratively repeated until a satisfactory solution is found or some stop criterion is reached, such as the maximum number of generations.

3 Methodology

The version of the Protein Data Bank (PDB) used in this work was #102. PDB file-encoded sequences are notoriously cumbersome to manipulate because the structural completeness of data is not always guaranteed. Besides, in PDB, proteins files are compressed using GZIP^{TM} format. Therefore, a specific tool is necessary for uncompressing and extracting correct formatted data from the relevant fields. A PDB file has information about the three levels of proteins structures but, for the purposes of this work, only the primary structure information was used: name, class and amino acids sequence. In the future, it is intended to use more data to improve the system.

The PDB is an archive of experimentally determined three-dimensional structures of biological macromolecules. The PDB available classification is: proteins, peptides, and viruses; protein/nucleic acid complexes; nucleic acids or carbohydrates and their sources: X-ray Diffraction and other or NMR.

The PDB files include atomic coordinates, bibliographic citations and structural information. These information follows a PDB format guide, e.g., there is a field named SEQRES that contains the amino acid or nucleic acid sequence of residues in each chain of a chosen macromolecule. These residues names are also standard for amino acids: ALA, ARG, ASN, ASP, CYS, GLN, GLU, GLY, HIS, ILE, LEU, LYS, MET, PHE, PRO, SER, THR, TRP, TYR, VAL or UNK (for an unknown amino acid in the structure).

The developed system allows the extraction of relevant information and makes the conversion of non-standard amino acids for the letter U (unknown), since this letter is not part of the standard letters set for amino acids representation.

Before loading the input file (created from thousands of PDB files), it is possible to set the system to discard those classes with less than a minimum number of proteins. This option is necessary because there are many classes with a very few number of proteins, which are not significant for classification purposes. Therefore, all tests reported in this paper have been performed over classes with more than 10 proteins. Enzymes were extracted from PDB using the EC number given by the International Union of Biochemistry and Molecular Biology (IUBMB). From a data mining viewpoint, each EC number corresponds to a class, i.e., a specific protein function. More precisely, the EC number consists of four digits, where each pair of adjacent digits is separated by a dot ("."), and it specifies the chemical reaction catalyzed by the corresponding enzyme. For instance, the enzyme alcohol dehydrogenase has the number EC.1.1.1.1.

Some of the enzymes stored in the PDB contained the called non-standard amino acids, from which no useful motif can be discovered. Therefore, as part of our data preparation procedure, we have only retrieved from PDB the enzymes whose primary sequence has at least 30 standard amino acids. After this simple filtering, the total number of proteins retrieved from PDB was 8,339. As usual in the literature, the classification accuracy rate is computed in a test set

separated from the training set [5] in order to evaluate the generalization ability of the discovered patterns. The enzymes are distributed across the six classes as follows: 1,483 proteins in class EC.1 (oxidoreductases); 1,766 in class EC.2 (transferases); 3,285 in class EC.3 (hydrolases); 675 in class EC.4 (lyases); 381 in class EC.5 (isomerases) and 209 in class EC.6 (ligases). Therefore, the training set had 600 enzymes, evenly distributed into these six classes. The test set was built with the remaining 7,739 enzymes.

4 Algorithm description

4.1 Encoding and fitness function

Using GA for real-world problems encompasses two crucial definitions: the encoding scheme of an individual and the fitness function. In the implemented system, every individual represents a motif, that is, a variable-length string of characters. The alphabet used for encoding is the 20 standard letters for representing amino acids, plus letter U.

Recall that our goal is to find a sequence of amino acids (motif) with a high discriminatory power – i.e., a pattern that occurs in most proteins of a given class and occurs in few or no proteins of all other classes. Therefore, this pattern can be characteristic of a given family, allowing it to be discriminated from all others - the essence of classification.

In order to discriminate an individual, we developed a special fitness function that is computed as follows. First the EA computes, for each class i, $i = 1, \ldots, 6$ (for the enzymes dataset used in this work), the relative frequency of occurrence of the motif in that class. This is simply the number of proteins of the i-th class where the motif occurs in the protein's sequence. Second, the EA computes, for each class i, a measure of the ability of the motif to discriminate between class i and the other classes, denoted Disc_i and given by the Eq. 1:

$$\text{Disc}(i) = F_i . \left(1 - (k-1)^{-1} . \sum_{j=1}^{n} F_{j, j \neq i} \right) , \qquad (1)$$

where F_i is the relative frequency of the individual's motif in the i-th class, n is the number of classes ($n = 6$ in this work), and k is the number of classes that contain at least one protein whose primary sequence contains the individual's motif. The rightmost term of the formula simply computes the average relative frequency of the motif in all the $(n-1)$ classes j such that $j \neq i$. This term is subtracted from 1, so that the term between square brackets is to be maximized – the higher its value, the better the value of Disc_i. Similarly, the value of F_i (the first term of the formula) is also to be maximized, so that a high value of Disc_i means that the motif occurs very often in class i but rarely in the other classes.

Once the value of Disc(i) has been computed for all n classes ($i = 1, \ldots, n$), the individual is associated with the class having the largest value of Disc(i). In other words, the motif represented by the individual is considered as a characteristic pattern for proteins of class i. The proposed fitness function is normalized in the range [0..1], making the interpretation of results somewhat easier, since 1 is the best possible value, meaning maximum discrimination.

4.2 Selection method and genetic operators

Selection is not an operator by itself. It is a procedure that takes place before the application of the genetic operators. The selection method used by the system is the stochastic tournament. This method randomly takes k individuals ($k = 2$) of the population and chooses the one with the largest fitness. Usually, k is a user-defined percentage of the population. This process is repeated P times (with reposition), where P is the population size. The copy of the selected P individuals will undergo the genetic operators, as follows.

In this work, the usual one-point crossover operator is stochastically applied with a predefined probability, using two individuals of the selected pool. Since the length of the chromosome is variable, the traditional concept of crossover point was slightly modified and adapted to our individual representation. The crossover point is a percentage (of the length of the individual) that defines the starting point from where the crossover breaks the string.

The mutation operator is used to foster more exploration of the search space and to avoid unrecoverable loss of genetic material that leads to premature convergence to some local minima. In general, mutation is implemented by changing the value of a specific position (an allele) of a chromosome with a given probability, denominated mutation probability. Due to the specificity and purpose of our system the mutation operator was implemented as a set of four different kinds of sub-operations over a single individual:

1. Left-adding: one randomly generated character (corresponding to an amino acid) is added to the left of the motif.
2. Right-adding: one randomly generated character (corresponding to an amino acid) is added to the right of the motif.
3. Random-changing: all the amino acids from a randomly selected starting point up to the end of the motif are changed, except the first and the last position.
4. Cutting-out: it removes a single character from the amino acid sequence. The removal position is randomly generated.

The mutation probability is a user-defined parameter, as usual in GA, however, the mutation is divided in four different sub-operations, each one having an equal probability of choice, which is made in a random basis.

Both crossover and mutation operators are "hill climbing-based operators" because they are implemented in such a way that a new individual is immediately evaluated after its generation and, if its fitness is lower than the parent's fitness the operation is undone. This procedure, although computationally expensive, makes the evolutionary process faster in terms

of number of generations, since the generated offspring will be always better than their parents (or will not be generated otherwise). To alleviate the computational load, in our system, after a genetic operator be selected according to a given probability, it can be applied as usual or as a hill climbing-based operator. This choice is done probabilistically according to a user-defined parameter – hill climbing-based operator rate.

4.3 Running parameters

As described earlier, the GA has several parameters. Hence, 35 preliminary runnings were performed to find good values for some of these parameters. In these preliminary runnings the expansion operator was turned off, because this is a computationally expensive operator and we wanted to perform some relatively quick runnings just to set some parameters. More precisely, these preliminary runnings produced the following parameter values: tournament size = 10%; probability of crossover = 80%; probability of mutation = 80%; population size = 200; number of generations = 300; hill climbing-based operator probability = 10%. Note that the hill climbing-based operator rate is low, in order to avoid losing population diversity and to prevent a premature convergence. Having fixed these parameters, nine runnings were made to evaluate the influence of the probability of crossover (20, 50, 80%) and probability of mutation (20, 50, 80%) and the best result had probability of crossover = 20%; probability of mutation = 80%. Another important parameter to be adjusted was the hill climbing-based operator probability and five tests were made with 0, 20, 40, 80 and 100%; the 40% value returned the best result. As expected, this operator increases the computational cost. For a 0% running, the time taken was 2.8 h (for the five-fold cross validation). For 20, 40, 80 and 100%, the times were 4.4, 9.1, 47 and 116 h, respectively.

Subsequently, other two runnings (using the two best obtained results) were made to evaluate the influence of expansion operator. Evaluations of those runnings showed that expansion operator increase the quality of obtained motifs.

A conventional GA returns, as its result, the best individual (the one with highest fitness) generated during the run. However, in our system the desired result is not a single individual, but rather, a set of individuals. The reason is that each individual represents a single amino acid sequence (motif), associated with a single class, and these patterns will be used further to classify proteins. Therefore, it is necessary to discover many patterns, associated with as many different classes as possible during genetic search.

In each generation, after the fitness of all individuals have been computed, some high-quality motifs are saved in a separated file, called the set of discovered patterns (SDP). In fact, the individuals representing those patterns still remain in the population; only a copy of them is saved into SDP. The criterion to select these individuals is their fitness – only those with fitness greater than a user-defined minimum quality threshold will be saved. Special care was taken to prevent adding motifs that are substrings of other motifs already in the SDP. This procedure results in the discovery of many motifs, associated with different classes, as desired.

5 Classification

Using the training data, motifs were discovered by using two different tools: our system, named Genetic Algorithm for Motif DIscovery (GAMDI) and Multiple EM for Motif Elicitation (MEME) [2].

MEME is a freely web tool[1] supported by the San Diego Supercomputer Center that uses statistical modeling techniques to automatically choose the best width, number of occurrences, and description for each motif. The web version of MEME used in this paper requires that the sequences in your group have less than 60,000 characters in total. We submitted the enzyme families to MEME web version.

The five best motifs discovered (a higher fitness means a better discriminatory ability of the motif, what makes it better) by each approach (MEME and GAMDI) were used to classify the test set (7,739 enzymes). Those motifs are viewed as attributes of a decision tree in which each node tests the presence of the motif.

There are many decision tree algorithms, and the most popular is the C4.5 algorithm [13]. The C4.5 decision tree method can discover relationships between the classification of objects and their attributes. The algorithm goal is to construct a decision tree with minimum number of nodes that gives the least number of misclassifications on training data. The C4.5 machine learning algorithm enabled us to extract a tree for enzymes classification, based on the previously discovered motifs.

To generate the decision trees for classification of unknown instances (enzymes) we used Waikato Environment for Knowledge Analysis (WEKA), a Java-based tool freely available in the Internet[2] [17]. Decision trees were generated using the standard parameters of the J48 algorithm (the WEKA version of C4.5), using motifs generated by GAMDI and MEME.

Figure 1 illustrates a hypothetic example of a decision tree generated by WEKA. Each internal node (including the root node) tests for the presence (1) or absence (0) of an attribute (in this case, a motif). For example, a rule from this tree is: if a protein has the motif LKG (LKG = 1), does not have the motif AELD (AELD = 0), but the motif VLG is present (VLG = 1), the protein will be classified as EC2 (that is, a transferase). Notice that it is possible to have many rules for the same class, for example EC4 (Fig. 1) has three associate rules: first, "if there is not the motif LKG nor DGL then EC4", second "if there are motifs LKG and AELD but not VLG, then EC4" and third "if there are motifs LKG and AELD and LKG but not DLG, then EC4".

[1] *http://meme.sdsc.edu/meme/website/intro.html*

[2] *http://www.cs.waikato.ac.nz/ml/weka*

Table 1 Comparative results for classification of enzymes

	GAMDI		MEME		HMMER		Neural networks	
Enzyme Class	Se	Sp	Se	Sp	Se	Sp	Se	Sp
Oxireductases	78.82	94.90	0.00	82.13	40.45	73.55	82.02	91.22
Transferases	79.02	93.08	85.00	78.97	23.54	93.13	72.08	92.48
Hydrolases	84.19	91.80	49.26	76.09	51.33	82.56	73.34	95.41
Lyases	81.78	97.40	51.59	94.25	38.43	95.24	73.77	95.72
Isomerases	74.66	98.46	0.00	96.37	58.72	94.67	76.23	96.85
Ligases	74.29	99.26	0.00	95.59	54.86	88.30	70.46	97.17
Average	78.79	95.82	30.97	87.73	44.56	87.91	74.65	94.81

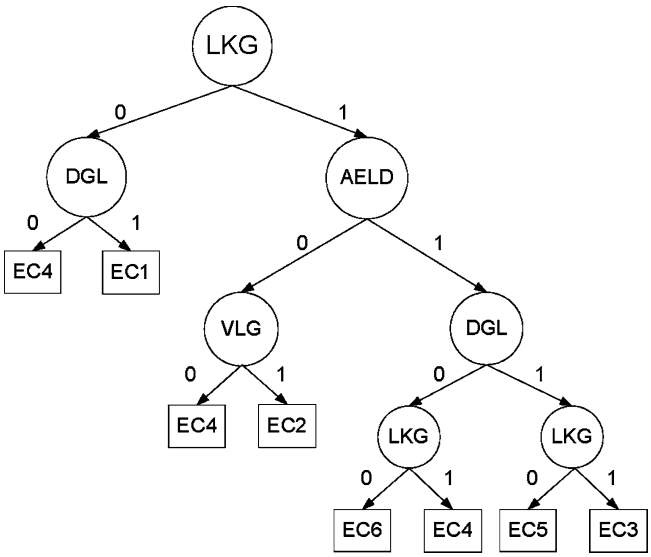

Fig. 1 An example of a decision tree

6 Results and discussion

We have performed tests to measure the quality of our algorithm for the enzyme subset of the PDB previously mentioned. Recall that the subset contains 600 (enzymes) equally divided into 6 classes and the test set contains the others 7,739 enzymes.

To further compare the performance of the GAMDI system, the classification accuracy was also compared against a neural network-based system [16] and a hidden Markov model-based system (HMMER) [4]. Notice that these tools do not intend to discover motifs for classification but to perform straight classification.

The results were obtained using a five-fold cross-validation procedure, in which the input data set is divided into mutually exclusive and exhaustive partitions. Then, the classification algorithm is run five times, in such a way that for each of these runs a different partition is used as the test set and the other four partitions are grouped into an training set. In each of those five runs the classification rate on the evaluation set is computed (see below). The classification rates on the test set are averaged over the five runs, and this average result is reported. We stress that for all systems compared, it was given exactly the same training and test sets in each of the five runs of the cross-validation procedure, making the comparison as fair as possible.

Table 1 shows the best results obtained over the runnings. The final performance was measured using sensitivity (Se) and specificity (Sp). Sensitivity measures the amount of positive cases that are correctly classified, whereas specificity measures the amount of negative cases that the classifier correctly rejected. Notice that when using sensitivity and specificity, a multi-class classification problem is reduced to a double-class. This is accomplished by considering the current class as positive and all other classes as negative. More details about this and other performance measures can be found in [5]. The average sensitivity obtained over all the 51 runnings was 78.79 and the average specificity was 95.82.

In Table 1, it can be observed that for almost all classes, specificity is higher than sensitivity. Due to the way Se and Sp are calculated, the positive class always has much less instances than the negative class (the sum of cases of the remaining five classes, in our implementation). Therefore, as expected, classifiers tend to display a better performance for classes with more instances.

When analyzing Table 1, two comparisons can be done: first comparing GAMDI with MEME, since both tools discover motifs that are used in a decision tree. Afterwards, the overall results can be compared with the other tools that use quite different methods.

For the neural networks and HMMER approaches, specificity values were again always higher than sensitivity. This means that all the compared systems are more efficient to predict when a given protein does not belongs to a class than the opposite. The GAMDI system has performed better than all the other methods, considering the balanced values for both sensitivity and specificity.

Hidden Markov models (HMMER) are very sensitive and dependent to the training set and, in our tests, they had poor generalization capability, one of the evident advantages of the GAMDI system, once the system was conceived to consider this ability.

Comparing GAMDI with MEME, it can be seen that the latter did not find consistent motifs to discriminate from one class to the others. In the other hand, this is an innate ability of GAMDI, accomplished by its fitness function. It is a matter of fact that MEME was not projected for the same purpose as GAMDI but, for the best of our knowledge, it is the tool that most closely can be compared with our system. In short, MEME discovers motifs in a group of proteins,

Table 2 Comparative computational time

System	Time
GAMDI	9.1 h
MEME	24 h
HMMER	2.7 h
Neural networks	Not informed by authors [16]

while GAMDI discovers motifs that discriminate a group of proteins from another.

Both systems, together with the C4.5 algorithm generate a somewhat comprehensive classifier, useful to the biologist. It is possible that those discovered motifs used in the decision trees are related to known specific secondary or tertiary structures. On the other hand, the hidden Markov model is at a lower level of abstraction, not so comprehensible as a decision tree. Finally, the neural networks system is completely obscure for the final user and can be considered a "black box" under the point of view of interpretation of the model.

The classification accuracy can be accessed by multiplying Se by Sp (as proposed in other evolutionary systems for data classification, see [3, 8] for instance). For all classes, the performance of GAMDI approach is comparable with the neural networks system, and both are quite superior to MEME and HMMER approaches. In fact, if we consider the average results, GAMDI is slightly superior to the neural system.

From a computational cost point of view, GAMDI presents an acceptable performance when compared to others, as shown in Table 2. Note that MEME runs on the Cray^{TM} T3E supercomputer at the San Diego Supercomputer Center, and others on a Pentium^{TM} III 550 MHz.

7 Conclusions and future work

We have proposed a system based on a modified GA for motif discovery, aiming to classify unknown proteins. The system was evaluated using the enzymes subset of the PDB.

The knowledge-augmented genetic operators of the GA, specifically designed for the PCP, have played an important role in the positive results achieved. Despite the computational cost, the use of the hill climbing-based operators was beneficial in the sense that it allowed the GA to obtain better motifs (motifs with higher fitness).

As explained, final performance was measured using sensitivity (Se) and specificity (Sp) and similar average results were obtained by the proposed tool (78.79 and 95.82) and the neural network-based system (74.65 and 94.81). Note that a rule generated by a tree is easily understandable by a human while a neural network output is not.

The other results obtained by MEME (30.97 and 87.73) and by HMMER (44.56 and 87.91) demonstrate that classifiers built specifically for proteins classification can obtain better results.

Finally, the results for the enzyme dataset using GAMDI and WEKA strongly suggest that the evolutionary computation method proposed is efficient to find motifs capable of discriminating between groups or proteins.

Future work includes an exhaustive test of the GA control parameters for fine-tuning, development of other specific genetic operators so as to increase efficiency and allow motifs to be described as regular expressions. Also, it is intended to apply this system to other sets of proteins, i.e. transmembrane proteins.

References

1. Abola EE, Sussman JL, Prilusky J, Manning NO (1997) Protein data bank archives of three-dimensional macromolecular structures. Meth Enzymol 277: 556–571
2. Bailey TL, Elkan C (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In: Proceedings of the second international conference on intelligent systems for molecular biology, AAAI Press, Menlo Park, pp 28–36
3. Bojarczuk CC, Lopes HS, Freitas AA (2004) A constrained-syntax genetic programming system for discovering classification rules: application to medical data sets. Artif Intell Med 30(1): 27–48
4. Durbin R, Eddy S, Krogh A, Mitchison G (1998) Biological sequence analysis: probabilistic models of proteins and nucleic acids. Cambridge University Press, Cambridge
5. Hand DJ (1997) Construction and assessment of classification rules. Wiley, New York
6. Holland JH (1975) Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor
7. Lehninger AL, Nelson DL, Cox MM (1998) Principles of biochemistry, 2nd edn. Worth Publishers, New York, pp 134–137
8. Lopes HS, Coutinho MS, Lima WC (1997) An evolutionary approach to simulate cognitive feedback in medical domain. In: Sanchez E, Shibata T, Zadeh LA (eds) Genetic algorithms and fuzzy logic systems. World Scientific, Singapore, pp 193–207
9. Manning AM, Brass A, Goble CA, Keane JA (1997) Clustering techniques in biological sequence analysis. In: Proceedings of the 1st European symposium on principles of data mining and knowledge discovery, pp 315–322
10. McGarrah DB, Judson RS (1993) An analysis of the genetic algorithm method of molecular conformation determination. J Comput Chem 14(11): 1385–1395
11. Murzin AG, Brenner SE, Hubbard T, Chothia C (1995) A structural classification of proteins database for the investigation of sequences and structures. J Mol Biol 247: 536–540
12. Piccolboni A, Mauri G (1998) Application of evolutionary algorithms to protein folding prediction. In: Proceedings of the artificial evolution 97, LNCS, 1363: 123–136
13. Quinlan JR (1993) C4.5: In: Programs for machine learning. Morgan Kaufmann, San Francisco
14. Salamov AA, Solovyev VV (1995) Prediction of protein secondary structure by combining nearest-neighbor algorithms and multiple sequence alignments. J Mol Biol 247: 11–15
15. Wang JTL, Sasha D, Wu CH (2000) Application of neural networks to biological data mining: a case study in protein sequence classification. In: Proceedings of the knowledge discovery in databases conference, pp 305–309
16. Weinert WR, Lopes HS (2004) Neural networks for protein classification. Appl Bioinformatics 3: 41–48
17. Witten IH, Frank E (1999) Data Mining: practical machine learning tools and techniques with java implementations. Morgan Kaufmann, San Francisco