479128 Aplicacaõ dos Computadores. SBU
Bioinformática
Genoma
Aprendizagem. máquina

Multi-agent Systems and Machine Learning Facilitating Genome Annotation

sistemas multiagentes

Ana L. C. Bazzan [1]
André C. P. L. F. Carvalho [2]

CNPq 1.03.04.00-2

Abstract: Multi-agent systems and Machine Learning techniques are being frequently used in genome projects. One of their uses is in the annotation process (annotation pipeline). Annotation employs a high number of programs and scripts, which can be automated, thus freeing the specialist to carry out more valuable tasks. Artificial Intelligence and Machine Learning techniques have been employed in several problems, such as phylogeny reconstruction, protein annotation, protein structure prediction, gene identification, gene expression analysis, sequences alignment, and so on. This paper discusses three Molecular Biology problems where Artificial Intelligence and Machine Learning techniques have been successfully employed: gene identification, gene expression analysis, and protein annotation.

1 Introduction

In order to investigate the role played by the genes in different organisms, several genome projects were initiated in the last decades. In any genome project, one of the key stages is the annotation process (annotation pipeline), which involves the handling of a large amount of data. Annotation employs a high number of programs and scripts. These can easily be automated, freeing the specialist to carry out more valuable tasks. However, there is still a need for automated and integrated tools to support the annotation pipeline. Until the last decade, annotation was primarily carried out manually, a tedious but extremely valuable resource. However, the idea of automating the annotation is not new. Since it is time-consuming, facilitating some parts of the process has been the motivation of many tools, as discussed in the next section.

The motivation for this work is the application of techniques from *Artificial Intelligence* (AI), in particular *Multi-Agent Systems* (MAS), and *Machine Learning* (ML) to solve problems in Bioinformatics. The use of more sophisticated techniques for these applications

[1] Instituto de Informática, UFRGS
Caixa Postal 15064
91501–970 – Porto Alegre, Brazil
bazzan@inf.ufrgs.br
[2] Instituto de Ciências Matemáticas e de Computação. USP
Caixa Postal 668
13560–970, São Carlos, Brazil
andre@icmc.usp.br

is important due to the large amount of unexplored knowledge in the current mass of biological data. They are particularly useful when dealing with large, not clearly structured, and/or heterogeneous databases, such as those kept by NCBI (http://www.ncbi.nlm.nih.gov/), and EBI (http://www.ebi.ac.uk).

In this paper, the authors briefly describe the use of ML techniques in three Bioinformatics problems: gene identification [18,44,51,55,57], gene expression analysis [10,23,24, 30,52], and partial protein annotation [6,34].

The paper is organized as follows. Section 2 presents the main aspects of Bioinformatics. It also briefly presents the techniques employed in this work: Section 2.2 discusses how ML techniques and MAS can be employed in Bioinformatics; Section 2.3 presents some environments for annotation, and Section 2.4 discusses the use of machine learning techniques for annotation. Three particular problems from Molecular Biology where ML techniques can be used, Gene Recognition, Gene Expression Analysis and Protein Annotation, are discussed in the Sections 3, 4 and 5, respectively. Finally, in Section 6, the main conclusions of this paper are presented.

2 Background

2.1 Bioinformatics

Bioinformatics or Computational Biology is a research area concerned with the investigation of tools and techniques from computing to solve problems from Biology, particularly Molecular Biology [2, 50]. The later studies the organisms' genomes, defined as their set of genetic information. This information is coded on genes along DNA (Deoxyribonucleic Acid) molecules. The DNA molecule is composed of sequences of nucleotides, which can be of four types: Adenine (A), Cytosine (C), Guanine (G) and Thymine (T). A single DNA molecule typically has thousands of genes, which are employed to produce proteins, essential components of living beings. This is accomplished in a process named gene expression.

The gene expression process has two stages: transcription and translation. In the transcription stage, a mRNA (messenger Ribonucleic Acid) is produced from a DNA strand. The mRNA is similar to the DNA. It is also composed of nucleotides, except for a Uracil (U) in the place of Thymine. In the translation stage, the mRNA molecule obtained is used to generate the final protein, which is made of molecules named aminoacids.

2.2 Machine Learning Techniques

ML is a research area that provides several techniques for extracting concepts (knowledge) from a given dataset [41]. These techniques are usually applied for the induction of a hypothesis, also known as classifier or predictor, through a process named training.

Several ML algorithms have been proposed in the literature and the following principle applies to most of them: given a set of known examples, a ML algorithm induces a classifier C, that should be able of predicting the class of any pattern from the same domain where the learning process occurred.

One analysis carried out on Molecular Biology datasets is the identification of genes on DNA sequences. For such, one can search for particular signals associated with gene expression, like Splice junctions. Several works have applied ML algorithms in the recognition of such signals [18,37,38,44,45,51,55,57].

Another important issue is the gene expression analysis, where the expression level of a subset of genes can be associated to the pathology of a tissue.

2.3 Agent-Based Annotation

Technologies of agents in Bioinformatics are useful because the necessary data is distributed among several sources, it is a dynamic area, its content is heterogeneous, and most of the work can be done in a parallel way. Hence, *information agents* can integrate multiple distributed heterogeneous information sources.

There are only a few multi-agent projects in the domain of Bioinformatics. The scientific community on agents and multi-agent systems is now turning its attention to a key issue to Bioinformatics as well: how to ensure the consistency of the integrated data. Next, we briefly review some of these efforts.

InfoSleuth [5] has been used to annotate livestock genetic samples. In [22], a prototype is described whose aim is to automate the annotation of a sequence of a virus. Their work is based on information gathering: search, filtering, integration, analysis, and presentation of the data to the user. It uses the author framework DECAF, a multi-agent system toolkit. The system has four overlapping multi-agent organizations. The first, Basic Sequence Annotation, is charged with integrating remote gene sequence annotations from various sources with the gene sequences at the Local Knowledge Base Management Agent (LKBMA). The second, Query, allows complex queries on the LKBMAs via a web interface. The third, Functional Annotation, is responsible for collecting information needed to make an informed guess as to the function of a gene, specifically using the three-part Gene Ontology.

GeneWeaver [12] is a multi-agent system for managing the task of genome analysis. Since all processes of identifying genes and predicting function of proteins (despite being labor-intensive and requiring expert knowledge) are computer-based tasks, it is possible to automate them. In case of Geneweaver this has been done using a multi-agent approach.

GeneWeaver is formed by a community of agents that interact with each other, each performing some distinct task, in an effort to automate the processes involved in, for example, determining protein function. Agents in the system can be concerned with the management of the primary databases, performing sequence analyses with existing tools, or storing and presenting resulting information.

A third agent-based tool is the MASKS environment [49], whose aim is to improve symbolic learning through knowledge exchange. The motivation of this work is to mimic human interaction in order to reach better solutions. This aims at supporting a recent practice in data mining which is the use of collaborative systems. Inductors are combined in an multi-agent system with autonomy to improve individual models through knowledge sharing.

These are necessary because even if data mining is a powerful technique for knowledge extraction, none of the embedded algorithm is good in all possible domains. Each algorithm contains an explicit or implicit bias that leads it to prefer certain generalizations over others. Therefore different data mining techniques applied to the same dataset hardly generate the same result. In general, combining inductors increases the accuracy by reducing the bias. This integration aims at overcoming limitations of individual techniques through hybridization or fusion of various techniques. MASKS groups different symbolic ML algorithms encapsulated in agents in order to classify data. Its goal is to improve symbolic learning through knowledge exchange.

2.4 Annotation with Machine Learning Techniques

Automated annotation and machine learning are combined in [34]. A machine learning approach to generate rules based on already annotated keywords of the SWISS-PROT database is described. Such rules can then be applied to yet un-annotated protein sequences. Details can be found in [34]. We describe here just some issues relevant to the approach discussed in Section 5.1. In short, the authors have developed a method to automate the process of annotation regarding keywords in SWISS-PROT, based on an ML algorithm called C4.5 [46].

This algorithm works on training data (in this case, previously annotated keywords regarding proteins) and generate rules to classify new instances. The training data comprises mainly taxonomy entries, InterPro classification, Pfam, and PROSITE patterns. Given these data (called attributes), C4.5 derives a classification rule for a target class (in this case, the keyword). Since dealing with the whole data in SWISS-PROT at once would be prohibitive, the authors divided it in protein groups according to the InterPro classification. Rules were generated and a confidence factor for each was calculated based on the number of false and true positives, by performing a cross-validation, and by testing the rate of error in predicting keyword annotation over the TrEMBL database. The resulting framework (called Spearmint) can be accessed on the web (http://www.ebi.ac.uk/spearmint).

3 Gene Recognition

One of the main applications of ML techniques in computational biology is the recognition of genes in DNA sequences. Genes are used as blueprints to produce proteins during the gene expression process, and they play important roles in the definition of the structure, function and regulatory mechanisms of cells. According to [18], the recognition of genes in DNA sequences can be performed by three different approaches:

- Search by content: Techniques following this approach look for segments of DNA that have the properties, usually statistical properties, of coding regions.
- Search by signal: look for signs indirectly associated with the presence of a gene. Examples of such signals are promoters and splice junctions.
- Search by similarity: try to identify a gene by searching databases of known genes. This approach is usually employed to confirm the results of the two previous approaches.

The search by content approach encompasses those techniques that are employed to recognize genes directly. For such, they try to identify segments of DNA sequences that have properties of coding regions. They are usually based on the knowledge of statistical properties of coding and non-coding regions. For such, several properties can be explored. According to [18], among such properties, one can mention: some amino acids are more frequently used in coding regions than others; different amino acids are encoded by different numbers of codons; codons that map to a given amino acid are not equally used in the majority of the organisms; a gene cannot have stop codons and some codons have a higher probability of being neighbors.

In the search by signal approach, genes are indirectly localized by looking for particular signals associated with their presence in a DNA sequence. These signs usually are particular regions of the DNA that have a specific function in the gene expression process, for example, connecting to an enzyme. Different signs can be looked for, such as: transcription initiation sites (promoters); transcription termination sites (terminators); Splice-junction sites; translation initiation sites (initiation codons) and translation termination sites (stop codons).

The recognition of these signals provides an important information to understand the regulatory mechanisms involved in the gene expression process, since several signals have a regulatory function. Some signals indicate, for example, aspects like the velocity of the gene expression process and the conditions required for its occurrence [53].

However, the detection of a signal is a problem by itself and different signs have different levels of difficulty to be identified. For example, while stop codons are easily identified, other signs, like promoters, are more difficult to be identified.

When ML algorithms are employed to perform search by signal in a DNA sequence, they usually run through a window of a fixed size over the sequence to verify if it contains the signal of interest, i.e., if an identifiable feature (signal) occupies a particular position in the window. ML techniques have been successfully employed to classify the sub-sequence covered by a particular window.

For such, ML-based classifiers are trained using a dataset composed by several positive and negative samples. Positive samples are sequences with the signal of interest in the middle. Negative samples are sequences without the signal of interest in the middle. Once a classifier has been trained, it can be utilized to find a particular signal in a DNA sequence by moving the window over the whole sequence.

Finally, the search by similarity approach looks for an indication of the function (and existence) of a gene by searching for similar genes from other organisms. Usually, this approach is not able to identify all the genes, since its performance is harmed by errors in the sequencing. Frequently, results achieved by this approach are utilized to reinforce predictions made by techniques of search by signal and/or search by content.

Next, two frequently investigated applications for gene recognition employing the search by signal approach are presented.

3.1 Applications

The applications reported in this section involve the recognition of two different signals which might indicate the presence of a gene in a particular DNA sequence: promoter recognition and splice site recognition.

3.1.1 Promoter Recognition The transcription of a DNA molecule starts with the binding of the RNA-polymerase to a specific site of a DNA sequence. This site is named promoter. The promoter is a DNA sequence that differs from sequences that are transcripted or translated, since its function is to be recognized by proteins. Thus, it must be long enough in order to be solely recognized. The appropriate size varies with the size of the genome. For example, in a bacterial genome, twelve base pairs constitute an appropriate promoter signal. The promoter recognition problem is, therefore, related to the identification of a given fixed size sequence contains a promotor (Y) or not (N). To accomplish this task, ML-based classifiers may be inducted.

Actually, the first application of ML in Bioinformatics was the recognition of promoters in the DNA sequence of the bacteria E. Coli [54] in 1982. The authors employed a Perceptron Neural Network [32] using the window-based approach.

Figure 1 illustrates the search for a promoter in a DNA sequence using this approach.

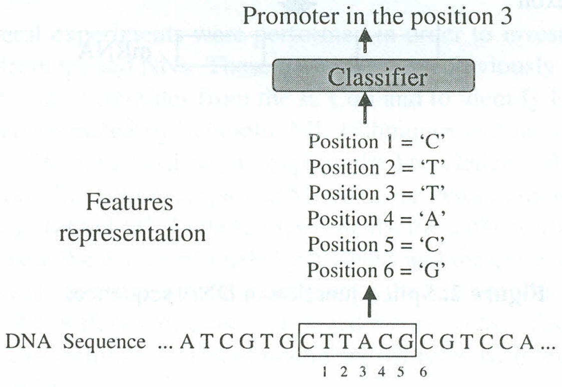

Figure 1. Search for a promoter in a DNA sequence [18]

3.1.2 Splice Site Recognition In eukaryote organisms (those in which most of the genetic material is inside the cells nucleus), the gene expression process includes an additional step. Eukaryote's genes consist of alternated segments of exons and introns. Exons are the segments of a gene that encode a protein (coding regions) and introns are those segments that do not encode a protein (non-coding regions). The intron segments are spliced-out before the translation step. These segments used to be called "superfluous" DNA [47]. Now it is known that they play an important role in the gene expression process. Splice junctions are the boundary points where splicing occurs. After splicing, only the exons are maintained. The intron/exon and exon/intron boundaries are named in the literature acceptors and donors sites, respectively. Figure 2 illustrates splice-junction in DNA sequences.

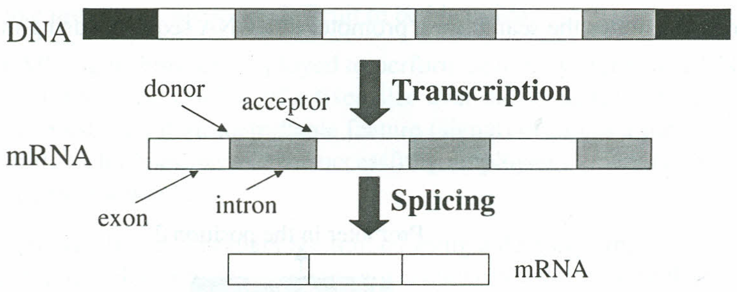

Figure 2. Splice-junction in DNA sequences

Different techniques have been successfully employed by several researchers for the recognition of splice junctions in DNA sequences [14]. For example, [35] reports the use of Decision Trees [11], K-Nearest Neighbors [25] and Artificial Neural Networks (NNs) [32]. In [38], Boosting [27] and Support Vector Machines (SVMs) [19] are employed. A hybrid approach combining Rules and Artificial Neural Networks is presented in [47].

A common problem in Bioinformatics datasets is the presence of noise, which may influence the performance achieved by ML techniques. Due to the imprecise nature of biological experiments, redundant and noisy samples can be present at a high rate. Noisy patterns can reduce the efficiency of the generated classifier and should be removed.

In [39], Lorena and de Carvalho present an empirical evaluation of the pre-processing obtained by five dataset reduction techniques in classification of splice-site. The experimental results obtained indicated that the elimination of potentially noise patterns could improve the overall performance. The same pre-processing techniques were also able to eliminate redundant and similar samples without harming the concept induction or even improving it.

Sub-symbolic ML techniques, like NNs and SVMs, have achieved very good performance when applied to several Bioinformatics problems, for example, gene recognition problems. However, sub-symbolic techniques usually represent their acquired knowledge in a numerical form, by the value of their weight connections and bias. There are several applications where the ability to explain the decisions taken by the ML technique is even more important than its performance.

Due to this limitation, NNs are commonly referenced as "black boxes". This lack of comprehensibility of the NNs decision making process is one of the main barriers to their widespread use to solve practical problems. In Bioinformatics, for example, the extraction of meaningful knowledge from trained NNs could increase the confidence of biologists and physicists on the decision provided by the NNs, shade new lights in the characteristics of the problem and help them to discover valuable knowledge in the data set.

In [9], several experiments were performed in order to investigate the extraction of meaningful rules from trained NNs. These networks were previously trained to identify promoters in sequences of nucleotides from the E. Coli and to identify human splice junctions. The rules were first extracted by symbolic ML techniques and later optimized by Genetic Algorithms (GAs). The final goal was to explain the knowledge embedded in trained NNs. In these experiments, the authors employed Multi-Layer Perceptron networks trained by the backpropagation algorithm [32]. In these experiments, the authors investigated the infidelity rate (how similar were the decisions made by the NNs and the extracted rules) and the comprehensibility (number and size of rules) comparing the rules produced by three techniques (CN2, C4.5 and C4.5 Rules) and those optimized by the GAs. The results suggested the potential of the use of optimized rules to obtain sets of rules with lower infidelity rates and better comprehensibility.

At another set of experiments performed by the research group of one of the authors, Multi-Layer Perceptron networks were optimized by GAs for promoter recognition in DNA sequences. The results obtained were better than those obtained by these techniques optimized by trial and error.

3.2 Discussion

There are also other works where search by signal using different signals and search by content have been successfully employed in the recognition of genes in DNA sequences [18, 44, 51, 55, 57]. However, due to the limitation of space, these approaches will not be addressed here.

A current trend is to use different sources of evidence [1] and combine techniques following different approaches for gene recognition, usually search by signal and search by content. Thus, genes identified by signal can have their identity confirmed by search for its similarity with a previously known gene.

In the translation process, the transcript can be spliced in different sections of the pre-mRNA sequence. This process is known as alternative splicing. There are many cases where the primary RNA transcript can be alternatively spliced in different sections of the sequence, thus generating alternative splice pathways. There has been a growing interest in the recognition of alternative splice sites [31].

,Finally, another key issue in gene recognition is the identification of pseudogenes [17]. These are small parts of a DNA sequence that still remain in the sequence, although they are no longer functional. They are more frequently found in eukaryotic genomes.

4 Gene Expression Analysis

One of most potentially benefited areas of the post-genomic age is medicine. The new technologies being developed may allow a more accurate insight about the origin and evolution of several diseases, which would lead to the development of more specific treatments, as well as to the design of new drugs. In special, the combat to cancer has already made promising advances, mainly due to the introduction of new wide scale gene expression analysis technologies, such as microarrays [40, 42].

Microarrays are hybridization-based methods that allow monitoring expression levels of thousands of genes simultaneously. The role of the genes of a cell in a given moment and under particular circumstances can be better understood by assessing their expression levels. The comparison between gene expressions patterns in normal and disease cells can provide important indications on the development of determined pathological states, as well as information that can lead to earlier diagnosis and possibly more efficient treatment.

Gene expression analysis works with the expression level of the genes in tissues measured under several conditions, like: normal and diseased (for example, cancer) tissue; a tissue under different drugs or treatments; and a tissue with different pathologies or diseases.

The main goal of such an analysis is to know which genes might be related to different tissues conditions by presenting a significantly different expression levels for them. However, the very large number of possibilities makes the discovery of such genes a very difficult task. Thus, the monitoring of the largest possible number of genes is necessary. By allowing the monitoring of the expression level of thousand of genes simultaneously, microarrays have become valuable medical tools.

Microarrays are not the only method employed to monitor the expression of a large amount of genes simultaneously. The large-scale gene expression analysis methods can be summarized in two main groups: tag counting methods (e.g. SAGE, MPSS) and hybridization-based methods (e.g. cDNA, oligonucleotide microarray). For ML techniques, there is not much difference if the gene expression dataset was generated by tag counting methods or hybridization-based methods. In spite of the method employed to acquire the gene expression data, their analysis involves several aspects that have been addressed in the literature [10,23].

When ML techniques are employed for the analysis of gene expression data, there is usually a disproportionate rate between the very high number of attributes (genes) and the low number of training samples (tissues), which can make it difficult to analyze the obtained results and reduce the classification accuracy. Gene expression profiles usually present the expression level of thousands of genes. Besides, and more important, large changes in a particular phenotype can be due to changes in the expression of a small subset of its genes. Thus, it is important to select subsets of relevant genes to work with. Gene selection gives biologists a small set of genes to carry out more specific, complex and more expensive investigations.

In order to overcome the problems related to the high number of attributes present in gene expression data, the next section discusses two strategies frequently employed in conjunction with ML classifiers for gene expression analysis.

4.1 Applications

Two approaches currently employed for gene selection are: gene ranking and feature selection. The first approach ranks the genes in order of importance for a given task. The second selects the best subset of genes in order to accomplish a given goal. These approaches are described next.

4.1.1 Ranking a subset of genes The main goal of gene ranking methods is to order the genes according to their ability to discriminate tissues related to a condition A from tissues related to a condition B. It looks for genes highly expressed in tissues from a class A and lowly expressed in tumor tissues from a class B, and vice-versa. Biologists want to know which are these genes in order to analyze them.

The metrics or scores for gene ranking can be divided into parametric and nonparametric scores. The parametric scores make assumptions about the form of the distribution of the scores within each group, while the nonparametric scores do not make such assumptions, thus being more robust [8].

The nonparametric measurements generally specify hypothesis in terms of population distributions rather than parameters such as means and standard deviations. These metrics are almost as capable of detecting differences among populations as the parametric ones when normality and other assumptions need to be satisfied. Nonparametric scores may be, and often are, more powerful in detecting population differences when these assumptions are not satisfied.

These metrics are evaluated according to their ability in selecting predictive genes.

This evaluation is made by constructing classifiers using the genes selected by each metric and then comparing the performance of these classifiers.

The gene raking approach is particularly interesting for health science researchers, since they are interested in discovery which genes are related to a given pathology. For such, they need to know which genes should be analyzed in their laboratories.

In [26], different metrics for gene ranking were evaluated, based on their ability to rank the most relevant genes whose expression level could help distinguish between tumor and normal tissues. The evaluation was carried out by training classifiers using the *N* genes ranked by each metric and then comparing the performance of these classifiers.

The performance of the classifiers was evaluated using the error rate in the classification of new tissues. The classifiers are generated using different machine learning algorithms, Support Vector Machines (SVMs) and C4.5. The experiments were conducted employing cancer-related datasets obtained from NCBI web site. These datasets had the expression level of a set of genes obtained using the SAGE technique.

The tests were performed using three different datasets: with the 4, the 10, and the 100 most relevant genes measured by the metrics employed. It was possible to classify the tissues using the expression level of only 4 genes. It was also observed that the rank of genes provided by the different metrics resulted in high differences in the tumor classification rates obtained by the classifiers employed, which shows the importance of further analysis in laboratory.

4.1.2 Selection of a subset of genes Another common gene expression analysis is the classification of a tissue given the expression level of its genes. The most relevant genes are not necessarily the best subset of genes for tissue classification. Thus, the main goal here is not the knowledge of the order of the genes to be analyzed in wet laboratories, but which subset of genes forms the best feature vector for the classification of a tissue in a give class (or classes).

As previously mentioned, gene expression datasets usually have the expression level of thousands of genes (attributes). Depending on the issue under investigation, this large number of attributes makes the analysis impractical. Several ML techniques have problems to work with patterns with very large number of attributes. Thus, they can benefit from a reduction in the number of attributes. Besides, the use of a compact subset of genes can bring other benefits:

- There would be a higher chance to identify important genes for classification, in a biological way;
- the effects of noise in gene expression analysis experiments analysis would be mini-

mized;

- accuracy of classifiers would be improved, which is important for diagnosis purposes;
- By focusing only in subset of genes, technologies of gene expression analysis could become more accessible (economically), since they could be manufactured with fewer genes associated and thus become a common clinical tool.

Thus, it is important to select subsets of relevant genes to work with. Feature selection techniques are frequently employed to find the best subset of genes. The subset of genes produced by a feature selection techniques is evaluated by constructing classifiers using the subset of selected genes.

Other ML approaches can also be used for dimensionality reduction. In [20], Genetic Algorithm (GA) [29] was employed to perform feature selection. GAs have already been employed for the gene selection problem. Most of the approaches so far investigated can be regarded as wrappers [36], since they explicitly depend on classification methods to select genes. The main disadvantage of this approach is its high computational cost, since the classifier embedded into the GA's fitness function has to be executed a large number of times. Another problem is that they are dependent of specific learning algorithms.

In order to overcome these difficulties, the authors proposed a simple and low cost method for gene selection. The proposed method follows the filter approach [36]. It is called the GA-Filter method. It utilizes only the intrinsic characteristics of the training data to select a subset of genes. As a result, it is not limited to any classifier and its computation is very efficient. The idea is to use GAs to search for genes that better represent the domain. Thus, the classification complexity of the problem can the minimized and, at the same time, genes relevant to the studied pathology can be revealed.

Experiments were performed using two different datasets obtained by microarrays: The small round blue cell tumors of childhood (SRBCT) dataset, first analyzed in [33]. The proposed technique was able to select a smaller number of genes with comparable accuracy rates.

In another work [21], the authors presented a new approach to perform gene selection using SVMs and GAs for multi-class problems. A similar combination had already been used for the same purpose by Frohlich et al [28]. However, in this previous work, the combination was employed just as a single example of the authors' system general framework. In [21], specific solutions were proposed, such as the optimization of the SVM hyperparameters *C* independently for all classifiers, the normalization of the samples to unit length, allowing the use of a correlation-based similarity measure together with the SVM and the use of the Generalized Approximate Cross-Validation [56] as a generalization estimator for SVMs.

The experiments were performed with the SRBCT dataset, first analyzed in [33]. The experiments had two main goals. First, the comparison of the performance of the classifiers trained with the set of genes selected by the proposed method and trained with the whole set of genes. The second goal was the comparison of the results obtained with multi GA-SVM method against those obtained by other results found in the literature, regarding both accuracy rate and dimensionality reduction. In the experiments, small subsets of genes were selected and the results were similar to those obtained by other gene selection approaches investigated.

4.2 Discussion

Methods like microarrays enable the measurement of the levels of thousands of mRNA molecules inside a cell and, consequently, the proteins being produced. Therefore, the role of the genes of a cell in a given moment and under some circumstances can be better understood by assessing their expression levels. In this context, the comparison between gene expression patterns through the measurement of the levels of mRNA in normal and disease cells can supply important indications on the development of determined pathological states, as well as information that can lead to earlier diagnosis and more efficient treatment.

Most of the works where ML techniques have been employed for gene expression analysis are related to either classification or clustering problems [8, 13, 23, 40, 52]. Classification problems include the classification of unknown tissues as normal tissue or tumor tissue. Clustering problems include the discovery of new subgroups of tumor tissues.

Amongst the most trivial applications of microarrays, the classification of new tissue samples is an essential step for the assessment of severe diseases. However, when dealing with gene expression data, there is usually a disproportionate rate between the high number of features (genes) and the low number of training samples (tissues).

The dimensionality problem is expected to become more serious when dealing with multi-class domains since these are intrinsically harder than the binary ones because the classification algorithm has to learn to construct a greater number of separation boundaries or relations [48].

5 Protein Annotation

5.1 An Agent-Based Environment for Automating Annotation

5.1.1 Overview In spite of the growing number of tools for analyzing DNA sequences, there is no easy solution to the annotation problem. The shortcomings are twofold. First, while no single program can outperform the human expert, when several different programs are used, the performance can increase. Second, each program has its own output format, thus posing difficulties when one wants to compare results. The aim of the Agent-based environmentT for aUtomatiC annotation of Genomes (ATUCG) [7] is to provide the team in charge of genome sequencing and annotation with a tool that facilitates the activities involved in carrying out those tasks.

Since ATUCG is an agent-based environment, agents are allocated to do most of the automated methods. Besides, agents are in charge of performing much of the tedious work required when using different programs and tools, namely the translation or formating of inputs and outputs. ATUCG consists of three layers, each having several agents. The interrelationships and general overview are depicted in Figure 3.

Layer I aims at automating the tasks behind the process of finding *open reading frames* (ORFs). Having received a file containing the DNA sequencing of a given organism, layer I helps the user to define the ORFs. To this means, specific programs are used, each regarding the expertise of a particular agent. The output that layer I passes to layer II is a file containing a list of non-redundant ORFs.

Layer II is associated with three main tasks: extraction and formatting of data, automatic annotation of data regarding profiles or families of proteins, and generation and validation of rules to automatically annotate the Keywords field in the SWISS-PROT database (http://www.expasy.ch/sprot/).

Layer III permits the user to validate the automatic annotation by helping him/her to perform the verification of the proposed annotation, mainly by "translating" the annotation rules into a semantically richer language and presenting it in a more understandable form. This is done by linking attributes presented in the rules to more complete data extracted from databases such as PROSITE (http://www.expasy.ch/prosite) and InterPro (http://www.ebi.ac.uk/interpro/).
Once the annotations are verified, they are collected into two files: one for accepted annotations, another for rejected ones.

In the present text we concentrate on layer II (next section), focusing on ML algorithms. The reader is referred to [7] for more details on the other layers.

5.1.2 Using Machine Learning in Layer II The aim of layer II is threefold:

1. design of agents which are experts on data extraction (e.g. from SWISS-PROT and InterPro databases) and formatting;

2. creation of a database of rules for annotation (this involves training and validation stages using standard ML techniques);

3. automatization of the annotation of fields related to motifs or profiles regarding families

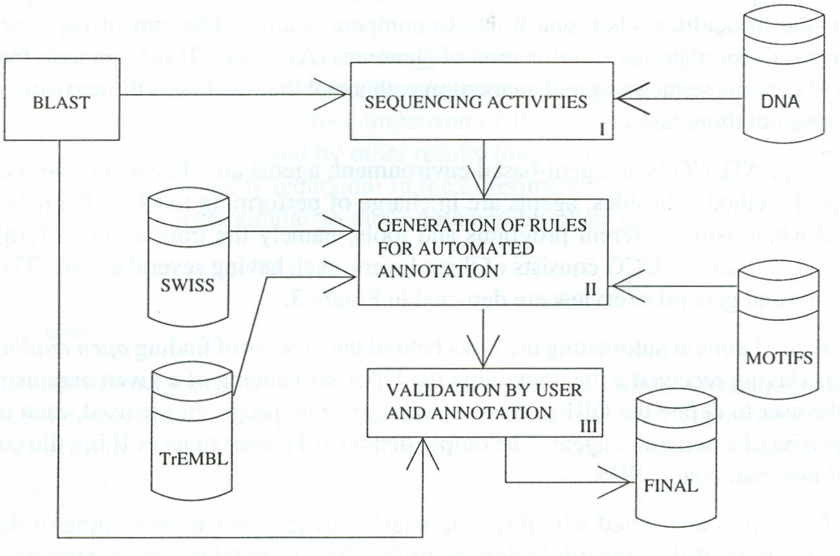

Figure 3. Overview of the three–layer architecture

of proteins (i.e. annotation regarding PROSITE and InterPro attributes).

The input to this layer is a file with a nonredundant list of ORFs. Besides, layer II has connections to several databases for in this layer training data has to be gathered in order to produce rules for automatic annotation. The main output passed to layer III is a set of proposed annotation for selected attributes. Figure 4 depicts the various databases and agents involved in the tasks associated with the layer II. As already mentioned, three main tasks are carried out in this layer: data extraction and reformatting; application of different ML techniques to train data and produce the rules for annotation; and the annotation regarding protein profiles, keywords, and other attributes.

The process called "data extraction and formating" deals with accessing two main databases, namely SWISS-PROT and TrEMBL, aiming to retrieve training and validation data respectively. The agent called "extraction agent" is invoked and has the following tasks: ask the user for attributes which will be used in the training process, retrieving data, and preparing the retrieved data in order to serve as input for our ML tools.

The attributes which the user typically wants to work with are proteins (coded according to the SWISS-PROT accession number) and attributes such as classification of proteins in domains or families. These classifications are stored in the databases of protein families: PROSITE, InterPro, PFAM, and eventually others. This procedure retrieves data regarding

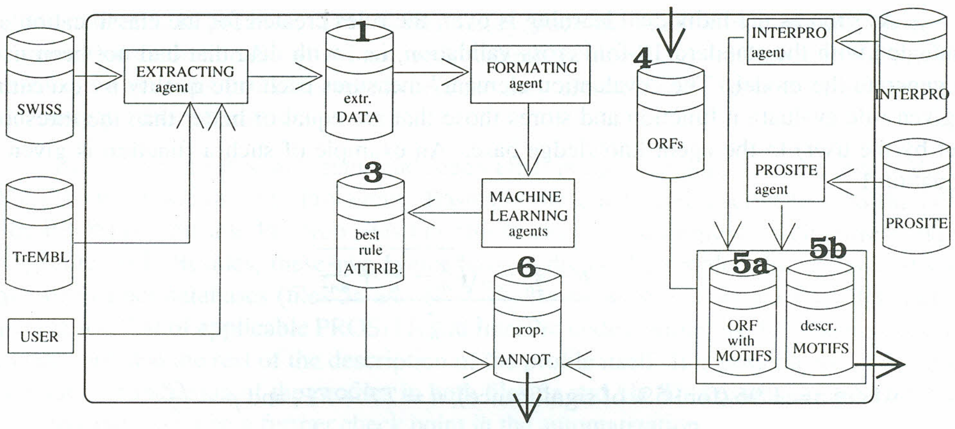

Figure 4. Layer II

accession number of proteins, keywords, and classification regarding the protein families.

Then the preprocessed data is sent to the various agents responsible for each ML algorithm. Steps 1 and 2 (asking the user and retrieving the data) are relatively simple. However, step 3 is not only more complex, but can also be performed in several alternative ways since each ML algorithm makes use of proper syntax to describe a hypothesis. Since the agents will eventually need to exchange their best rules for classification, it is necessary to have these transformed to the same format. The details of the training using ML algorithms, and the design and use of cooperative learning can be found in Bazzan *et al.* [6] and Schroeder and Bazzan [49] respectively.

The rules which will determine whether or not a given protein can be annotated with a given keyword are generated using the ML algorithms. In all cases the algorithms follow approximately the same approach. First, a training set is given to the algorithm. Depending on this algorithm, the output varies but, as mentioned before, there is an agent which is in charge of translating all outputs to a standard format. Then, agents receive and process the data, and generate their annotation rules. Regarding the different learning elements, symbolic classification techniques similar to C4.5 are used: CN2 [15], T2 [43], and Ripper [16].

Learning happens in two stages. The first one is dedicated to individual learning. The input is the pre-processed training set and the configuration set. After that, the agent applies its rules of classification to the examples. The objective of the individual learning is to create an individual model of the domain problem. This model is a set of rules that were approved by the "evaluation element".

As soon as the individual learning is over, the rules created for the classification are evaluated with the standard 10-fold cross-validation, i.e. with data that had not been used to generate the model. The "evaluation element" measures each rule quality by executing a given rule evaluation function and stores those that are equal or better than the threshold (set by the user) to the agent knowledge base. An example of such a function is given in Equation 1.

$$c = \frac{p + \frac{z^2}{2n} - z\sqrt{\frac{p}{n} - \frac{p^2}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}} \tag{1}$$

where $z = 1.96$ (for 95% of significance), $n = TP + FP$, and $p = \frac{TP}{n}$.

Most of the time the individual learning stage produces a rule set with well described target classes along with poor described ones. This happens due to the algorithm heuristics applied to the data to extract knowledge.

In the next stage, cooperative learning, the goal is to improve the individual results. The input for the cooperation stage is the knowledge base that reflect the knowledge obtained during the individual learning. The cooperative learning consists of two further steps. During the first one, the agent queries other agents' knowledge bases. The first agent to start the interaction is the one with the poorest overall accuracy. The agent searches for its equivalent rules with better quality. The rules that fill this requisite are added to the agent "model merge". Each agent repeats this process from the poorer to the richer average accuracy.

We say that a rule is equivalent to another when the two describe the same class and the attributes overlap. This way a high quality rule is added to the learner ModelMerge when it is similar, overlaps, subsumes, or is in conflict with a low quality rule produced by the learner. For example, consider the rules $R1$ and $R2$ that describe class **C**. Rule $R1$ contains the attribute-value test for attributes **x** and **y**, while rule $R2$ includes tests with attributes **x** and **z**. These two rules are related. When the communication ends, the individual knowledge that was not changed is copied into the "modelMerge" component. At this moment each agent possesses two distinct models of the domain problem.

When the cooperative learning ends, the new model stored in the agent ModelMerge component is evaluated by using the test file. The individual model resultant is the one that covered the highest number of instances from the test file. The output generated by the environment is the best agent model. At the end of this process one has a list of the best rules to describe how to annotate each attribute (file 3 in Figure 4).

After data extraction, reformatting, and application of ML techniques to train data and produce the rules for annotation, the last activity in layer II is the annotation related to profiles of motifs in proteins. This is mandatory since the agent which actually performs the annotation automatically would not work without the necessary annotation of the profiles regarding the proteins just discovered (because they are unknown, there is no information concerning profiles of motifs).

To accomplish this, an agent reads each ORF in file 4 (Figure 4) and query both the PROSITE agent and the InterPro agent. Basically these then call a script to send the ORF to the PROSITE and InterPro web sites in order to get a list of profiles which correspond to the specific ORF. Besides, these two agents have to divide the profile information and save it to two distinct databases (files 5a and 5b in Figure 4). File 5a contains the ORF and its classification (list of applicable PROSITE and InterPro codes) while file 5b contains not only the codes but also the rest of the description of the profile itself. It is possible to ask the user to validate the inclusion of the profiles in both files 5a and 5b. This extension in the system is planned and would be a further check point in the automatization.

Finally, the last and central activity in layer II is the annotation of the "Keywords" field. Having partially annotated the record for each protein (for instance, each has already information on profiles and taxonomy), the next step is to apply the rules generated by the ML algorithms. This is the task of the "keyword agent". It reads files 3 and 5a (Figure 4) and do the following:

1. reads an entry from file 5a (ORF together with applicable profiles) ;
2. repeats for each rule in file 3 (remember that each rule corresponds to a different keyword):
   (a) parse the rule;
   (b) tests the rule: tests all preconditions and concludes whether or not the keyword should be annotated;
   (c) do the annotation: creates an entry in file 6 for the ORF, copy the attributes such as profiles, and add the keyword (in case a keyword does not apply, insert nothing);
3. close files.

At the end of this procedure, two files are sent to the next layer: files 5b and 6. In layer III, the user can verify the annotation.

5.2 An Example: Annotating Keywords for Proteins Related to the Organism *A.thalia*

5.2.1 Data Processing using ATUCGRFs from the model organism *Arabidopsis thaliana*
are used, which are available in public databases like SWISS-PROT to feed the layer II (remember that layer I is useful only when dealing with new, unknown sequences which is not

```
@relation Cell_division

@attribute IPR000642 {TRUE, FALSE}
@attribute IPR003593 {TRUE, FALSE}
@attribute IPR003959 {TRUE, FALSE}
@attribute IPR003960 {TRUE, FALSE}
@attribute IPR000819 {TRUE, FALSE}
@attribute IPR001687 {TRUE, FALSE}
...
@attribute IPR001162 {TRUE, FALSE}
@attribute IPR004791 {TRUE, FALSE}
@attribute Cell_division {yes, no}

@data

TRUE,TRUE,TRUE,TRUE, ... ,FALSE,FALSE, yes
...
```

Figure 5. Input file to C4.5 (in this case, the file refers to the Keyword "Cell division")

the case here as public data is used). The data used comes from a local version of the SWISS-PROT database collected in the second semester of 2003. After the query, 2371 proteins were found which relates to *A. thaliana*. Many keywords appeared in the data but the focus is on those whose number of instances is higher than 100 (24 keywords in the present study).

The attributes which appeared in this data are only accession numbers for all attributes related to domains classifications such as InterPro, PROSITE, etc. which appear in SWISS-PROT as cross-referenced databases. For clarity, the use of the framework with a single ML agent is discussed first. The "C4.5 agent" is based on the standard C4.5 algorithm.

Figure 5 shows one example of a input file (there is a similar one for each keyword) for the C4.5 algorithm. Basically, the first line of these files indicates the target class or relation (keyword). The following lines indicate how each attribute is mapped for each protein. The last of these is the target attribute (keyword). Finally, there comes one line for each protein, all formed by as many entries (separated by comma) as there are attributes.

The following procedure of evaluation was performed: once the data is extracted (file 1 in Figure 4), the best rule for each keyword (file 3 in Figure 4) is found and applied to that data. For instance, if a protein is annotated with keyword **K**, then the rule for **K** is checked. This procedure is repeated for each protein and its keywords. Then the total number of times that the proteins is correctly annotated using our rules is counted.

```
Rule 1:
     IPR008271 = y
     -> class Transferase  [97.0%]
Rule 2:
      IPR008271 = n
      -> class n  [91.4%]
Default class: n
Evaluation on training data (2371 items):
Rule  Size  Error  Used  Wrong          Advantage
----  ----  -----  ----  -----          ---------

 1     1   3.0%    46    0 (0.0%)    46 (46|0)  Transferase
 2     1   8.6%  2325  191 (8.2%)     0 (0|0)    n
Tested 2371, errors 191 (8.1%)  <<
        (a)  (b)     <-classified as
        ----  ----
         46  191    (a): class Transferase
            2134    (b): class n
```

Figure 6. Rules generated by C4.5 (keyword "Transferase")

5.3 Results

Next, some preliminary results achieved by the cooperative learning are discussed. To test the validity of the cooperative learning, two symbolic classificators (CN2 and C4.5) were applied on the Keywords field. Here, just two classificators are used in order to give a didactic explanation of how the tool works.

Initially, an individual model was constructed for each classifier. The CN2 and C4.5 rules were both evaluated by the same evaluation function. The accuracy is calculated based on *true positives* (instance exist in data and is predicted) and *false positives* (instance does not exist in data but is predicted). There is a communication between the "CN2 agent" and the "C4.5 agent". The agent with the poorer accuracy starts by asking the other whether it has rules for a given class **K** as well as its quality. If the agent gets an affirmative answer with superior quality in it, then it is necessary to check the equivalence. If the quality and equivalence tests are satisfied, the agent adds the new rule to its "model merge" component.

A rule is depicted in Figure 6. This rule says that if the protein has the InterPro classification IPR008271, then it shall have the keyword "Transferase". Similar boolean tests go on throughout the rule. The last test finally prescribes whether or not a given protein should have the keyword: if all tests fail, then the protein shall not be annotated with the keyword. Once the rules were generated, the evaluation of the quality of these rules was

Table 1. Training Evaluation: Error Rates for Positive (false positives), Negative (false negatives) Classes, and True Positives

Keyword	FN	Rate (%)	FP	Rate (%)	TP
ATP–binding	104	4.6	5	17.2	116
Calcium	39	1.7	1	1.4	72
Chloroplast	340	14.3	0	0	0
Coiled_coil	69	2.9	0	0	31
DNA–binding	85	3.8	0	0	123
Glycoprotein	109	4.8	8	11.0	97
Heme	19	0.9	0	0	144
Hydrolase	169	7.3	1	2.2	45
Iron	58	2.5	0	0	73
Metal–binding	87	3.8	0	0	78
Mitochondrion	119	5	0	0	0
Multigene family	639	32.3	24	2.9	338
Nuclear_protein	214	9.5	0	0	123
Oxidoreductase	166	7.5	0	0	144
Phosphorylation	43	1.9	17	37.0	75
Protein_transport	81	3.5	0	0	31
Repeat	184	8.0	0	0	66
Ribosomal_protein	160	6.7	0	0	0
Signal	192	8.6	2	2.7	141
Transcription_regulation	80	3.5	0	0	98
Transferase	191	8.2	0	0	46
Transit_peptide	329	13.9	0	0	0
Transmembrane	283	12.9	8	4.4	174
Transport	144	6.2	0	0	66

done. The standard evaluation of the training data based on the number of true positives (TP) and false positives (FP) was used.

5.4 Discussion

Besides the confidence computed by Equation 1, the efficency of the training has to be analyzed regarding the two classes: the negative (training instances which generate a rule indicating when a keyword should *not* be annotated); and the positive one (instances which indicate when a keyword should be annotated).

Table 1 shows the results regarding errors for each class (i.e each keyword, which appears in the first column). The second column shows the raw number of false negatives (FN) and the third one this number expressed as percentage (remember that the number of positives and negatives is different for each class). The raw number of false positives (FP) and the percentage are expressed in columns four and five respectively. The last column gives the number of true positives (TP).

If we look at the third and fourth columns, the results seem quite good: most of the rules for annotating keywords produce no false positives at all. The error rate is acceptable for almost all keywords with the exception of "ATP_binding", "Phosphorylation", and "Multigene family". Rates regarding the negative class (i.e. the false negatives) are also low (except for "Multigene family").

In the case of annotation, the error rate regarding false positives is more important than the one regarding false negatives. This is so because if nothing is annotated, this is not a big issue, i.e. the protein will remain unannotated. However, failure regarding false positives is more serious since the annotating would be wrong. Thus, more emphasis to false positives should be given.

When all rules are generated, it is possible to apply them for the annotation of unknown proteins. One has first to annotate the attributes (e.g. classify each protein regarding InterPro, PROSITE, etc. families). This annotation is an important step to enrich the data submitted to the public databases and also for internal use in genome projects.

6 Conclusions

This paper has highlighted three applications of ML and agents techniques to Bioinformatics. In particular, it was discussed how ML techniques can be employed for gene recognition looking for particular signals in DNA sequences, and for gene expression analysis, where the expression level of a subset of genes has been used for tissues classification. Also, an application related to the organism *A. thaliana* was discussed, regarding annotation of proteins keywords. We showed that the efficiency of the annotation concerning the number of false positives is acceptable.

Regarding the ATUCG environment, in the future, we intend to address other fields to be automatically annotated as well. Also, we have been targeting the imbalance in the classes [3, 4].

Concerning the gene expression analysis, the next step is the investigation of the clusters created by using clustering techniques. These techniques will be used to find group of genes and tissues that are related. Future experiments for gene recognition will investigate how the combination of different signals can improve the correct classification of sequences and how previous information obtained from experts can be employed to improve the performance obtained by different classifiers.

Acknowledgments

The authors would like to thank CNPq, FAPESP and FAPERGS for their support. The authors would also like to acknowledge the anonymous reviewers of the preliminary version for their comments and suggestions, as well as Ana Carolina Lorena, Bruno Feres, Eduardo Spinosa, Katti Faceli, Murilo Naldi and Rodrigo Bianchi for their help with the manuscript.

References

[1] J. E. Allen, M. Pertea, and S. L. Salzberg. Computational gene prediction using multiple sources of evidence. *Genome Research*, 14(1):142–148, 2004.

[2] P. Baldi and S. Brunak. *Bioinformatics: the Machine Learning approach.* MIT Press, 2nd edition, 2001.

[3] G. E. A. P. A. Batista, A. L. C. Bazzan, and M. C. Monard. Balancing training data for automated annotation of keywords: a case study. *Revista Tecnologia da Informação*, 3(2):15 – 20, 2004. Also appeared in Proc. of the Second Brazilian Workshop on Bioinformatics. SBC, 2003. p.21 – 28.

[4] G. E. A. P. A. Batista, M. C. Monard, and A. L. C. Bazzan. Improving rule induction precision for automated annotation by balancing skewed data sets. In *Knowledge Exploration in Life Science Informatics: International Symposium (KELSI)*, volume 3303 of *Lecture Notes in Computer Science*, pages 20–32. Springer, Milan, 2004.

[5] R. J. Bayardo, Jr., W. Bohrer, R. Brice, A. Cichocki, J. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. InfoSleuth: Agent-based semantic integration of information in open and dynamic environments. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 195–206, New York, 1997. ACM Press.

[6] A. L. C. Bazzan, S. Ceroni, P. M. Engel, and L. F. Schroeder. Automatic annotation of keywords for proteins related to *Mycoplasmataceae* using machine learning techniques. *Bioinformatics*, 18(S2):S1–S9, October 2002.

[7] A. L. C. Bazzan, R. Duarte, A. N. Pitinga, Schroeder L. F., S. C. Silva, and F. A. Souto. ATUCG–an agent-based environment for automatic annotation of genomes. *International Journal of Cooperative Information Systems*, 12(2):241–273, June 2003.

[8] Amir Ben-Dor, Nir Friedman, and Zohar Yakhini. Overabundance analysis and class discovery in gene expression data. Technical Report AGL-2002-4, Agilent Laboratories, 2002.

[9] R. Bianchi, C. R. Milar, and A. C. P. de L. F. de Carvalho. Promoter and splice site identification with explanation by extracting knowledge from artificial neural networks. *Revista da Tecnologia da Informação*, 3(1):21–26, 2003.

[10] A. Brazma and J. Vilo. Gene expression data analysis. *FEBS Letters*, 480(1):17–24, 2000.

[11] L. A. Breslow and D. W. Aha. Simplifying decision trees: a survey. *Knowledge Engineering Review*, 12(1):1–40, 1997.

[12] K. Bryson, M. Luck, M. Joy, and D. Jones. Applying agents to bioinformatics in GeneWeaver. In *Proc. of the Fourth Int. Workshop on Collaborative Information Agents*, Lect. Notes in Computer Science. Springer-Verlag, 2000.

[13] M. Burset and R. Guig. Computational methods for the identification of differential and coordinated gene expression. *Human Molecular Genetics*, 8(10):353–367, 1999.

[14] R. Castelo and R. Guigo. Splice site identification by idlbns. *Bioinformatics*, 20:i69–i76, 2004.

[15] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.

[16] W. W. Cohen. Fast effective rule induction. In *Proc. of the 12th International Machine Learning Conference*, pages 115–123, San Francisco, CA, 1995. Morgan Kaufman.

[17] L. Coin and R. Durbin. Improved techniques for the identification of pseudogenes. *Bioinformatics*, 20:i94–i100, 2004.

[18] M. W. Craven and J. W. Shavlik. Machine learning approaches to gene recognition. *IEEE Expert*, 9(2):2–10, Abril 1994.

[19] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and other kernel-based learning methods)*. Cambridge University Press, Cambridge, UK, 1 edition, 2000.

[20] B. Feres de Sousa and A. de Carvalho. Gene selection using genetic algorithms. In *Lectures Notes in Computer Science, 1st International Symposium on Biological and Medical Data Analysis (ISBMDA-2004)*, volume LNCS 3337, pages 479–490. Springer Verlag, August 2004.

[21] B. Feres de Sousa and A. de Carvalho. Gene selection based on multi-class svms and genetic algorithms. *Accepted for the Journal Genetics and Molecular Research*, 2005.

[22] K. Decker, X. Zheng, and C. Schmidt. A multi-agent system for automated genomic annotation. In *Proc. of the Int. Conf. Autonomous Agents*, Montreal, 2001. ACM Press.

[23] J. Dopazo, E. Zanders, I. Dragoni, G. Amphlett, and F. Falciani. Methods and approaches in the analysis of gene expression data. *Journal Immunol. Methods*, 250(1/2):93–12, 2001.

[24] E. R. Dougherty, J. Barrera, M. Brun, S. Kim, R. M. Cesar Junior, Y. Chen, M. Bitner, and J.M. Trent. Inference from clustering with application to gene expression microarrays. *Journal of Computational Biology*, 9(1):105–126, 2002.

[25] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. John Wiley & Sons, 2001.

[26] K. Faceli, A. C. P. L. F. de Carvalho, and W. A Silva Jr. Evaluation of gene selection metrics for tumor cell classification. *Genetics and Molecular Biology*, 27(4):651–657, 2004.

[27] Y. Freund and R. Schapire. A short introduction to boosting. *J. Japan. Soc. for Artif. Intel.*, 14(5):771–780, 1999.

[28] H. Frohlich and et al. Feature selection for support vector machines by means of genetic algorithms. In *15th IEEE International Conference on Tools with AI*, pages 142–148, 2003.

[29] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing, Boston, MA, 1989.

[30] T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caliguiri, C.D. Bloomfield, and E.S. Lander. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, 5439(286):531–537, 1999.

[31] S. Gupta, D. Zink, B. Korn, M. Vingron, and S. A. Haas. Genome wide identification and classification of alternative splicing based on est data. *Bioinformatics*, 20(16):2579–2585, 2004.

[32] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

[33] J. Khan and et al. Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. *Nat Med*, 7(6):673–9, Jun 2001.

[34] E. Kretschmann, W. Fleischmann, and R. Apweiler. Automatic rule generation for protein annotation with the C4.5 data mining algorithm applied on SWISS-PROT. *Bioinformatics*, 17:920–926, 2001.

[35] A. Lapedes, C. Barnes, C. Burks, R. Farber, and K. Sirotkin. Application of neural networks and other machine learning algorithms to dna sequence analysis. In G. Bell and T. Marr, editors, *Computers and DNA, SFI Studies in the Sciences of Complexity*, volume 7, pages 157–182, 1989.

[36] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery Data Mining*. Kluwer Academic Publishers, Boston, USA, 1 edition, 1998.

[37] A. C. Lorena, G. E. A. P. A. Batista, A. C. P. L. F. Carvalho, and M. C. Monard. Splice junction recognition using machine learning techniques. In *Proc. of the Brazilian Workshop on Bioinformatics (WOB)*, pages 32–39, 2002.

[38] A. C. Lorena and A. C. P. L. F. de Carvalho. Human splice site identification with multiclass support vector machines and bagging. In *Lecture Notes on Artificial Intelligence, Joint 13th ICANN and 10th ICONIP*, volume 2714, pages 234–241, Istanbul, 2003. Springer Verlag.

[39] A. C. Lorena and A. C. P. L. F. de Carvalho. Evaluation of noise reduction techniques in the splice junction recognition problem. *Genetics and Molecular Biology*, 27(4):665–672, 2004.

[40] D. Shalon M., R. Heller, A. Chai, P. O. Brown, and R. W. Davis. Parallel human genome analysis: microarray-based expression monitoring of 1000 genes. *Proc. Natl. Acad. Sci.*, 93(20):10614–10619, 1996.

[41] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[42] D. Murphy. Gene expression studies using microarrays: Principles, problems, and prospects. *Advan. Physiol. Educ.*, 26(4):256–270, 2002.

[43] Auer P., Holte R., and Maass W. Theory and applications of agnostic PAC-learning with small decision trees. In *Proc. of the 12th International Machine Learning Conference*. Morgan Kaufmann, 1995.

[44] P. Pavlidis, T. Furey, M. Liberto, D. Haussler, and W. Grundy. Promoter region-based classification of genes. In *Proc. of the Pacific Symposium on Biocomputing*, pages 151–163, 2001.

[45] A. Pedersen, P. Baldi, Y. Chauvin, and S. Brunak. The biology of eukaryotic promoter prediction: a review. *Computers and Chemistry*, 23:191–207, 1999.

[46] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[47] S. Rampone. Splice-junction recognition on gene sequences (DNA) by BRAIN learning algorithm. In *Neural Networks Proceedings*, volume 1, pages 774–779. IEEE World Congress on Computational Intelligence, 1998.

[48] Ryan Rifkin and et al. An analytical method for multi-class molecular cancer classification. *SIAM Review*, 45(4):706–723, 2003.

[49] L. F. Schroeder and A. L. C. Bazzan. A multi-agent system to facilitate knowledge discovery: an application to bioinformatics. In *Proc. of the Workshop on Bioinformatics and Multi-Agent Systems (BIXMAS'2002)*, pages 44–50, Bologna, Italy, 2002.

[50] J. C. Setúbal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishing Company, 1997.

[51] J. W. Shavlik. Finding genes by case-based reasoning in the presence of noisy case boundaries. *Proc. of the DARPA Cased-Based Reasoning Workshop*, pages 327–338, 1991.

[52] D. K. Slonim, P. Tamayo, J. P. Mesirov, T. R. Golub, and E. S. Lander. Class prediction and discovery using gene expression data. In *RECOMB*, pages 263–272, 2000.

[53] M. Souto, A. Lorena, A. Delbem, and A. de Carvalho. Técnicas de aprendizado de máquina em problemas de biologia molecular (machine learning techniques in molecular biology problems). In *I Jornada de Atualização em Inteligência Artificial (in Portuguese), JAIA 2003, XXI Congresso da Sociedade Brasileira de Computação , SBC'2003*, pages 1–50, August 2003.

[54] G. D. Stormo, T. D. Schneider, and L. Gold. Use of the Perceptron algorithm to distinguish translation inition sites in E. coli. *Nucleic Acids Research*, 9:2997–3011, 1982.

[55] E. C. Uberbacher, J. R. Einstein, X. Guan, and R. J. Mural. Gene recognition and assembly in the GRAIL system: Progress and challenges. *Proc. of the International Conference on Bioinformatics, Supercomputing and Complex Genome Analysis*, pages 465–476, 1993. World Scientific, Singapura.

[56] G. Wahba. Support vector machine, reproducing kernel hilbert spaces and the randomized gacv. In A. Smola B. Scholkopf, C. Burges, editor, *Advances in Kernel Methods-Support Vector Learning*, Cambridge, MA, 1999. MIT press.

[57] Y. Xu, R. Mural, J. Einstein, M. Shah, and E. Uberbacher. GRAIL: A multi-agent neural network system for gene identification. In *Proceedings of the IEEE*, volume 84, pages 1544–1552, October 1996.