12

A Comparative Study of Machine Learning and Evolutionary Computation Approaches for Protein Secondary Structure Classification

César Manuel Vargas Benítez, Chidambaram Chidambaram,
Fernanda Hembecker and Heitor Silvério Lopes
Bioinformatics Laboratory, Federal University of Technology Paraná (UTFPR)
Curitiba - PR - Brazil

1. Introduction

Proteins are essential to life and they have countless biological functions. Proteins are synthesized in the ribosome of cells following a template given by the messenger RNA (mRNA). During the synthesis, the protein folds into a unique three-dimensional structure, known as native conformation. This process is called protein folding. The biological function of a protein depends on its three-dimensional conformation, which in turn, is a function of its primary and secondary structures.

It is known that ill-formed proteins can be completely inactive or even harmful to the organism. Several diseases are believed to result from the accumulation of ill-formed proteins, such as Alzheimer's disease, cystic fibrosis, Huntington's disease and some types of cancer. Therefore, acquiring knowledge about the secondary structure of proteins is an important issue, since such knowledge can lead to important medical and biochemical advancements and even to the development of new drugs with specific functionality.

A possible way to infer the full structure of an unknown protein is to identify potential secondary structures in it. However, the pattern formation rules of secondary structure of proteins are still not known precisely.

This paper aims at applying Machine Learning and Evolutionary Computation methods to define suitable classifiers for predicting the secondary structure of proteins, starting from their primary structure (that is, their linear sequence of amino acids).

The organization of this paper is as follows: in Section 2 we introduce some basic concepts and some important aspects of molecular biology, computational methods for classification tasks and the protein classification problem. Next, in Sections 3 and 4, we present, respectively, a review of the machine learning and evolutionary computation methods used in this work. In Section 6, we describe the methodology applied to develop the comparison of different classification algorithms. Next, Section 7, the computational experiments and results are detailed. Finally, in the last Section 8, discussion about results, conclusions and future directions are pointed out.

2. Background

2.1 Molecular biology

Proteins are considered the primary components of living beings and they have countless biological functions. Finding the proteins that make up an organism and understanding their function is the foundation of molecular Biology (Hunter, 1993).

All proteins are composed by a chain of amino acids (also called residues) that are linked together by means of peptide bonds. Each amino acid is characterized by a central carbon atom (also known as Cα) to which are attached (as shown in Figure 1) a hydrogen atom, an amino group (NH_2), a carboxyl group (COOH) and a side-chain that gives each amino acid a distinctive function (also known as radical R). Two amino acids form a peptide bond when the carboxyl group of one molecule reacts with the amino group of the other. This process of amino acids aggregation is known as dehydration by releasing a water molecule (Griffiths et al., 2000). All amino acids have the same backbone and they differ from each other by the side-chain, which can range from just a hydrogen atom (in glycine) to a complex heterocyclic group (in tryptophan). The side-chain defines the physical and chemical properties of the amino acids of a protein (Cooper, 2000; Nelson & Cox, 2008).

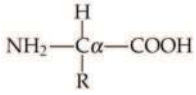


Fig. 1. General structure of an α-amino acid. The side-chain (R element) attached to the Cα defines the function of the amino acid

There are numerous amino acids in the nature, but only 20 are proteinogenic. They are shown in Table 1. The first to be discovered was asparagine, in 1806. The last, threonine, was identified in 1938 (Nelson & Cox, 2008).

To understand the structures and functions of proteins, it is of fundamental importance to have knowledge about the properties of the amino acids, defined by their side-chain. Thus, the amino acids can be grouped into four categories: hydrophobic (also called non-polar), hydrophilic (also called polar), neutral, basic and acid (Cooper, 2000). Kyte & Doolittle (1982) proposed an hydrophobicity scale for all 20 amino acids. See the Table 1 for detailed information about the proteinogenic amino acids.

Polar amino acids can form hydrogen bonds with water and tend to be positioned preferably outwards of the protein, i.e., they are capable to interact with the aqueous medium (which is polar). On the other hand, hydrophobic amino acids tend to group themselves in the inner part of the protein, in such a way to get protected from the aqueous medium by the polar amino acids.

According to this behavior in aqueous solution, one can conclude that the polarity of the side chain directs the process of protein structures formation (Lodish et al., 2000).

From the chemical point of view, proteins are structurally complex and functionally sophisticated molecules. The structural organization of proteins is commonly described into four levels of complexity (Cooper, 2000; Griffiths et al., 2000; Lodish et al., 2000; Nelson & Cox, 2008), in which the upper cover the properties of lower: primary, secondary, tertiary and quaternary structures.

The primary structure is the linear sequence of amino acids. This is the simplest level of organization, it represents only the peptide bonds between amino acids.

Amino acid name	symbol	K&D	Normalised Value	Type
Isoleucine	ILE	+4.5	10.00	Hydrophobic
Valine	VAL	+4.2	9.68	Hydrophobic
Leucine	LEU	+3.8	9.26	Hydrophobic
Phenylalanine	PHE	+2.8	8.20	Hydrophobic
Cysteine	CYS	+2.5	7.88	Hydrophobic
Methionine	MET	+1.9	7.25	Hydrophobic
Alanine	ALA	+1.8	7.14	Hydrophobic
Glycine	GLY	-0.4	4.81	Neutral
Threonine	THR	-0.7	4.49	Neutral
Serine	SER	-0.8	4.39	Neutral
Tryptophan	TRP	-0.9	4.28	Neutral
Tyrosine	TYR	-1.3	3.86	Neutral
Proline	PRO	-1.6	3.54	Neutral
Histidine	HIS	-3.2	1.85	Hydrophilic
Glutamine	GLN	-3.5	1.53	Hydrophilic
Asparagine	ASN	-3.5	1.53	Hydrophilic
Glutamic acid	GLU	-3.5	1.53	Hydrophilic
Aspartic acid	ASP	-3.5	1.53	Hydrophilic
Lysine	LYS	-3.9	1.10	Hydrophilic
Arginine	ARG	-4.0	1.00	Hydrophilic

Table 1. Kyte and Doolittle (K & D) hydrophobicity scale and the normalized scale used in the computational experiments

The secondary structure of a protein refers to the local conformation of some part of a three-dimensional structure. There are, basically, three main secondary structures: α-helices (Pauling et al., 1951a), β-sheets (Pauling et al., 1951b) and turns (Lewis et al., 1973). In the structure of an α-helix, the backbone is tightly turned around an imaginary helix (spiral) and the side-chains of the amino acids protrude outwards the backbone (Figure 2(a)). The β-sheet is formed by two or more polypeptide segments of the same molecule, or different molecules, arranged laterally and stabilized by hydrogen bonds between the NH and CO groups (Figure 2(b)). Adjacent polypeptides in a β-sheet can have same direction (parallel β-sheet) or opposite directions (antiparallel β-sheet). Functionally, the antiparallel β-sheets are present in various types of proteins, for example, enzymes, transport proteins, antibodies and cell-surface proteins (Branden & Tooze, 1999). Turns are composed by a small number of amino acids and they are usually located in the surface of proteins forming folds that redirect the polypeptide chain into the protein. They allow large proteins to fold in highly compact structures.

Secondary structures can be associated through side-chain interactions to motifs (Branden & Tooze, 1999; Griffiths et al., 2000; Nölting, 2006). Motifs are patterns often found in three-dimensional structures that perform specific functions. For instance, the helix-turn-helix motif is important in DNA-protein interactions.

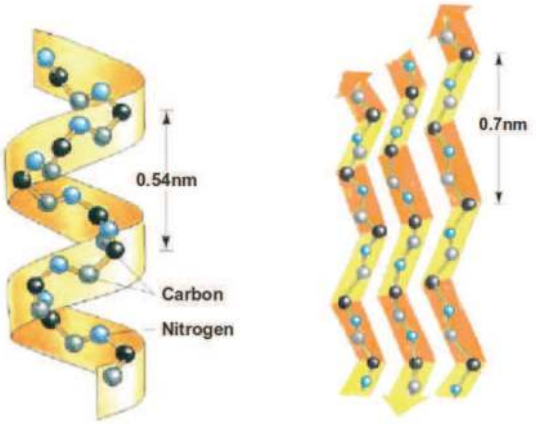


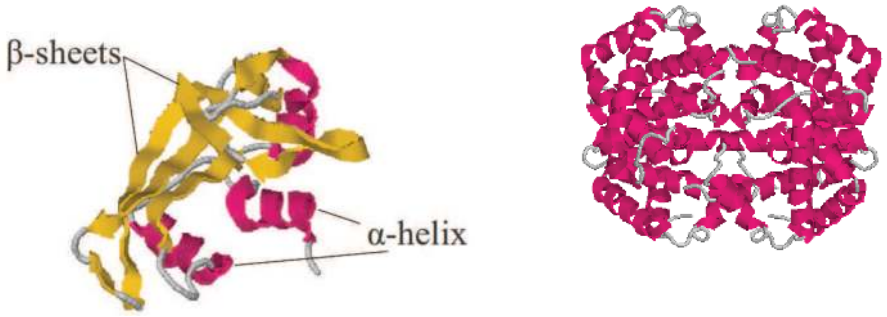
Fig. 2. α-helix (a) and β-sheet (b) structures. Adapted from (Alberts et al., 2002)

The tertiary structure represents the conformation of a polypeptide chain, i.e. the three-dimensional arrangement of the amino acids. The tertiary structure is the folding of a polypeptide as a result of interactions between the side chains of amino acids that are in different regions of the primary structure. Figure 3(a) shows an example of tertiary structure, where one can observe the presence of two secondary structures: α-helix and β-sheet.
Finally, the arrangement of three-dimensional structures constitutes quaternary structure (as shown in Figure 3(b)). Figures 3(a) and 3(b) were drawn using RasMol [1] from PDB files (see Section 2.2).
The Proteins can be classified into two major groups, considering higher levels of structure (Nelson & Cox, 2008): fibrous and globular proteins. Both groups are structurally different: fibrous proteins consist of a single type of secondary structure; globular proteins have a nonrepetitive sequence and often contain several types of secondary structure. Helices are the most abundant form of secondary structure in globular proteins, followed by sheets, and in the third place, turns (Nölting, 2006).

2.2 Protein databases and classification

Finding protein functions has been, since long ago, an important topic in the Bioinformatics community. As mentioned in Section 2, the function of a protein is directly related to its structure. Due to its great importance for Medicine and Biochemistry, many research has been done about proteins (including the many genome sequencing projects) and, consequently, many information is available. There are many resources related to protein structure and function. Table 2 lists some protein databases. Basically, the protein databases can be classified into two classes: sequence and structure databases.

[1] RasMol is a molecular visualization software. Available at http://www.rasmol.org



(a) (b)

Fig. 3. Tertiary structure of Ribonuclease-A (a) and quaternary structure of Hemoglobin (b).

Database	Description	Web Address
PDB	repository of protein structures	http://www.pdb.org
UniProtKB/TrEMBL	repository of protein amino acid sequences, name/description, taxonomic data and citation information	http://www.uniprot.org/
PIR	protein sequence databases	http://pir.georgetown.edu/
PROSITE	documentation entries describing sequence motif definitions, protein domains, families and functional patterns	http://www.expasy.org/prosite/
PRINTS	Fingerprints information on protein sequences	http://www.bioinf.man.ac.uk/dbbrowser/
BLOCKS	Multiple-alignment blocks	http://blocks.fhcrc.org/
eMOTIF	protein motif database, derived from PRINT and BLOCKS	http://motif.stanford.edu/emotif/
PRODOM	protein domain databases	http://protein.toulouse.inra.fr/prodom.html
InterPro	protein families and domains	http://www.ebi.ac.uk/interpro/

Table 2. Some important protein databases

In this work we used the Protein Data Bank (PDB) (Berman et al., 2000; Bernstein et al., 1977) that is an international repository of three-dimensional structure of biological

macromolecules. The data is, typically, obtained by X-ray crystallography (Drenth, 1999; Sunde & Blake, 1997) or NMR spectroscopy (Nuclear Magnetic Resonance) (Jaroniec et al., 2004; Wüthrich, 1986). Almost all protein structures known today are stored in the PDB (Bernstein et al., 1977).

Despite the growing number of protein sequences already discovered, only a few portion of them have their three-dimensional and secondary structures unveiled. For instance, the UniProtKB/TrEMBL repository (The UniProt Consortium, 2010) of protein sequences has currently around 13,89 million records (as in March/2011), and the PDB registers the structure of only 71,794 proteins. This fact is due to the cost and difficulty in unveiling the structure of proteins, from the biochemical and biological point of view. Therefore, computer science has an important role here, proposing models and methods for studying the Protein Structure Prediction problem (PSP).

There are two basic approaches that are used to the prediction of protein functions: prediction of the protein structure and then prediction of function from the structure, or else, classifying proteins and supposing that similar sequences will have similar functions (Tsunoda et al., 2011). Several approaches to solve the PSP exist, each addressing the problem by using a computational method to obtain optimal or quasi-optimal solutions, such as Molecular Dynamics with *ab initio* model (Hardin et al., 2002; Lee et al., 2001), neural nets (Yanikoglu & Erman, 2002) and evolutionary computation methods with lattice (Benítez & Lopes, 2010; Lopes, 2008; Scapin & Lopes, 2007; Shmygelska & Hoos, 2005) and off-lattice (Kalegari & Lopes, 2010) models.

However, there is no consensus about protein classification which is done using different properties of proteins through several approaches. Many computational techniques have been used to classify proteins into families, such as structural transformations (Ohkawa et al., 1996), data compression (Chiba et al., 2001), genetic programming (Koza, 1996), Markov chains (Durbin et al., 1998) and neural networks (Wang & Ma, 2000; Weinert & Lopes, 2004).

Other papers focus on motifs discovery, as a starting step for protein classification. For instance, (Tsunoda & Lopes, 2006) present a system based on a genetic algorithm that was conceived to discover motifs that occur very often in proteins of a given family but rarely occur in proteins of other families. Also, Tsunoda et al. (2011) present an evolutionary approach for motif discovery and transmembrane protein classification, named GAMBIT. Techniques of clustering for sequences analysis were presented by (Manning et al., 1997). Chang et al. (2004) proposed a Particle Swarm Optimization (PSO) approach to motif discovery using two protein families from the PROSITE.

Wolstencroft et al. (2006) describes the addition of an ontology that captures human understanding of recognizing members of protein phosphatases family by domain architecture as an ontology. G.Mirceva & Davcev (2009) present an approach based on Hidden Markov Models (HMMs) and the Viterbi algorithm to domain classification of proteins. Davies et al. (2007) present an hierarchical classification of G protein-coupled receptors (GPCRs) The GPCRs are a common target for therapeutic drugs (Klabunde & Hessler, 2002).

3. Machine learning methods

In this section, we focus on the Machine Learning (ML) algorithms applied to the classification of a secondary protein data set. Most of the methods discussed in this section can be considered as important supervised ML techniques. It is frequently cited in the literature that the efficiency of ML algorithms can be quite different from data set to data set. Therefore, it is usual to test the data set with many different ML algorithms. For this purpose,

Weka[2] (Witten et al., 2011) is an appropriate and a flexible tool which contains a collection of state-of-the-art ML algorithms and data preprocessing tools. It allows users to test the algorithms with new data sets quickly.
It is not an easy task to select which algorithm is the most suitable for a specific domain or problem. Also, as there are many algorithms that use numerical data, an expert is needed to tune the data. In this context, the comparison of many different algorithms is not an easy task. Such a comparison can be performed by statistical analysis of the accuracy rate obtained from trained classifiers on some specific data set (Kotsiantis, 2007). To support this comparison and to evaluate the quality of the methods, three main measures can be considered (Mitchell, 1997):

- The classification rate of the training data
- The correct classification rate of some test data
- The average performance using cross validation

The most well-known classification algorithms are grouped in the Weka workbench, including Bayesian classifiers, Neural networks, Meta-learners, Decision trees, Lazy classifiers and rule-based classifiers (Witten et al., 2011). Bayesian methods include Naïve Bayes, complement Naïve Bayes (CNB), multinomial Naïve Bayes, Bayesian networks and AODE. Both decision trees and rule-based classifiers belong to the category of logic-based supervised classification algorithms (Kotsiantis, 2007). Decision trees algorithms include several variants, such as NBTree, ID3, J48 (also known as C4.5) and alternative decision trees (ADTree). PART, decision tables, Rider, JRip, and NNge are included in the group of rule learners. Lazy Bayesian rules (LBR), Instance-Based learning schemes (IB1 and IBk), Kstar, and Locally-Weighted Learning (LWL) are part of the lazy learning algorithms.
Besides these basic classification learning algorithms, there are some meta-learning schemes such as LogitBoost, MultiBoostAB, and ADABoost. These boosting algorithms enable users to combine instances of one or more of the above-mentioned algorithms. In addition to these all algorithms, Weka workbench includes neural networks methods such as Multiplayer Perceptron (MLP), Sequential Minimal Optimization algorithm (SMO), Radial Basis Function (RBF) network, and logistic and voted perceptron (Witten et al., 2011). In the following subsections, we discuss specific aspects and present a brief comparison of some classification methods mentioned previously and used in the protein classification task of this work.

3.1 Bayesian methods

Bayesian methods are the most practical approaches amongst many learning algorithms and those that provide learning based on statistical approaches. These methods are characterized by induction of probabilistic networks from the training data (Kotsiantis, 2007). In the Weka workbench there are several methods, such as: CNB, NaïveBayes, NaïveBayesSimple and AODE. Bayesian methods emerged as alternative approaches for decision trees and neural networks and, at the same time, being competitive with them for real-world applications (John & Langley, 1995). However, it is known that, to apply such methods, prior knowledge of many probabilities is required (Mitchell, 1997). Naïve Bayes (NB) is one of the learning algorithms widely applied to classification tasks. The NB classifier is the most effective algorithm of the Bayesian group of algorithms, and it can easily handle unknown or missing values (Mitchell, 1997). The main advantage of the NB is that it does not require a long

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

time to train the classifier with the data set. This classifier is based on the assumption of attribute independence. Some Bayesian variants have tried to alleviate this assumption, such as the NBTree (Zheng & Webb, 2000) algorithm, which has a high computational overhead. To improve the prediction accuracy without increasing the computational cost, Webb et al. (2005) developed a variant algorithm of NB called AODE (Aggregating One-Dependence Estimators) that uses a weaker attribute independence assumption than NB. NB classifier traditionally relies on the assumption that the numerical attributes are generated by a single Gaussian distribution. However, this is not always the best approximation for the density estimation on continuous attributes. Following this direction, John & Langley (1995) proposed a new approach described as flexible Bayes, in which a variety of nonparametric density estimation methods were used instead of Gaussian distribution. This approach is implemented through NaïveBayesUpdateable class in the Weka tool. Bayesian networks (BN) methods use several search algorithms that works under conditions of uncertainty. All BN learners are considered as slow and not suitable for large data sets (Cheng et al., 2002). Unlike decision trees and neural networks, the main aspect of this method is that it obtains prior information of the data set from the structural relationships among the features (Kotsiantis, 2007).

3.2 Decision trees

Basically, the methods of this category classify data by building decision trees, in which each node represents a feature in an instance and each branch represents the value of a node. They are heuristic, non-incremental, and do not use any world knowledge. In this category, there are many learning methods, amongst of which, ID3, J48 and C4.5 are most well-known (Kotsiantis, 2007). Not only these algorithms are based on decision trees. There are also some hybrid versions like NBTree, LMTree, RandomForest and ADTree. In general, these versions have competitive performance with the traditional C4.5 algorithm. The earliest version of ID3 was developed by Quinlan (1993) and its improved version is C4.5 (Martin, 1995). ID3 allows to work with errors in the training data, as well as missing attribute values. This method uses a hill-climbing strategy to search in the hypothesis space, so as to find out a decision tree that correctly classifies the training data. This learning method can handle noisy training data and is less sensitive to errors of individual training examples (Mitchell, 1997). From the decision trees, during the learning process, a set of rules in the disjunctive form are obtained by traversing the different possible paths in the entire tree. A limitation of the ID3 algorithm is that it cannot guarantee the optimal solution. Another improved version of the algorithm, J48, implements Quinlan's C4.5 algorithm by generating a pruned or an unpruned decision tree (Witten et al., 2011). The NBTree algorithm was proposed by Kohavi (Kohavi, 1996) to overcome the accuracy problem encountered with both Naïve-Bayes and decision trees in small data sets. NBTree is a hybrid algorithm which induces a mix of decision-tree and Naïve-Bayes classifiers. Eventually, this algorithm outperforms both C4.5 and Naïve-Bayes. To predict numeric quantities, Landwehr et al. (2005) introduced a new learning method combining tree induction methods and logistic regression models into decision trees. This method is denominated logistic model trees (LMTree). LMTree produces a single tree which is not easily interpretable, but better than multiple trees. This algorithm achieved performance higher than decision trees with C4.5 and logistic regression.

RandomForest is a classifier consisting of a collection of tree-structured classifiers in which each tree depends on the values of a random vector, sampled independently and with the same distribution for all trees in the forest (Breiman, 2001).

Freund & Schapire (1999) presented a new type of classification algorithm, combining decision trees and boosting, called alternative decision trees (ADTree). An important feature of the ADTree algorithm is the measure of confidence or classification margin. This learning algorithm was compared with other improved version of C4.5 with boosting, denominated as C5.0. According to the experiments, it can be realized that the ADTree is competitive with C5.0 and generate easily interpretable small rules.

3.3 Neural Networks

Neural Network (NN) learning methods are robust to errors in the training data. They provide a good approximation to real-valued, discrete-valued, and vector-valued target functions. High accuracy and high speed rate of classification are some relevant aspects of NN classifiers (Kotsiantis, 2007). Algorithms like Multiplayer Perceptron (MLP), SMO, RBFNetwork and Logistic are part of the Weka workbench. A RBF neural networks is a particular NN constructed from spatially localized kernel functions, and uses a different error estimation and gradient descent function called the radial basis function (RBF). This method uses a cross-validation technique to stop the training which is not present in other NN algorithms (Mitchell, 1997). Platt (1998) proposed an improved algorithm for training support vector machines (SVMs) called Sequential Minimal Optimization SMO. The results obtained for real-world test sets showed that the SMO is 1200 times faster than linear SVMs and 15 times faster than non-linear SVMs.

3.4 Meta-learning methods

Most of the methods of this category such as boosting, bagging and wagging are common committee learning approaches that reduce the classification error from learned classifiers. Boosting is the one of the most important recent advancements in classification algorithms, since it can improve dramatically their performance (Friedman et al., 2000). It is also known as machine learning meta-algorithm or discrete AdaBoost. AdaBoost, as a boosting algorithm, can efficiently convert a weak learning algorithm into a short learning algorithm. Furthermore, it is an adaptive behavior with error rates of the individual weak hypotheses (Freund & Schapire, 1999). AdaBoost calls a given weak learning algorithm repeatedly in a series of rounds and trains the classifiers by over-weighting the training samples that were misclassified in the next iteration. A complete algorithm description can be found in Friedman et al. (2000). One of the important properties of the ADABoost algorithm is the identification of outliers which are either mislabeled in the training data or inherently ambiguous and hard to categorize (Freund & Schapire, 1999). Many other developments on AdaBoost resulted in variants such as Discrete AdaBoost, Real AdaBoost, LPBoost and LogitBoost. Both ADABoost and bagging generic techniques can be employed together with any baseline classification technique. Wagging is variant of bagging, that requires a base learning algorithm capable of using training cases with differing weights (Webb, 2000). MultiBoosting is an extension technique of AdaBoost with wagging. It offers a potential computational advantage over AdaBoost (Webb, 2000).

3.5 Rule-based methods

Rules can be extracted from the data set using many different machine learning algorithms. The IF-THEN rules is a convenient way to represent the underlying knowledge present in the data set, which can be easily understood by domain experts. Among a variety of rule-based methods that were investigated, decision trees and separate-and-conquer strategy

are considered the most important (Frank & Witten, 1998). Based on these approaches, emerged two dominant rule-based learning methods: C4.5 (Quinlan, 1993) and RIPPER (Cohen, 1995). In fact, RIPPER is an optimized algorithm which is very efficient in large samples with noisy data, and produce error rates lower than or equivalent to C4.5 (Cohen, 1995). In the Weka workbench, rule-based classifiers have many learning algorithms, including PART, DecisionTable, Ridor, JRip and NNge. Decision tables, which are simple space hypotheses, are represented by DTM (Decision Table Majority). Kohavi (1995) evaluated the power of decision tables through Inducer DTM (IDTM). IDTM, on some data sets, obtained comparable performance accuracy as C4.5. JRip implements a propositional rule learner called Repeated Incremental Pruning to Produce Error Reduction (RIPPER), proposed by Cohen (1995). Frank & Witten (1998) proposed a new approach, by combining the paradigms of decision trees and separate-and-conquer, called PART. PART is based on the repeated generation of partial decision trees in a separate-and-conquer manner. Not only accuracy of a learning algorithm, but also the size of a rule set resulted from the learning process is important. The size of a rule strongly influences on the degree of comprehensibility. Rule sets produced by PART are generally smaller as C4.5 and more accurate than RIPPER (Frank & Witten, 1998). Another algorithm that is part of Weka workbench is the Non-Nested Generalized Exemplars (NNge), proposed by Martin (1995). NNge generalizes exemplars without nesting (exemplars contained within one another) or overlaping. By generalization, examples in the data set that belongs to the same class are grouped together. When tested against domains containing both large and small disjuncts, NNge performs better than C4.5. NNge performs well on data sets that combine small and large disjuncts, but it performs poorly in domains with a high degree of noise (Martin, 1995).

3.6 Lazy methods

Instance-based methods are considered as lazy learning methods because the classification or induction process is done only after receiving a new instance or a training example. Learning process will be started on the stored examples only after when a new query instance is encountered (Mitchell, 1997). Nearest Neighbor algorithm is the one of the most basic instance-based methods. The instance space is defined in terms of Euclidean distance. However, since Euclidean distance is inadequate for many domains, several improvements were proposed to the instance-based nearest neighbor algorithm which are known as IB1 to IB5 (Martin, 1995). Zheng & Webb (2000) proposed a novel algorithm called Lazy Bayesian Rule learning algorithm (LBR) in which lazy learning techniques are applied to Bayesian tree induction. Error minimization was maintained as an important criteria in this algorithm. Experiments done with different domains showed that, on average, LBR overcomes mostly all other algorithms including Naïve Bayes classifier and C4.5. The Locally Weighted Learning algorithm (LWL) is similar to other lazy learning methods, however it behaves differently when classifying a new instance. LWL algorithm constructs a new Naïve Bayes model using a weighted set of training instances (Frank et al., 2003). It empirically outperforms both standard Naïve Bayes as well as nearest-neighbor methods on most data sets tested by those authors.

4. Gene expression programming and GEPCLASS

Gene expression programming (GEP) is proposed by Ferreira (2001), and it has features of both genetic algorithms (GAs) and genetic programming (GP). The basic difference between the three algorithms is the way the individuals are defined in each algorithm. In the

traditional GAs individuals are called chromosomes and are represented as linear binary strings of fixed length. On the other side, in GP, individuals or trees are non-linear entities of different sizes and shapes. On the other hand, in GEP, the individuals are encoded as linear strings of fixed length (chromosomes) which are afterwards expressed as non-linear entities of different sizes and shapes (simple diagram representations or expression trees (ETs)) (Ferreira, 2001). In fact, in GEP, chromosomes are simple, compact, linear and relatively small entities, that are manipulated by means of special genetic operators (replication, mutation, recombination and transposition). ETs, in turn, are the phenotypical representation of the chromosome. Unlike genetic algorithms, selection operates over ETs (over the phenotype, not the genotype). During the reproduction phase, only chromosomes are generated, modified and transmitted to the next generations. The interplay of chromosomes and ETs allows to translate the language of chromosomes into the language of ETs. The use of varied set of genetic operators introduce genetic diversity in GEP populations always producing valid ETs. In the same way as other evolutionary algorithms, in GEP, the initial population must be defined either randomly or using some previous knowledge collected from the problem. Next, chromosomes are expressed into ETs which will be then evaluated according to the definition of the problem resulting in a fitness measure. During the iteration process, the best individual(s) is(are) kept and the rest are submitted to a fitness-based selection procedure. Selected individuals naturally go through modifications by means of genetic operators leading to a new generation of individuals. The whole process is repeated until a stopping criterion is met (Weinert & Lopes, 2006). The structural organization of GEP genes are based on Open Reading Frames (ORF), a biological terminology. Although the length of genes is constant, the length of ORFs are not. GEP genes are composed of a head that contain symbols that represent both function and terminals, and a tail that contains only terminals. GEP chromosomes are basically formed by more than one gene of different lengths. Unlike GP, in which an individual of the population is modified by only one operator at a time, in GEP, an individual may be changed by one or several genetic operators. Besides the genetic operations like replication, mutation and recombination, GEP includes operations based on transpositions and insertion of elements.

In this work, we used the GEPCLASS system[3] that was specially developed for data classification with some modifications regarding the original GEP algorithm (Weinert & Lopes, 2006). Prior implementing data classification using evolutionary algorithms, it is necessary to definet whether an individual represents a single rule (Michigan approach) or a complete solution composed by a set of rules (Pittsburg approach) (Freitas, 1998). This system can implement both approaches, either by an explicit decision of a user, or allowing the algorithm decide by itself which one is the most suitable for a given classification task, during the evolutionary process. GEPCLASS can manage both continuous and categorical (nominal) attributes in the data set. If a given attribute is nominal, GEPCLASS uses only $=$ or $\neq$ as relational operators. Otherwise, if the attribute is continuous or ordered categorical, all relational operators can be used.

Figure 4 shows a 2-genes chromosome with different lengths for heads and tails. In the chromosome part, upward arrows show the points delimiting the coding sequence of each gene. This chromosome transformed into an ET and, later, to a candidate rule. GEPCLASS uses variable-length chromosomes that can have one or more genes. Genes within a given chromosome are of the same size. Using chromosomes with different lengths in the population can introduce healthy genetic diversity during the search.

[3] Freely available at: *http://bioinfo.cpgei.ct.utfpr.edu.br/en/softwares.htm*

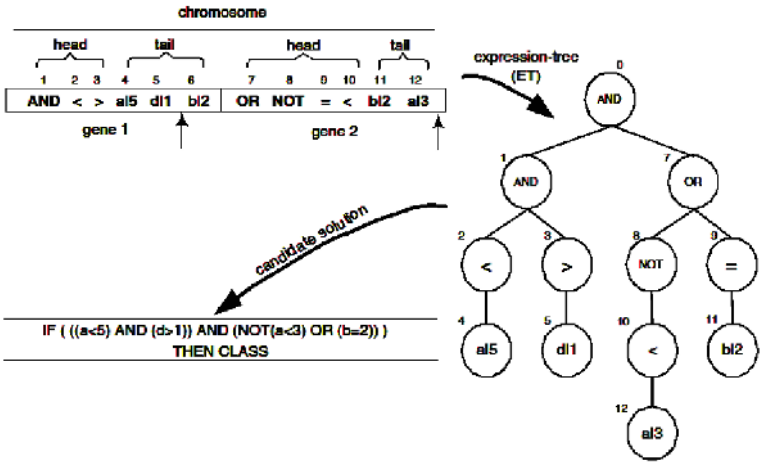


Fig. 4. Example of chromosome structure, expression tree (ET) and rule in GEPCLASS. Adapted from (Weinert & Lopes, 2006).

5. Hidden Markov models

A hidden Markov model (HMM) has an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observations (Rabiner, 1989). A HMM can be visualized as a finite state machine composed of a finite set of states, $a_1, a_2, \ldots, a_n$, including a beginning and an ending states (G.Mirceva & Davcev, 2009). The HMM can generate a protein sequence by emitting symbols as it progresses through a series of states. Any sequence can be represented by a path through the model. Each state has probabilities associated to it: the transition and the emission probabilities. Although the states are hidden, there are several applications in which the states of the model can have physical meaning.

HMMs offer the advantages of having strong statistical foundations that are well-suited to natural language domains and they are computationally efficient (Seymore et al., 1999). Therefore, HMMs have been used for pattern recognition in many domains and, in special, in Bioinformatics (Durbin et al., 1998; Tavares et al., 2008).

6. Methodology

6.1 The data set and sequence encoding

A number of records of human globular proteins was selected and downloaded from the PDB. The original files were scanned and all annotated secondary structures were extracted by using a parser developed in Java programming language.

These primary sequences extracted from the files had a variable length from 2 to 29 amino acids, and they were divided into five classes, following the PDB nomenclature: $HELIX$, $SHEET0$, $SHEET1$ and $SHEET2$ and $TURN$ with 1280, 284, 530, 498 and 119 sequences, respectively. $HELIX$, $SHEET0$ ($SHEET1$ and $SHEET2$) and $TURN$ represent α-helices, β-sheets and turns, respectively. In order to properly train the classifiers it is also necessary a "negative" class. That is, a class dissimilar to the others that represent no secondary structure.

This was accomplished by creating a $NULL$ class of 3000 variable-sized sequences. Therefore, the database created for this work had 5711 records, with six unbalanced classes.
The natural encoding of the protein sequence is a string of letters from the alphabet of letters representing the 20 proteinogenic amino acids. However, this encoding is not appropriate for some classification algorithms. Thus, another encoding was proposed, converting the string of amino acid symbols into a real-valued vector, by using a physico-chemical property of the amino acids, as suggested by (Weinert & Lopes, 2004). This was accomplished using the Kyte and Doolittle hydrophobicity scale. The real-valued vector was normalized in the range 1.00 – 10.00 (as shown in Table 1).

6.2 Computational methods

Three different computational approaches were used in this work. First, we applied several machine learning algorithms using the Weka workbench, as mentioned before. The algorithms in Weka were grouped into Bayesian, neural networks, meta-learners, trees, lazy-learners and rule-based. The input file for this algorithms was formatted to Attribute-Relation File Format (ARFF). This is an ASCII text file that describes a set of instances sharing a set of attributes.
An ARFF file has two sections: the header and data information. The header of the ARFF file contains the name of the relation, a list of attributes and their types. The convention used for data was: p_1, p_2,..., p_{29}, corresponding to positions in the amino acid sequence from 1 to 29, followed by a nominal class attribute, that identifies the class of each instance ($HELIX$, $SHEET0$, $SHEET1$, $SHEET2$, $TURN$ and $NULL$). Where p_i are real-valued, as explained in Section 6.1.
A second approach used was HMMs. To test this approach, we have used the HMM package for Weka[4]. Similar to other input files, the input file for this approach was also formatted as ARFF. The HMM classifiers only work on a sequence of data which in Weka is represented as a relational attribute. Data instances have a single nominal class attribute and a single relational sequence attribute. The instances in this relational attribute consist of single nominal data instances using the natural encoding of the protein sequence, i.e., the string of letters representing the amino acid sequence (for example, "GLY, TRP, LEU, ..., LEU").
Finally, Gene Expression Programming (GEP) was used for generating classification rules by means of the software GEPCLASS (Weinert & Lopes, 2006). The input file to GEP is similar to the ARFF file that was used before, but without header information, just with the real-valued data instances as detailed in the section 6.1.

7. Experiments and results

The main objective of this work is to compare the performance of ML algorithms and evolutionary computation approaches for protein secondary structure classification. The main motivation to test several algorithms is to identify the most suitable one for protein secondary structure classification.
The performance of the methods are measured according to their predictive accuracy and other statistical parameters drawn from confusion matrix. The processing time and memory load were not considered for the experimental analysis.
The training/testing methodology includes a 10-fold cross-validation (Kohavi, 1995) in which the data set is divided into 10 parts. In the first round, one part is used for testing and the

[4] Available at *http://www.doc.gold.ac.uk/ mas02mg/software/hmmweka/*

remaining, nine parts are used for training. This procedure is repeated until each partition has been used as the test set. The reported result is the weighted average of the 10 runs. The purpose of cross-validation is to avoid biased results when a small sample of data is used.
The outcome of a classifier can lead to four different results, according to what was expected and what, in fact, was obtained. Therefore, when classifying unknown instances, four possible outcomes can be computed:

- tp: true positive: the number of instances (proteins) that are correctly classified, i.e., the algorithm predicts that the protein belongs to a given class and the protein really belongs to that class;
- fn: false positive: the number of instances that are wrongly classified, i.e., the algorithm predicts the protein that belongs to a given class but it does not belong to it;
- tn: true negative: the number of instances that are correctly classified as not belonging to a given class, i.e., the algorithm predicts that the protein does not belong to a given class, and indeed it does not belong to it.
- fp: false negative: the number of instances of a given class that are wrongly classified, i.e., the algorithm predicts that the protein does not belong to a given class but it does belong to it.

Combining the above-cited four outcomes obtained from a classifier, we have then calculated, for each class, metrics commonly used in ML: sensitivity (Se), specificity (Sp), the predictive accuracy and the Matthews Correlation Coefficient (MCC) (Matthews, 1975), defined in Equations 1, 2, 3 and 4, respectively. Then, the weighted average of these metrics was calculated. The results are shown in Table 3. The best results, according to these metrics, are shown in bold in the table.

$$Se = \frac{tp}{tp + fn} \tag{1}$$

$$Sp = \frac{tn}{tn + fp} \tag{2}$$

$$Accuracy = \frac{tp + tn}{tp + fp + fn + tn} \tag{3}$$

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp) \times (tp + fn) \times (tn + fp) \times (tn + fn)}} \tag{4}$$

The visual comparison of performance of the classifiers was done using a ROC (Receiver Operating Characteristics) plot (Fawcett, 2006). The ROC plot is a useful technique for visualizing and comparing classifiers and is commonly used in decision making in ML, data mining and Bioinformatics (Sing et al., 2005; Tavares et al., 2008). It is constructed using the performance rate of the classifiers. The ROC analysis can be used for visualizing the behavior of diagnostic systems and medical decisions (Fawcett, 2006). In a ROC plot axes x and y are defined as (1-Sp) and Se, respectively. These axes can be interpreted as the relative trade-offs between the benefits and costs of a classifier. Thus, each classifier correspond to their (1-Sp, Se) pairs. The best prediction would be that lying as close as possible to the upper left corner, representing 100% of sensitivity (no fn) and specificity (no fp). Figure 5 shows the ROC plot for the classifiers evaluated in this work. It should be noted that some of the classifiers have achieved almost the same performance. Therefore, some points are superimposed in

Group	Method	*Se*	*Sp*	*MCC*	Accuracy rate (%)
Bayes	NaiveBayes	0.601	0.910	0.47	56.58
	AODE	0.688	0.292	0.57	68.78
Neural Net	MLP	0.599	0.774	0.39	59.87
	Logistic	0.559	0.604	0.21	55.90
	SMO	**0.792**	0.792	**0.67**	**79.16**
Meta	ADABoost	0.774	0.875	0.65	77.43
	logitBoost	0.769	0.879	0.65	76.89
	Bagging	0.741	0.854	0.60	74.07
Trees	J48	0.719	0.87	0.58	71.88
	RandomForest	0.774	0.874	0.65	77.41
	RepTree	0.717	0.868	0.58	71.72
Lazy	IB1	0.727	0.869	0.59	72.74
	IBk	0.683	0.86	0.53	68.33
	Kstar	0.730	0.874	0.59	72.96
Rules	Jrip	0.667	0.739	0.45	66.70
	PART	0.726	0.862	0.59	72.63
	Ridor	0.670	0.831	0.50	66.99
HMM		0.629	**0.919**	0.54	62.86
GEP		0.750	0.630	0.320	75.43

Table 3. Classifier performance

the graph. The top classifiers are identified in the ROC space: SMO (Sequential Minimal Optimization algorithm), ADABoost (ML meta-algorithm) and RandomForest (collection of tree-structured classifiers).

8. Conclusion and future works

Prediction of secondary structure protein has become an important research area in Bioinformatics. Since the beginning, similar sequences of proteins are used to predict the function of new proteins. In this work, several methods from ML and evolutionary computation for classifying protein secondary structures were compared. Considering sensitivity and specificity alone, SMO and NaiveBayes achieved the highest values, meaning that the first was good to detect secondary structures when they are present, and the latter was good to classify sequences that are not secondary structures. The highest sensitivity value was below 0.8, suggesting the presence of semantic noise in the data set given by the inherent variability of amino acid sequences in the secondary structure of proteins.

According to the results shown in Table 3 and considering the accuracy rate, we can conclude that SMO, Meta classifiers (ADABoost and logitBoost), RandomForest, and GEP methods achieved better performances, all above 75% of accuracy. These results can be considered very expressive, taking into account the difficulty of the classification problem, imposed, as mentioned before, mainly by biological variability of the secondary structure of proteins.

It is also possible to observe that the ROC plot shows the differences between methods more clearly than Table 3, when considering both, sensitivity and specificity. For instance, it is possible to observe that the HMM and NaiveBayes achieved the lowest number of false

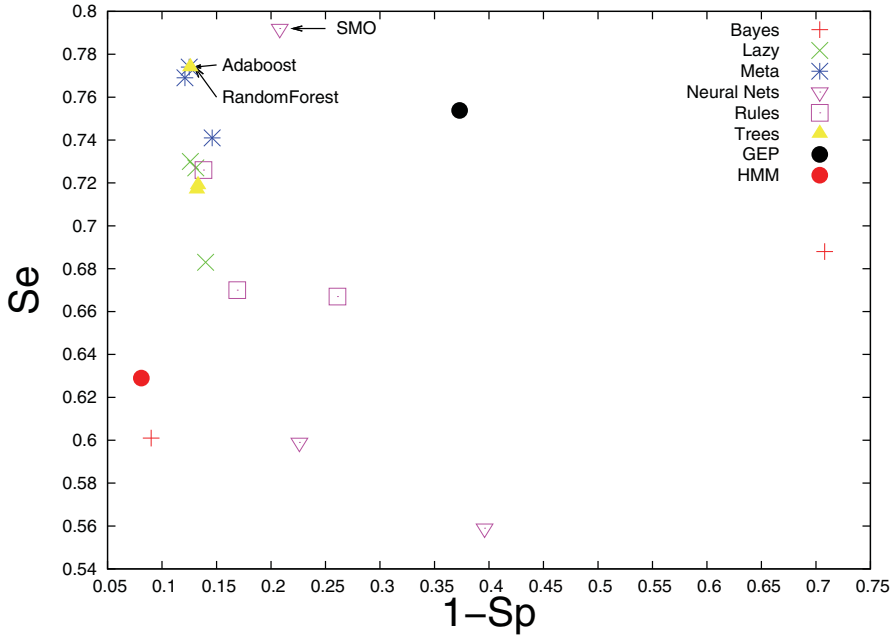


Fig. 5. ROC plot

negatives (highest specificity), however, they perform badly considering sensitivity. In the upper-left corner are the three best-performing methods. However, GEP is slightly distant from them, making clear that that considering only the accuracy rate may be misleading. The analysis of the classifiers using the MCC leads to similar results as the ROC curve. Therefore, based on our results, we can conclude that ROC and MCC are the best way to analyze the performance of methods in this classification problem.

It is important to mention that no effort was done to fine-tune parameters of any method. In all cases, the default parameters of the methods were used. However, it is a matter of fact that adjusting parameters of classifiers (for instance, in GEP and HMM methods), the overall performance can be significantly improved. On the other hand, such procedure could lead to a biased comparison of classifiers.

Although SMO and Meta-learners achieved the best classification performance, it should be highlighted the importance of rule-based methods (including GEP), since a set of classification rules are more comprehensive and expressive to humans than other type of classification method expressed by numbers.

Future work will include the use of hybrid techniques incorporating ML and evolutionary computation methods, as well as hierarchical classification methods.

9. Acknowledgements

This work was partially supported by the Brazilian National Research Council (CNPq) under grant no. 305669/2010-9 to H.S.Lopes; and a doctoral scholarship do C.M.V. Benítez from CAPES-DS.

10. References

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. & Walter, P. (2002). *Molecular Biology of The Cell*, Garland Science, New York, USA.

Benítez, C. & Lopes, H. (2010). Hierarchical parallel genetic algorithm applied to the three-dimensional hp side-chain protein folding problem, *Proc. of IEEE International Conference on Systems, Man and Cybernetics*, IEEE Computer Society, pp. 2669–2676.

Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shindyalov, I. & Bourne, P. (2000). The protein databank, *Nucleic Acids Research* 28(1): 235–242.

Bernstein, F., Koetzle, T., William, G., Meyer, D., Brice, M. & Rodgers, J. (1977). The protein databank: A computer-based archival file for macromolecular structures, *Journal of Molecular Biology* 112: 535–542.

Branden, C. & Tooze, J. (1999). *Introduction to Protein Structure*, Garland Publishing, New York, USA.

Breiman, L. (2001). Random forests, *Machine Learning* 45: 5–32.

Chang, B., Ratnaweera, A. & Watson, S. H. H. (2004). Particle swarm optimisation for protein motif discovery, *Genetic Programming and Evolvable Machines* 5: 203–214.

Cheng, J., Greiner, R., Kelly, J., Bell, D. & Liu, W. (2002). Learning bayesian networks from data: An information-theory based approach, *Artificial Intelligence* 137(1-2): 43–90.

Chiba, S., Sugawara, K. & Watanabe, T. (2001). Classification and function estimation of protein by using data compression and genetic algorithms, *Proc. of Congress on Evolutionary Computation*, IEEE Press, Piscataway, USA, pp. 839–844.

Cohen, W. W. (1995). Fast effective rule induction, *In Proceedings of the Twelfth International Conference on Machine Learning*, Morgan Kaufmann, pp. 115–123.

Cooper, G. (2000). *The Cell: A Molecular Approach*, Sinauer Associates, Sunderland, UK.

Davies, M. N., Secker, A., Freitas, A. A., Mendao, M., Timmis, J. & Flower, D. R. (2007). On the hierarchical classification of G protein-coupled receptors, *Bioinformatics* 23(23): 3113–3118.

Drenth, J. (1999). *Principles of Protein X-Ray Crystallography*, Springer-Verlag, New York, USA.

Durbin, R., Eddy, S., Krogh, A. & Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, Cambridge University Press, Cambridge, USA.

Fawcett, T. (2006). An introduction to ROC analysis, *Pattern Recognition Letters* 27: 861–874.

Ferreira, C. (2001). Gene expression programming: a new adaptative algorithm for solving problems, *Complex Systems* 13(2): 87–129.

Frank, E., Hall, M. & Pfahringer, B. (2003). Locally weighted naive bayes, *Proc. Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann.

Frank, E. & Witten, I. H. (1998). Generating accurate rule sets without global optimization, *Proc. 15th International Conference on Machine Learning*, Morgan Kaufmann, pp. 144–151.

Freitas, A. (1998). *Data Mining and Knowledge Discovery with Evolutionary Algorithms*, Heidelberg, Germany.

Freund, Y. & Schapire, R. E. (1999). A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence* 14(5): 771–780.

Friedman, J., Hastie, T. & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting, *Annals of Statistics* 28(2): 337–407.

Mirceva G. & Davcev, D. (2009). HMM-based approach for classifying protein structures, *International Journal of Bio-Science and Bio-Technology* 1(1): 37–46.

Griffiths, A., Miller, J., Suzuki, D., Lewontin, R. & Gelbart, W. (2000). *An Introduction to Genetic Analysis*, 7th edn, W.H. Freeman, New York, USA.

Hardin, C., Pogorelov, T. & Luthey-Schulten, Z. (2002). Ab initio protein structure prediction, *Current Opinion in Structural Biology* 12(2): 176–181.

Hunter, L. (1993). *Artificial Intelligence and Molecular Biology*, 1st edn, AAAI Press, Boston, USA.

Jaroniec, C., MacPhee, C., Bajaj, V., McMahon, M., Dobson, C. & Griffin, R. (2004). High resolution molecular structure of a peptide in an amyloid fibril determined by magic angle spinning NMR spectroscopy, *Proceedings of the National Academy of Sciences of the USA* 101(3): 711–716.

John, G. H. & Langley, P. (1995). Estimating continuous distributions in bayesian classifiers, *Proc. of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann, pp. 338–345.

Kalegari, D. & Lopes, H. (2010). A differential evolution approach for protein structure optimisation using a 2D off-lattice model, *International Journal of Bio-Inspired Computation* 2(3/4): 242–250.

Klabunde, T. & Hessler, G. (2002). Drug design strategies for targeting G-protein-coupled receptors, *Chembiochem* 3(10): 928–944.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection, *Proc. of 14th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, San Francisco, USA, pp. 1137–1143.

Kohavi, R. (1996). Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid, *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining Conference*, pp. 202–207.

Kotsiantis, S. B. (2007). Supervised machine learning: A review of classification techniques, *Informatica* 31(3): 249–268.

Koza, J. (1996). Classifying protein segments as transmembrane domains using genetic programming and architecture-altering operations, *in* P. Angeline & K. Kinnear Jr. (eds), *Advances in Genetic Programming*, Vol. II, MIT Press, Cambridge, USA.

Kyte, J. & Doolittle, R. (1982). A simple method for displaying the hydropathic character of proteins, *Journal of Molecular Biology* 157: 105–132.

Landwehr, N., Hall, M. & Frank, E. (2005). Logistic model tree, *Machine Learning* 59(1): 161–205.

Lee, M., Duan, Y. & Kollman, P. (2001). State of the art in studying protein folding and protein structure prediction using molecular dynamics methods, *Journal of Molecular Graphics and Modelling* 19(1): 146–149.

Lewis, P., Momany, F. & Scheraga, H. (1973). Chain reversals in proteins, *Biochimica et Biophysica Acta* 303(2): 211–229.

Lodish, H., Berk, A., Matsudaira, P., Kaiser, C., Krieger, M., Scott, M., Zipursky, L. & Darnell, J. (2000). *Molecular Cell Biology*, 4th edn, W.H. Freeman, New York, USA.

Lopes, H. (2008). Evolutionary algorithms for the protein folding problem: A review and current trends, *Computational Intelligence in Biomedicine and Bioinformatics*, Vol. I, Springer-Verlag, Heidelberg, pp. 297–315.

Manning, A., Keane, J., Brass, A. & Goble, C. (1997). Clustering techniques in biological sequence analysis, *in* J. Komorowski & J. Zytkow (eds), *Principles of Data Mining and Knowledge Discovery*, Vol. 1263 of *Lecture Notes in Computer Science*, Springer-Verlag, Heidelberg, Germany, pp. 315–322.

Martin, B. (1995). *Instance-based learning: Nearest neighbor with generalisation*, Master's thesis, Department of Computer Science, University of Waikato, Waikato, New Zealand.

Matthews, B. (1975). Comparison of the predicted and observed secondary structure of T4 phage lysozyme, *Biochimica et Biophysica Acta* 405(2): 442–451.

Mitchell, T. M. (1997). *Machine Learning*, McGraw-Hill, New York, USA.

Nelson, D. & Cox, M. (2008). *Lehninger Principles of Biochemistry*, 5^{th} edn, W.H. Freeman, New York, USA.

Nölting, B. (2006). *Protein Folding Kinetics*, 2^{nd} edn, Springer-Verlag, Berlin, Germany.

Ohkawa, T., Namihira, D., Komoda, N., Kidera, A. & Nakamura, H. (1996). Protein structure classification by structural transformation, *Proc. of IEEE International Joint Symposia on Intelligence and Systems*, IEEE Computer Society Press, Piscataway, USA, pp. 23–29.

Pauling, L., Corey, R. & Branson, H. (1951a). Configurations of polypeptide chains with favored orientations of the polypeptide around single bonds: two pleated sheets, *Proceedings of the National Academy of Sciences of the USA* 37(11): 729–740.

Pauling, L., Corey, R. & Branson, H. (1951b). The structure of proteins: two hydrogen-bonded helicel configurations of the polypeptide chain, *Proceedings of the National Academy of Sciences of the USA* 37: 205–211.

Platt, J. C. (1998). Fast training of support vector machines using sequential minimal optimization, *in* C. B. B. Schoelkopf & A. Smola (eds), *Advances in Kernel Methods*, MIT Press, Cambridge, USA.

Quinlan, J. (1993). *C4.5: Programs for Machine Learning*, San Francisco, USA.

Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition, 77(2): 257–286.

Scapin, M. & Lopes, H. (2007). A hybrid genetic algorithm for the protein folding problem using the 2D-HP lattice model., *in* A. Yang, Y. Shan & L. Bui (eds), *Success in Evolutionary Computation*, Vol. 92 of *Studies in Computational Intelligence*, Springer, Heidelberg, Germany, pp. 205–224.

Seymore, K., McCallum, A. & Rosenfeld, R. (1999). Learning hidden Markov model structure for information extraction, *AAAI 99 Workshop on Machine Learning for Information Extraction*, pp. 37–42.

Shmygelska, A. & Hoos, H. (2005). An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem, *BMC Bioinformatics* 6: 30.

Sing, T., Sander, O., Beerenwinke, N. & Lengauer, T. (2005). ROCR: visualizing classifier performance in R, *Bioinformatics* 21: 3940–3941.

Sunde, M. & Blake, C. (1997). The structure of amyloid fibrils by electron microscopy and X-ray diffraction, *Advances in Protein Chemistry* 50: 123–159.

Tavares, L., Lopes, H. & Lima, C. (2008). A comparative study of machine learning methods for detecting promoters in bacterial DNA sequences, *in* D.-S. Huang, D. S. L. Donald C. Wunsch II & K.-H. Jo (eds), *Advanced Intelligent Computing Theories and Applications*, Vol. 5227 of *Lecture Notes in Computer Science*, Springer-Verlag, Heidelberg, pp. 959–966.

The UniProt Consortium (2010). The universal protein resource (UniProt) in 2010, *Nucleic Acids Research* 38: D142–D148.

Tsunoda, D. & Lopes, H. (2006). Automatic motif discovery in an enzyme database using a genetic algorithm-based approach, *Soft Computing* 10: 325–330.

Tsunoda, D., Lopes, H. & Freitas, A. (2011). A genetic programming method for protein motif discovery and protein classification, *Soft Computing* pp. 1–12. DOI 10.1007/s00500-010-0624-9.

Wang, T. L. J. & Ma, Q. (2000). Application of neural networks to biological data mining: A case study in protein sequence classification, *In Proc. of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 305–309.

Webb, G. I. (2000). Multiboosting: a technique for combining boosting and wagging, 40: 159–197.

Webb, G. I., Boughton, J. R. & Wang, Z. (2005). Not so naïve Bayes: aggregating one-dependence estimators, *Machine Learning* .

Weinert, W. & Lopes, H. (2004). Neural networks for protein classification, *Applied Bioinformatics* 3(1): 41–48.

Weinert, W. & Lopes, H. (2006). GEPCLASS: A classification rule discovery tool using gene expression programming, *in* X. Li, O. R. Zaïane & Z.-H. Li (eds), *Advanced Data Mining and Applications*, Vol. 4093 of *Lecture Notes in Computer Science*, Springer-Verlag, Heidelberg, Germany, pp. 871–880.

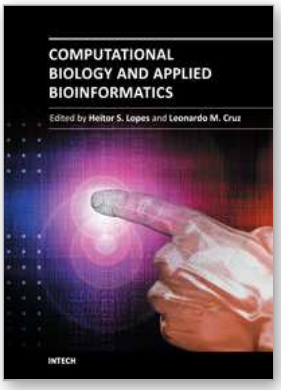
Witten, I., Frank, E. & Hall, M. (2011). *Data Mining: Practical Machine Learning Tools and Techniques*, 3^{rd} edn, Morgan Kauffmann, San Francisco.

Wolstencroft, K., Lord, P., Tabernero, L., Brass, A. & Stevens, R. (2006). Protein classification using ontology classification, *Bioinformatics* 22(14): e530–e538.

Wüthrich, K. (1986). *NMR of Proteins and Nucleic Acids*, John Wiley & Sons, New York, USA.

Yanikoglu, B. & Erman, B. (2002). Minimum energy configurations of the 2-dimensional HP-model of proteins by self-organizing networks, *Journal of Computational Biology* 9(4): 613–620.

Zheng, Z. & Webb, G. I. (2000). Lazy learning of bayesian rules, *Machine Learning* 41: 53–87.



Computational Biology and Applied Bioinformatics

Edited by Prof. Heitor Lopes

ISBN 978-953-307-629-4
Hard cover, 442 pages
Publisher InTech
Published online 02, September, 2011
Published in print edition September, 2011

Nowadays it is difficult to imagine an area of knowledge that can continue developing without the use of computers and informatics. It is not different with biology, that has seen an unpredictable growth in recent decades, with the rise of a new discipline, bioinformatics, bringing together molecular biology, biotechnology and information technology. More recently, the development of high throughput techniques, such as microarray, mass spectrometry and DNA sequencing, has increased the need of computational support to collect, store, retrieve, analyze, and correlate huge data sets of complex information. On the other hand, the growth of the computational power for processing and storage has also increased the necessity for deeper knowledge in the field. The development of bioinformatics has allowed now the emergence of systems biology, the study of the interactions between the components of a biological system, and how these interactions give rise to the function and behavior of a living being. This book presents some theoretical issues, reviews, and a variety of bioinformatics applications. For better understanding, the chapters were grouped in two parts. In Part I, the chapters are more oriented towards literature review and theoretical issues. Part II consists of application-oriented chapters that report case studies in which a specific biological problem is treated with bioinformatics tools.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820

Fax: +385 (51) 686 166
www.intechopen.com

Fax: +86-21-62489821