Roma Tre University
Ph.D. in Computer Science and Engineering

Logic mining techniques for biological data analysis and classification

Emanuel Weitschek

Logic mining techniques for biological data analysis and classification

A thesis presented by
Emanuel Weitschek
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Engineering

Roma Tre University
Dept. of Informatics and Automation

April 2013

Committee:
Dr. Paola Bertolazzi

Reviewers:
Prof. Mourad Elloumi
Prof. Indra Neil Sarkar

In theory, there is no difference between theory and practice.
But, in practice, there is.

To my wonderful family

Acknowledgments

I am grateful to my tutor Paola Bertolazzi for her serious guidelines, precious advice and strong support during the last years of my PhD program.

A big thank to Giovanni Felici for introducing me in data mining and for the successful research collaborations.

This dissertation is dedicated to all members of my research team. They showed me how amusing and fun research can be.

A very special thank to all my external collaborators: Robin Van Velzen, Mike Trizna, Ivan Arisi, Massimo Ciccozzi, Marco Ciotti, Alessandra Pierangeli, Alessandra Lo Presti, Dimitris Polychronopoulos and Fabio Cunial, without you my success in several data mining applications would not have been possible.

A special thank to Guido Drovandi, a very good research colleague, computer engineer and finally a nice friend.

Thanks to Daniele Santoni, a skilled bioinformatician.

I would like to thank Cristina De Cola, Marianna De Santis, Stefania De Angelis and Giulia Fiscon my great room neighbors!

Thanks to all my friends for supporting me during the PhD period. I really enjoyed the time spent with you.

I would like to thank the members of IASI CNR for their hospitality in their research facilities.

Finally, I would like to thank all the beautiful and wonderful periods and moments of my life.

Contents

Introduction

Advances in molecular biology lead to an exponential growth of biological data, also thanks to the support of computer science. The primary sequences data base GenBank is doubling its size every 18 months [HDL+12], actually consisting in more than 160 billions sequences. The 1000 genomes project [CZBS+12] released whole DNA sequences of a large number of individuals, producing more than 3000 billions DNA base pairs. Analyzing these enormous amount of data is becoming very important in order to shed light on biological and medical questions. The challenges are in managing this huge amount of data, in discovering its interactions and in the integration of the biological know-how. Extracting relevant information in huge amounts of biological data is one of the most challenging problems. Biological experiments and clinical interactions, are just some examples of structures composed of thousands and thousands of elements. Clearly, it is impossible to have a visual or textual human readable representation of all these data. It is difficult, even for an expert, to manually distinguish the important information from those that are not.

The analysis of biological data requires new methods to extract compact and relevant information; effective and efficient computer science methods are needed to support the analysis of complex biological data sets. The interdisciplinary field of data mining, which guides the automated knowledge discovery process, is a natural way to approach the complex task of biological data analysis [Ell12]. Data mining deals with structured and unstructured data, that are, respectively, data for which we can give a model or not. For example, in biological contexts it is important to highlight those substrings that are frequent

in DNA or protein sequences, or the most relevant characteristics that can be used to generate realistic biological sequences.
The objective of this dissertation is to study and apply methods to manage and to retrieve relevant information in biological data sets, contributing to the advancement of research in this promising field. In this dissertation new data mining methods are presented and proven to be effective in many biological data analysis problems. The particular field of logic data mining, where a data classification model is extracted in form of propositional logic formulas (in disjunctive or conjunctive normal form), is investigated and a new system, called Data Mining Big (DMB), for performing a complete knowledge discovery process is described. The system presents new methods for discretization, clustering, feature selection and classification. All methods have been integrated in three different tools: BLOG, MALA and DMiB, the first dedicated to the classification of species, the second to the analysis of gene expression profiles and the third for multipurpose use. These tools were applied to species classification with DNA Barcode sequences, viruses identification, gene expression profiles analysis, clinical patient characterization, tag snp classification, non coding DNA identification and whole genome analysis. A comparison with other data mining methods was performed. The analysis results were all very positive and contributed to the gain of important additional knowledge in biology and medicine, like the detection of nine core genes able to distinguish Alzheimer diseased versus control experimental samples, or the identification of the characteristic nucleotides positions in the five actually known human polyomaviruses. Moreover, a new technique based on alignment free sequence analysis and logic data mining, that is able to perform the classification of sequences without the strict requirement of computing an alignment between them, is presented. This is a major advantage as the problem of alignment is computationally hard and many biological sequences are not alignable, because of their intrinsic nature, e.g. non coding regions. Also in this case the performed experiments on whole genomes and on conserved non encoding elements show the success of this approach. The models extracted from several analysis are logic formulas able to characterize the different classes of the data set in a clear and compact way. The model is a strong plus for the domain expert, that gains a precious and directly interpretable knowledge.

Goal of this work is to show that logic data mining methods - as DMB - are recommended for performing biological knowledge discovery. This is proven by several experiments that solve important life science issues with the analysis of real data sets.

The dissertation is structured in four chapters.
Chapter 1 introduces the field of data mining with particular stress on biological data. Some methods and algorithms of data mining main applications - classification and clustering - are illustrated.
In chapter 2 the logic data mining system DMB is presented, its approach to the knowledge discovery process is described in detail, and its computational steps are illustrated: sampling, discretization, clustering, feature selection, logic formulas extraction, noise reduction and classification.
Chapter 3 presents many practical biological problems and experiments, where the logic data mining system DMB has been successfully applied, spanning from DNA Barcode sequences classification, to gene expression profiles analysis, to polyomaviruses identification, to clinical patient trials, and to tag snp characterization.
Chapter 4 presents a new technique based on alignment free sequence analysis and logic data mining, that classifies biological sequences without the strict requirement of alignment or of an overlapping gene region. Additionally, an application of an alignment free metric to next generation sequencing reads filtering is described.
Finally the conclusions and future tasks are drawn.
The work conducted during the PhD program has been devoted to the design and the development of existing and new methods for data mining. A clear identification of the original contributions can be derived from the publication list reported at the end of the dissertation.

Chapter 1

Data mining and classification in bioinformatics

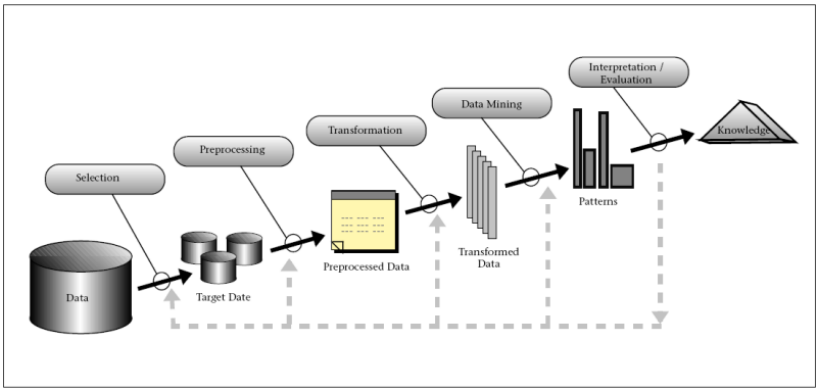
1.1 Introduction

Due to the large amount of biological data available and to its continuing exponential growth, the life scientists have to be supported by automatic tools and methods for knowledge extraction. The best candidate discipline is data mining, an interdisciplinary field, that involves computer science, statistics and database management. In this chapter data mining and its two main applications - classification and clustering - are introduced. Data mining is a way of recognizing common properties in a set of data. Two main problems are tractable by data mining: classification and clustering.
Classification is the action of assigning an unknown object into a predefined class after examining its characteristics [DFE09] and possibly after computing a clear human interpretable model of the data. This last affirmation is very important for the application of data mining to biological problems, where the scientist has to get an insight of the analyzed data.
Clustering partitions objects into groups, such that similar or related objects are in same clusters [TSK05]. This technique is also precious for solving biological problems, where the experimental data has to be grouped without an a prior knowledge.
These two main aspects of data mining are described in detail in this first chapter.
Finally, a brief section is dedicated to the relation between biology, data mining

and bioinformatics, a discipline that merges computer and life science.

1.2 Data mining

Data mining is defined as the extraction of knowledge with computer science algorithms for discovering hidden information in heterogeneous data sets [WF05]. It is the process of finding previously unknown patterns and trends in databases and using that information to build predictive models [Kin98]. An alternative definition is the process of data selection and exploration and building models using vast data stores to uncover previously unknown patterns [Mil00]. Data mining is an interdisciplinary field, it involves computer science, statistics and database management.
Data mining is a discipline, that emerged during the early nineties [KT10]. It was standardized with the Cross-Industry Standard Process for Data Mining (CRISP-DM) [fC00] by a consortium of leading data mining users and suppliers [She00]. The CRISP-DM Consortium defined the following steps for the data mining process: business understanding, data understanding and preparation, modeling, evaluation and deployment.
Business understanding is the identification of the business objectives. Data understanding and data preparation are prior the modeling step, the real data analysis stage. The evaluation stage aim is to interpret the results by applying the models to the data set and to visualize them.
The main aim of data mining is to extract knowledge from data. The knowledge discovery process was described by Fayyad et al. [FPSS96] and is drawn in figure 1.1. It starts from a structured or unstructured database; the first step is the selection and sampling of the data which outputs the target data. Then a preprocessing and cleaning phase is applied which produces cleaned data. The transformation and reduction step transforms the data, before the data mining step computes the models or patterns. Finally, the visualization and evaluation steps aim is to represent the knowledge conveyed by the models or patterns.
The main applications of data mining are classification, clustering, association, sequencing, predictive analysis, estimation, description and visualization.
Classification is the automatic identification of an object into a set of classes. A target categorical class variable is predicted after examining the attributes of the object.
Clustering - called also group analysis - is the process of grouping a collection of objects in similar ones according to a certain distance measure.



(a)

Figure 1.1: The knowledge discovery process in [FPSS96]

The aim of association is to find which attributes relate to each other.
The aim of estimation is to give a certain interval in which a numerical attribute will fall in.
Predictive analysis groups classification and estimation.

Data definition

In a general data mining problem a data set is a collection of "objects" organized in records. Each object has a fixed number of attributes or features, which can be discrete or continuous. A data object is described by a set of features represented as a multidimensional vector. The features can be quantitative or qualitative, discrete (binary) or continuous, nominal or ordinal.

1.3 Classification

Classification is the action of assigning an unknown object into a predefined class after examining its characteristics [DFE09]. It is an important data mining task, where the value of a variable, named class variable, is predicted on the basis of some independent variables. Examples from biology include separating healthy specimen from diseased ones, identifying the living species, predicting the secondary structure of proteins. Classification is called also supervised

learning: unknown objects are assigned to a class using a model that was derived from objects with a known class. These objects are called the training set.
The input data for a classification problem is a collection of objects $\mathcal{O}$ organized in records. Each object is characterized by a tuple $(\mathbf{x}, y)$, where $\mathbf{x}$ is the feature set and y is the class. The class must be a discrete attribute, whereas the features can be also continuous. Classification is the task of learning a function f that maps each attribute set **a** to one of the predefined classes y [TSK05]. The function is called also classification model. More formally, we consider the object to classify in vectors of R^n, that means for each object we know the values of n features, we know that the objects belong to m classes and that each object x belongs exactly to one class and for each object x $\in R^n$ we denote its class by c(x) $\in$ 1, ..., m. The classification problem is formulated as follows. Given t objects $x_1, \ldots, x_t$ whose class is known, build a function and use it to classify new objects, whose class is unknown. This function is called also classifier and is defined as $c' : R^n \to 1, \ldots, m$. The goal is to have $c'(x) = c(x)$ and $\forall x$, the predicted class should be the correct class. A classification function is learned from (or fitted to) a training data and applied to a test data. This process is called supervised learning. In the case of descriptive modeling a classification model is intended as an explanatory tool to distinguish between objects of different classes. In the case of predictive modeling the classification model is used to predict the class of unknown records.

Classification methods

A classification method is a systematic approach for building a classification model from the training data set [TSK05]. The most used and successful methods are Decision Trees, Logic Formulas, Neural Networks, Support Vector Machines and Naive Bayes classifiers. Each method is characterized by a learning algorithm that computes a model for representing the relationship between the class and the attributes set of the records. This model should fit to the input data and to new data objects belonging to the same classes of the input data. A general approach for solving classification problems is:

1. Split the data in training set and test set;
2. Perform training;
3. Compute the classification model;
4. Apply the model to the training and to the test set;

5. Evaluate.

A training set containing objects with known classes is provided to the classification method. The training set is used to build a classification model. The model is applied to the training set and to a test set, containing objects with unknown class labels. Finally, the performance of the classification model is evaluated. The evaluation is done by counting the number of training and test objects correctly and incorrectly classified. These counts are reported in a so-called *confusion matrix*. Each element e_{ij} of this matrix represent the number of records from class i predicted to be of class j. The correct classified elements are on the main diagonal of the confusion matrix. The confusion matrix gives

	Class 0	**Class 1**
Class 0	O_{00}	O_{01}
Class 1	O_{10}	O_{11}

Table 1.1: A confusion matrix

a detailed view of the classification performance. Often it is also necessary to have a compact performance indicator. We can summarize the classification correctness by the percentage of correct, wrong and not classified items. Also additional performance metrics are introduced during the evaluation of the classification performance. With reference to a specific class, we have:

- *true positives (tp)*: objects of that class recognized in the same class (e.g., in the case of the confusion matrix represented in table 1.1, for class 0 the number of true positive is O_{00}, while for class 1 it is O_{11});
- *false positives (fp)*: objects not belonging to that class recognized in that class (e.g., for matrix in table 1.1, the number of false positives for class 0 is O_{01}, while for class 1 it is O_{10});
- *true negatives (tn)*: objects not beloging to that class and not recognized in that class (e.g., in the case of the confusion matrix represented in table 1.1, for class 0 the number of true negatives is O_{11}, while for class 1 it is O_{00});
- *false negatives (fn)*: object belonging to that class not recognized in that class (e.g., for matrix in table 1.1, the number of false negatives for class 0 is O_{10}, while for class 1 it is O_{01});

- *accuracy (a), also called correct rate*: $a = \frac{tp+tn}{tp+tn+fp+fn}$;
- *precision (p), also called positive predictive value (ppv)*: $p = \frac{tp+tn}{tp+tn+fp+fn}$;
- *recall (r), also called true positive rate or sensitivity*: $r = \frac{tp}{tp+fn}$;
- *specificity (s), also called true negative rate*: $s = \frac{tn}{tn+fp}$;
- *error rate (e)*: $e = \frac{fp+fn}{tp+fp+tn+fn}$.

The most important classification methods are described in the next subsections: Decision Trees [Qui93, Qui96], Artificial Neural Networks [DD01], Support Vector Machines [Vap98, CST00], Nearest Neighbor classifiers [Das90], Neighbor Joining [SN87], Error Correcting Output Codes [BFF10], and Logic Data Mining (Logic Formulas for classification) [FT02]. The following sections describe the most important classification methods that are used for biological data.

Decision Trees

Decision trees are called also classification trees. In a decision tree each node is associated to a binary predicate, that represents the attributes of the objects in the data set. The special class attributes are represented in the tree by the leaves. Every node has two outgoing branches, one is associated to objects whose attributes satisfy the predicate, the other to the one who do not. Therefore the path from the root to each node represents a set of conditions and includes a certain number of objects that meet the set of conditions. Generally, decision trees are constructed from the training data with recursive procedures. The most used rule is the entropy rule or information gain rule, which finds at each node a predicate that optimizes an entropy function, of the defined partition. In bioinformatics the most used tree decision classifiers, such as C4.5 [Qui93, Qui96], rely on entropy rules.
The classification process with a decision tree is achieved in the following way: for a new input object the predicates are applied to its attributes starting from the root, each node defines the path split, the final leaf outputs the class attribute of the object.
Potential limitations of the method are:

- the computation of an optimal decision tree is an NP-complete problem
- decision trees can be unstable when the training data changes (sampling)

- complexity (too much splits and large trees)

Artificial Neural Networks

The Artificial Neural Network is a computational model, which aims to simulate the human brain structure [DD01]. The biological neural system is composed by neurons, connected with other neurons. The links between neurons are called axons, which transmit nerve impulses form one neuron to another. A neuron is connected with an axon via dendrites. A synapse is the contact point between a dendrite and an axon. The human brain learns by changing the strength of the synaptic connection between neurons stimulated by an impulse. An artificial neural network is engineered following this model: it is composed of interconnected nodes and directed links.
The simplest attempt to build an artificial neural network is the perceptron [WL90]. The perceptron is composed of two types of nodes: input nodes, which represent the input attributes, and an output node, which represents the model output. The nodes are the neurons. Each input node is connected via a weighted link to the output node. The weight is the strength of a synaptic connection between neurons. A perceptron calculates its output value by computing a weighted sum on its input, subtracting a noise factor t from the sum and then evaluating the sign of the result.
An input node simply transmits the value to the outgoing link without performing any computation. The output node is a mathematical system that calculates the weighted sum of its inputs, subtracts the noise and then produces the output by applying the sign function to the resulting sum.
The multilayer artificial neural networks allow the resolution of more advanced classification problems. The network has a more complex structure. It contains some intermediate layers between the input and output layers. These layers are called hidden layers and its nodes hidden nodes. In feed forward neural networks the nodes are only connected with nodes of the next layer. In recurrent neural networks the nodes may be connected also with nodes from the same layer.
The network may also use different activation functions, like linear, sigmoid and hyperbolic tangent. These functions allow the resolution of non linear classification problems, e.g. the XOR problem.
Potential limitations of the method are:

- the necessity to set up the right topology to avoid overfitting;
- it is susceptible to noise;

- the high computational cost for training.

Support Vector Machines

Support Vector Machines (SVM) [Vap98, CST00] are particularly suited for binary classification tasks. In this case, the input data are two sets of n dimensional vectors, and a SVM tries to construct a separating hyperplane, which maximizes the margin (defined as the minimum distance between the hyperplane and the closest point in each class) between the two data sets. More, formally, given a training data T consisting of t points, where $T = \{(x_i, c(x_i))|x_i \in R^n, c(x_i) \in \{1, 1\}\}_{i=1}^{t}$ (in this case, the class for each point is labeled -1 or 1) we want to find a hyperplane $ax = b$ which separates the two classes. Ideally, we want to impose the constraints $ax_ib \geq 1$ for x_i in the first class and $ax_ib \leq 1$ for x_i in the second class. These constraints can be relaxed by the introduction of non-negative slack variables s_i and then rewritten as $c(x_i)(ax_i - b) \geq 1 - s_i$ for all x_i.
A training data with $s_i = 0$ is closest to the separating hyperplane $\{x : ax = b\}$ and its distance from the hyperplane is called margin. The hyperplane with the maximum margin is the optimal separating hyperplane. It turns out that the optimal separating hyperplane can be found through a quadratic optimization problem, i.e., minimize $\frac{1}{2}||a||2 + C\sum_{i=1}^{t} s_i$ over all (a, b, s) which satisfy the constraint $c(x_i)(ax_i - b) \geq 1 - s_i$. Here, C is a margin parameter, used to set a trade off between the maximization of the margin and minimization of classification error. The points that satisfy the previous constraint with the equality are called support vectors.
Support Vector Machines can also be used less effectively for multi class classification. Basically, in the multi class case, the standard approach is to reduce the classification to a series of binary decisions, decided by standard Support Vector Machines. Two such approaches are one-vs-the-rest and pairwise comparison (other, similar approaches exist in the literature). Assume there are m classes. In the one-vs-the-rest approach, a binary classifier that separates each single class from the union of the remaining classes, is built. Then, to classify a new point x, each of the m classifiers is ran. For each class k, assuming x is in k, the answers may have some inconsistencies. Let $\epsilon(x, k)$ be the number of errors that the m classifiers made under the hypothesis that x does in fact belong to k. Then, x is eventually assigned to the class $\overline{k}$ for which $\epsilon(x, bk)$ is minimum (i.e., the class consistent with most of the answers given by the classifiers).
In the pairwise comparison method, one trains a classifier for each pair of

classes, so that there are $m(m-1)$ independent binary classifiers. To classify a new data point x, each of these classifiers is run, and its output is viewed as a vote to put x in a certain class. The point is eventually assigned to the class receiving most votes.
Potential limitations of the method are:

- the choice of the right type of kernel function;
- the high computational requirements for multi-class problems;
- the output of a non human interpretable model.

Nearest Neighbor classifiers

The k-nearest neighbor algorithm is a classifier based on the closest training data in the feature space [Das90]. This is a very simple classifier which has been applied to a large number of biological classification problems, like DNA Barcode classification and gene expression analysis, etc. Given a training set of objects whose class is known, a new input object is classified by looking at its k closest neighbors typically in the Euclidean distance of the training set. Each of this neighbors can be imagined as casting a vote for its class. The input is eventually assigned to the class receiving the most votes (if $k = 1$, then the object is simply assigned to the class of its nearest neighbor). k is a positive integer, usually small, and its best value depends upon the data. While larger k reduce the influence of noise in classification, they tend to degrade clear-cut separation between classes. Usually the best k for a particular problem is found by some heuristic ad- hoc approach. One of the main drawbacks of this method is that it may be biased, i.e., classes which are over- represented in the training data set tend to dominate the prediction of new vectors. To counter this phenomenon, there are correcting strategies that take this bias effect into account for better classification. Another main drawback of this approach is that no model is computed to classify the data.
Other potential limitations of the method are:

- the requirement of a proximity measure;
- the non computation of a classification model;
- the high computational costs during the classification phase: every record in the test set has to be compared with every record in the training set;
- it is susceptible to noise.

Neighbor Joining

The Neighbor Joining (NJ) algorithm [SN87] is a bottom-up clustering method. Computational efficiency being its main advantage, Neighbor joining is the most widely used method for classifying DNA data. The underlying assumption is that the DNA sequences of distinct species form discrete clusters in the phylogenetic tree, because genetic variation within species is smaller than that between species. A phylogenetic tree reports the relationships in evolution between biological species by analyzing genetic similarities and common characteristics. NJ tries to compute an an approximation of a phylogenetic tree called phenetic tree.
This method iteratively joins the two elements of the data set V that are at the minimum distance. The distance is computed with the Q-matrix, an $N \times N$ matrix (where N is the number of elements) that contains in position (u, v) the distance between element u and v defined as follows:

$$q_{uv} = d_{uv}(N-2) - \sum_{w \in V} d_{uw} - \sum_{w \in V} d_{vw}$$

where d_{uv} is the distance from u to v in a distance matrix D. The neighbor joining algorithm performs the following steps:

1. compute the Q-matrix of the set of elements V from the distance matrix D;

2. add a node w to the phylogenetic tree joining the two element $(u, v \in V)$ with the lowest value in the Q-matrix. Nodes u and v are removed from V and w is inserted;

3. calculate the new distances D of the nodes from the new node w. The new distances from w are defined as $d_{wz} = \frac{1}{2}(d_{uz} + d_{vz} - d_{uv})$; and

4. start the algorithm again with the new V set and D matrix.

Sequences of unknown specimens can subsequently be included in the tree to see in which cluster they appear. NJ is based on the minimum evolution criterion: in presence of multiple hits at equal sites particular distance measure are adopted; the resulting topology is derived according to the smallest value of the sum of all branches and is selected as an approximation of the correct tree.

Error Correcting Output Codes

The ideas underlying Error Correcting Output Codes (ECOC) classification comes from research on the problem of data transmission over noisy channels. The method is based on d binary classifiers, each of which defines a binary partition of the union of the m classes. For each input object, these classifiers produce as output a d-ary vector over $C = \{-1, 1\}^d$, which can be seen as the *codeword*, or *signature*, of the object. Each class is assigned a unique codeword in C. The class codewords can be arranged in an $m \times d$ matrix, which has the property that: no column is made of only $1s$ or $-1s$ (that classifier would not distinguish any class from the others), and no two columns are identical or complementary (they would be the same classifier). Under these conditions, the maximum number of classifiers (columns) possible is $2^{m-1} - 1$. If the minimum Hamming distance between any two matrix rows is h, then even if a row undergoes up to $\lceil (h-1)/2 \rceil$ bit-flips, its original correct value can still be recovered.
The classification is performed as follows. Given a new input object x, its codeword $w(x)$ is computed via the d classifiers. Then, the object is assigned to the class whose codeword has smallest Hamming distance to $w(x)$. The method performs its best with codewords of maximum length (exhaustive coding). In this case, codewords for each class can be built so that the Hamming distance between each two codewords is $(2^{m-1} - 1)/2$. The method was proposed in [DB95]. One of its limitations is that the number of classifiers is exponential in the number of classes.

Boosting

Boosting is the process by which a powerful classifier is built by the incremental use of more classifiers. The underlying idea is that, even if none of the classifiers performs particularly well by itself, when they are taken together (weighted with different weights), they yield quite accurate classifiers. Each time a new weak classifier is added, the training data is re-weighted. In particular, objects that are misclassified are given more importance (higher weight) and object correctly classified see their weight decreased. This way, each next weak classifier must focus more on the objects that the preceding weak classifiers found more difficult to classify. A popular boosting algorithm, which has found also many applications in bioinformatics, is AdaBoost [FS97].

Logic Data Mining

In this type of classification the classifier uses logic propositional formulas in disjunctive or conjunctive normal form ("if then rules") for classifying the given records, this classification method is also called ruled based. In [TSK05] the following description of ruled based classifiers is given: The formulas of the model generated by the classifier are normally constituted by literals in disjunctive normal form: $R = (r_1 \, or \, r_2 \, or \, \ldots \, or \, r_k)$. R is the formula set and r_i are the classification formulas or rules or disjuncts. Each classification formula can be represented in the following way: r_i: (Condition)$\rightarrow y_i$. The left hand side of the formula is called antecedent or precondition. The antecedent contains a conjunction of attribute tests:
Condition = $(A_1 \, op \, v_1) \, and \, (A_2 \, op \, v_2) \, and \, \ldots \, and \, (A_k \, op \, v_k)$
where A_j is an attribute, v_j is a value of the attribute A_j, and op is one of the following logical operators: $=, \neq, \geq, \leq, >, <$. Each attribute value pair $(A_j \, op \, v_j)$ is a conjunct. The right hand side of the of the formula is called consequent. A formula r covers a record x if the precondition of r matches the attributes of x. r is fired or triggered whenever it covers a record. A logic formula based classifier classifies on the basis of the formula triggered by the record. There are two properties that ensure that every record is covered by exactly one formula: mutual exclusion and exhaustion. A formula set has mutually exclusive formulas if every record triggers only one formula and no two formulas are triggered by the same record. This ensures that every record is covered by at most one formula. A formula set has exhaustive coverage if there is a formula for each combination of attribute values. This property ensures that every record is covered by at least one formula. This two properties are very difficult to obtain for a formula set and normally the formulas based classifiers do not output a formula set with these properties. If the formula set is not exhaustive then a default formula has to be added: r_d: $() \rightarrow y_d$. This formula covers the remaining records and the class y_d is typically assigned to the majority class of training records not covered by the existing formulas. If the formula set is not mutually exclusive, then a record can be covered by several formulas, which may classify the record in different and conflicting classes. To overcome this problem there are two approaches: the first approach consists in ordering the formulas, the second by assigning votes. By ordering the formulas in decreasing order based on a quality measure (e.g.: accuracy, coverage) a record is classified by the highest-ranked formula that covers it. Every record is classified by the "best" formula covering it. Often the formulas are also sorted on the class basis: formulas that belong to the same class appear together in

the formula set. The ordered formulas are called decision lists. In the second approach a record may trigger multiple classification formulas. Each triggered formula is considered as a vote for the class which the formula represents. The record is then assigned to the class that receives the highest number of votes. The vote may be weighted also by the formula's accuracy. With this approach the formulas are less susceptible to errors caused by the wrong selection of the classification formula. The computation of the model is also more efficient because the formulas do not have to be sorted or kept in a particular order. The main disadvantage is that every record has to be tested with every formula in the formula set.
For extracting a set of classification formulas there are two main classes of methods: direct extraction from data and indirect extraction, which extract the formulas from other classification models, like Decision Trees. As example for indirect method we can derive from a decision tree the logic formulas whose clauses are represented by the paths from the root to the leaves.
Direct methods partition the attribute space into smaller subspace so that all the record that belong to a subspace can be classified using a single classification formula. Examples of direct methods for computing separating formulas are RIPPER [Coh95], LSQUARE [FT02, FT06, Tru04], LAD [BIK$^+$96], RIDOR [GC95] and PART [FW98]. These approaches normally produce sub optimal formulas because the formulas are generated in a greedy way.
The major strength of classification formulas is the expressiveness of the models, that are very easy to interpret. Especially in biology and biomedical science the human interpretable model of the logic formulas is very useful to provide the scientist with a compact view of the analyzed data.
Potential limitations of the method are:

- it is susceptible to noise;
- it is high computationally demanding;
- the length of the logic formulas may be not interpretable by humans.

In chapter 2 of the dissertation a logic data mining system whose aim is to extract the logic classification formulas is presented. The system relies on the fact that the rules are determined using a particular problem formulation, the minimum cost satisfiability problem, or MINSAT, a well know and hard combinatorial optimization problem [FT02]. The original method, proposed in [FT02] and in [BFW09], has been implemented with improvements and extensions that lead to the system called DMB - Data Mining Big. The dissertation

focuses on Logic Data Mining, i.e. classification with logic formulas, and on DMB.

1.4 Clustering

Clustering or cluster analysis is a technique whose aim is to partition objects into groups, based on a distance function, such that similar or each other related objects are in same clusters and different or unrelated objects are in distinct ones [DFE09]. Cluster analysis groups the objects analyzing their attributes, without considering the class if available. For this fact it is also called unsupervised learning: it creates an implicit labeling of objects that is given by the clusters and is derived only from the attributes of the objects.
A typical clustering procedure is composed by the following steps [XI05]:
1) Feature selection:
In this step the fundamental attributes are selected and extracted, in order to reduce the data that has to be analyzed as input by the real clustering algorithm.
2) Clustering algorithm design or selection:
This is the real cluster analysis step, where the object are grouped together. For defining the objects partition a distance measure (or distance metric) between them has to be defined. The measure has to maximize the similarity of the objects in the same clusters and the dissimilarity of the objects in different clusters. Various distance metrics are presented below.
3) Cluster validation:
In this step the clusters are validated with statistical measures, like entropy, purity, precision, recall and correlation. According to these measures the clustering algorithm may be adjusted or fine tuned.
4) Results interpretation:
Finally the results have to be presented to the user and knowledge has to be transferred with a graphical interface and cluster lists.
Clustering algorithms can be divided in two main groups: partition algorithms (non hierarchical) and hierarchical algorithms. In the partition algorithms the objects are divided into a given number of clusters. In the hierarchical algorithms the objects are grouped in clusters starting from a initial clusters (that contains all the objects) or vice versa. The most important partition clustering algorithm is K-means [Mac67]. Widely used hierarchical clustering algorithms are AGNES (AGglomerative NESting)[KR90] and DIANA (DIvisive ANAlysis) [KR90]. A new type of clustering algorithm, particularly designed for the

attributes of a data set, is presented in chapter 2 section 2.6 of the dissertation, where the clusters are elected by a discretization procedure.

Distance measures

In cluster analysis the grouping of the objects occurs according to a distance measure or metric. A key point, when designing or applying a clustering algorithm, is the decision of which distance measure to choose. The distance measures has to maximize the similarity of the objects in the same clusters and to minimize the similarity of the objects in different clusters. A distance measure has to satisfy these properties:
a) Symmetry: $d(x, y) = d(y, x)$
b) Positivity: $d(x, y) \geq 0 \, \forall x, y$
c) Triangle inequality: $d(x, y) \leq d(x, z) + d(z, y) \, \forall x, y \, and \, z$
So the distance has to be symmetric, positive and between two objects x,y it has to be lower than or equal to the sum of the distances from x to a third object z and from y to z. Several distance measures have been defined for cluster analysis, the most commonly used are reported below.

Euclidean distance
Given two data sets X and Y of length n:
$X = x_1, x_2, ..., x_i, ..., x_n$
$Y = y_1, y_2, ..., y_i, ..., y_n$
The Euclidean distance between X and Y is defined as
$D(X, Y) = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$

Manhattan distance
Given two data sets X and Y of length n:
$X = x_1, x_2, ..., x_i, ..., x_n$
$Y = y_1, y_2, ..., y_i, ..., y_n$
The Manhattan distance between X and Y is defined as
$D(X, Y) = \sum_{i=1}^{N} |(x_i - y_i)|$

Minkowski distance
Given two data sets X and Y of length n:
$X = x_1, x_2, ..., x_i, ..., x_n$
$Y = y_1, y_2, ..., y_i, ..., y_n$
The Minkowski distance between X and Y is defined as
$D(X, Y) = (\sum_{i=1}^{N} (x_i - y_i)^r)^{\frac{1}{r}}$

where r is an input parameter, if $r = 1$ we have the Manhattan distance, if $r = 2$ the Euclidean distance if $r = \infty$ the maximum distance of the vector components.

Mahalanobis distance
Given two data sets X and Y of length n:
$X = x_1, x_2, ..., x_i, ..., x_n$
$Y = y_1, y_2, ..., y_i, ..., y_n$
The Mahalanobis distance between X and Y is defined as
$D(X,Y) = \sqrt{(X-Y) \circledast (X,Y)^{-}1(X-Y)^T}$
where $\circledast$ is the co-variance matrix operator.

Lagrange-Tchebychev distance
Given two data sets X and Y of length n:
$X = x_1, x_2, ..., x_i, ..., x_n$
$Y = y_1, y_2, ..., y_i, ..., y_n$
The Lagrange-Tchebychev distance between X and Y is defined as
$D(X,Y) = max_{1 \leqslant i \leqslant m} |X_i, Y_i|$

Correlation distance
Given two data sets X and Y of length n:
$X = x_1, x_2, ..., x_i, ..., x_n$
$Y = y_1, y_2, ..., y_i, ..., y_n$
The Correlation distance between X and Y is defined as
$D(X,Y) = p(X,Y)$
where p(X,Y) is the Pearson correlation coefficient
$p(X,Y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2}\sqrt{(y_i - \bar{y})^2}}$

The Pearson correlation varies between -1 and 1. This metric is commonly used in bioinformatics, e.g. in microarrays gene expression profiles analysis. We have a Pearson correlation coefficient near to 1 if the expression levels in X and Y increase or decrease at the same time, near to -1 if they have opposite behavior or near to 0 if their behavior is not related.
Other widely used distances are the Squared Euclidean distance, the Standardized Euclidean distance, the Angle between vectors and the binary vector similarity. For a complete survey on clustering distances see [XI05], [DFE09] and [KR90].

Clustering algorithms

Clustering algorithms can be divided in two main groups: partition algorithms (non hierarchical) and hierarchical algorithms. In the partition algorithms the objects are divided into a given number of clusters. In the hierarchical algorithms the objects are grouped in clusters starting from an initial cluster (that contains all the objects) or vice versa. The most important partition clustering algorithm is K-means [Mac67]. Widely used hierarchical clustering algorithms are AGNES (AGglomerative NESting)[KR90] and DIANA (DIvisive ANAlysis) [KR90]. A new type of clustering algorithm will be presented in chapter 2 section 2.6, where the clusters are elected by a discretization procedure. Clustering algorithms are sometimes also referred to as unsupervised classification approaches.

K-means

K-means is a partition clustering algorithm introduced by MacQueen in [Mac67]. The goal of K-means is to find K clusters of the provided data set, where K is a user specified integer parameter. K-means assigns a set of objects into K cluster with no hierarchical structure. It firstly chooses K initial objects as centroids and assigns each object to the closest centroid. In this initial step the algorithm already defines K clusters represented by the centroids of the objects. The centroids are then updated for every cluster according to its containing objects. With these new centroids a new assignment of the objects is then performed. The update and assignment procedure is iterated until no objects are moved between the clusters, this means that the centroids remain the same. The simplicity of K-means lets the implementation be very simple. The required time is $O(I * K * m * n)$ with I the number of iterations to converge, n the number of attributes and m the number of records. I can safely be bounded and normally all changes occur in the first iterations. The required storage is $O((m + K) * n)$. K-means is linear in time and space. The main drawbacks of K-means are: i) it does not find the optimal solution ii) the solution depends from the first random assignment of the clusters iii) the user has to specify the number of clusters to be found (K). In [XI05] various implementation and evolution of K-means are presented.

Hierarchical clustering algorithms

Hierarchical clustering algorithms use a similarity matrix to arrange data in a hierarchical structure. Two approaches are common in these algorithms:

```
K-means algorithm
1:  Select K points as initial centroids
2:  repeat
3:    form K clusters by assigning each point to its closest
    centroid
4:    Recompute the centroid of each cluster
5:  Until centroids do not change
```

Figure 1.2: K-means algorithm

Agglomeration and **Divisive**.
In the agglomeration approach the algorithm begins with each record as cluster and in each step it merges the two closest records based on a distance function. A pseudo code is given in figure 1.3.

```
Agglomeration hierarchical clustering algorithm
1:  Assign every record to a single cluster
2:  Calculate the proximity matrix
3:  repeat
4:    merge the closest two clusters
5:    update the proximity matrix (recalculate the distances)
6:  until only one cluster remains
```

Figure 1.3: Agglomeration hierarchical clustering algorithm

In the divisive approach the algorithm begins with a unique cluster of all records and at each step it removes the most distant record. A pseudo code is given in figure 1.4. As reported before the most used hierarchical clustering algorithms are AGNES [KR90] and DIANA [KR90].

1.5 Data mining and bioinformatics

Modern biology is frequently combined with computer science, leading to Bioinformatics: a discipline where biology and computer science are merged together in order to design and develop efficient methods for analyzing biological data, for supporting in vivo, in vitro and in silicio experiments and for automatically solving complex life science problems. In [LGG+01] Bioinformatics is

```
Divisive hierarchical clustering algorithm
1:  Assign all records to a single cluster
2:  repeat
3:    chose the most populated cluster
4:    update the proximity matrix (recalculate the distances)
5:    separate the most distant record from this cluster
6:  until all clusters contains a single record
```

Figure 1.4: Divisive hierarchical clustering algorithm

defined with the following sentence: Bioinformatics is conceptualizing biology in terms of macromolecules (in the sense of physical-chemistry) and then applying "informatics" techniques (derived from disciplines such as applied maths, computer science, and statistics) to understand and organize the information associated with these molecules, on a large-scale. The role of the bioinformatician, a computer scientist and biology domain expert, who is able to deal with these problems, is emerging: a figure that models and formulates the particular biological issue, proposes available or new computational methods and tools, performs the required data acquisition, preparation and analysis and finally provides a solution validated with statistical proofs. For accomplishing these tasks the bioinformatician has to perfectly know the currently evolving state of the art of different computational biology methods, algorithms and software available for solving distinct life science questions.
The exponential growth of biological data was originated by the DNA sequencing method invented by Sanger in early eighties. In late nineties significant advances in sequence generation techniques, largely inspired by massive projects such as the Human Genome Project, were contributing to this growth. Actually the genomic sequences are doubling every 18 months. Today high throughput data from modern parallel sequencing machines, like Illumina and Pacific Bioscience, are collected and huge amounts of biological data are currently available on public and private sources. Very large data sets, that are generated by several different biological experiments, need to be automatically processed and analyzed with computer science methods. Knowledge extraction and data mining techniques can discover models and patterns in these huge data sets to get an insight and compact information from the data. Data Mining is used to detect the most fundamental attributes that characterize the analyzed data set and the relations between them.
Ad hoc systems and methods specialized in biological data analysis are de-

manded. In the majority of cases, when dealing with biological data sets and problems, new tailored data mining tools, as the one presented in chapter 2 of the dissertation, are necessary. These tools must be able to extract the relevant information in large biological data sets by providing a compact and human interpretable model.

1.6 Conclusions

In this chapter the field of data mining has been introduced with particular emphasis on classification.
The most important applications of data mining - classification and clustering - have been illustrated. Regarding classification the following methods have been illustrated, describing their strengths and their potential limitations: Decision Trees [Qui93, Qui96], Artificial Neural Networks [DD01], Support Vector Machines [Vap98, CST00], Nearest Neighbor classifiers [Das90], Neighbor Joining [SN87], Error Correcting Output Codes [BFF10], and Logic Data Mining (Logic Formulas for classification) [FT02]. The Logic Data Mining method has been described accurately, as in the next chapter a systems that relies on it is proposed.
For clustering the following algorithms have been illustrated: K-means [Mac67] (partition clustering), AGNES (AGglomerative NESting)[KR90] and DIANA (DIvisive ANAlysis) [KR90] (hierarchical clustering).
Finally the focus has been devoted to the classification of biological data and on bioinformatics.

Chapter 2

DMB: A logic data mining system

2.1 Introduction

Data Mining Big (DMB) is a workflow framework of data mining methods and software tools engineered for the analysis and the classification of biological data, characterized by an underlying logic formalization and several optimization problems that are used to formulate the different steps of the data mining process. DMB includes a number of mathematical models and algorithms previously designed by the Data Mining Group of the Institute of Systems Analysis and Computer Science A.Ruberti" from the National Research Council of Italy, results of projects and collaborations with other research institutions. These algorithms and models have been extended, improved, implemented and integrated, leading to the DMB system.
DMB is composed of five main steps:

1. discretization: transformation of numerical features into discrete ones;
2. discrete cluster analysis: clustering of the features with the same behaviors;
3. feature selection: selection of the features that are considered to be more relevant;
4. logic formulas extraction;
5. classification.

In this chapter, the solutions adopted for discretization, clustering, feature selection and formula extraction are introduced, pointing out some computational issues related with their solution algorithms. Finally three different analysis tools are presented, specially designed for DNA Barcode sequences classification (BLOG), microarray gene expression profiles characterization (MALA), and a multipurpose tool (DMIB). The DMB tools have recently been engineered and made available on the web (`dmb.iasi.cnr.it`) with graphic user interfaces.

2.2 General presentation of DMB

Data Mining Big (DMB) is a system for performing knowledge extraction from biological data sets. The main characteristic of the DMB system is the production of logic formulas as a model to characterize the data. DMB takes as input a matrix containing the elements and their attributes, plus a class label for each element. Regardless of the form of the input data it returns as output an explanation in terms of logic formulas of the type if [(X is in Ax) and (Y is in Ay) or (Z is in Az)] then CLASS = C, where X, Y, Z are attributes of the elements, Ax, Ay, Az are set of possible values that the attributes can take, and C is one of the classes in which the elements are partitioned.
The methods adopted are based on several optimization models and algorithms [FT02, FT06, BFFL08, BFF10, BFL10]. New solutions for discretization, clustering, feature selection and formula extraction are introduced, pointing out some computational issues related with their solution algorithms.
The different modules of DMB are composed into flows to solve different types of classification problems. Each composition of modules that performs a complete analysis is called a "flow". Three different flows, resulting in three analysis tools, are presented, specially designed for DNA Barcode sequences classification (BLOG), gene expression profiles characterization (MALA), and a multipurpose tool (DMIB). The description of the currently implemented flows and a more detailed explanation for their usage are given in chapter 3 of the dissertation.
Barcoding with Logic Formulas (BLOG)
BLOG is a data mining software devoted to the automatic classification of species through the analysis of a small portion of mitochondrial DNA, called DNA Barcode. The aim of the system is to identify logic rules that are able to recognize the species (also referred as class) of a specimen by analyzing its DNA Barcode sequence. For a description of BLOG refer to [BFW09, vVWFB12,

WvVFB13] and chapter 3 of the dissertation.

MicroArray Logic Analyzer (MALA)

MALA is specifically designed for the analysis of Microarray data. The real values representing the gene expressions are discretized into a limited number of intervals for each cell of the array; the obtained discrete variables are then used to select a small subset of the genes that have strong discriminating power for the considered classes. For additional details refer to [WFB12, ADB$^+$11] and chapter 3 of the dissertation.

Data Mining in Big (DMIB)

Data Mining in Big (DMIB) is a general flow and tool for the deployment of our software for logic data analysis.

The method has been engineered and released with a graphic user interface on `dmb.iasi.cnr.it`.

DMB relies on the software architectural pattern Pipes and Filters [BHS07] for computing streams of data. The software tools and its aplications are described in chapter 3 of the dissertation.

2.3 Data types and representation

The terminology adopted in the chapter is introduced. It is assumed that each object (or record) is described by its attributes (or features). Each object belongs to one and only one class. The data set is composed of n objects, belonging to two or more classes. The objects (or records) are refered also as elements, and the attributes as features, when we are in a mathematical setting. The $i-th$ element of the data set is represented by the vector $f_i = (f_{i1}, f_{i2}, ..., f_{im})$, where $f_{ij} \in R$; the data matrix is represented by the sequence of vectors $f_1, f_2, ..., f_n$. The attribute data types that can be handled are quantitative or qualitative, discrete (binary) or continuous, nominal or ordinal. Given this matrix representation of the data set, when appropriate the elements may also be referred to as rows, while the features as columns. The classification method adopted is basically a two-class separation method, in the sense that it identifies the logic formula that separates the elements of one class in the data set from the remaining elements of the data set (such elements may belong to one or more classes). When needed, we refer to the two classes to be distinguished as class A and class B.

2.4 Sampling and the supervised paradigm of machine learning

Sampling is the step in the data mining process which prepares and divides the original data set in disjoints sets in order to be processed by the real data mining algorithm. The data set has to be split in training set and test set. The training set is used by the data mining algorithm to build the model of the data, this step is also called learning. The validity of the model is verified on the test set in terms of classification accuracy. The DMB methodology handles two types of data set sampling: percentage split and cross validation.
The percentage split sampling partitions the original data set in two disjoint sets, one for training and one for testing. The partition is done randomly according to a user defined percentage, e.g. 80% of the data for training and 20% for testing.
Cross validation is a standard sampling technique that splits the dataset in a random way in k disjoints sets, the data mining procedure is run k times with different sets. At a generic run k the k subset is used as test set and the remaining k-1 sets are merged and used as training set for building the model. Every of the k sets contains a random distribution of the data. The cross validation sampling procedure builds k models and each of this model is validated with a different set of data. Classification statistics are computed for every model and the average of these represents an accurate estimation of the data mining procedure performance.

2.5 Discretization

Part of the following section was published in [WFB12].

The formal problem of Logic Data Mining is defined in the following way: given a data set of objects $\mathbf{x} \in B^p$, where $B = \{0, 1\}$ and $p \geq n$, belonging to m different classes compute a boolean function b, such that $b(\mathbf{x}) = true$ when x is assigned to the correct class. Among the different approaches to solve this problem we remark the works by Felici and Truemper [FT02] and Boros [BIM99].
From the definition of the problem, a logic data miner is able to deal only with binary domains, so the objects features have to be transformed into discrete values and to be binarized. Therefore, Logic Data Mining methods apply when the data set $\mathcal{O}$ is binary, that is $\mathcal{O} \in B^n \times B^n$. When some of the variables

are represented by integer or real numbers we need to apply a transformation and convert each such variable into a new discrete variable that is suitable for treatment into any logic framework. This step amounts in determining a set of cutpoints over the range of values that each variable may assume and thus define a number of intervals over which the original variable may be considered discrete. In general, biological data sets are large arrays of integer or real numbers, that corresponds to measures on the items. Such data are not suitable for logic methods, and a transformation is needed to adapt the data sets. The approach is to binarize the data set, that is converting the real data set $\mathcal{O} \in R^n \times I$ where $I = 1, \ldots, m$ in $\mathcal{O} \in B^p \times B^p$. The classical approach is the definition of cutpoints for each feature, this consists of the identification of a set of intervals of values for each numeric feature.
DMB relies on a classification algorithm specifically designed to deal with binary domains, therefore it applies a transformation and converts each numeric variable into a new discrete variable that is suitable for treatment into a logic framework. This step is called *Discretization* and amounts in determining a set of cutpoints over the range of values that each variable may assume. DMB defines a number of intervals over which the original variable may be considered discrete (cutpoints). In binarized data, the new features can be viewed as binary, or logic, variables, that indicate whether the measure of one of the original real features belongs to a certain interval. Different methods for discretization have been proposed in literature (an extended list is available on `www.keel.es`). A widely adopted and effective discretization technique called CAIM is described in [KC04]. DMB supports two types of discretization, that differ on the rule adopted to select the first set of intervals. The first type uses unsupervised rules, the second supervised.

Unsupervised Discretization

Unsupervised discretization techniques do not take care of the class label and compute the cut points in base of information gain or of entropy of the given feature. In these methods the user has to submit the maximum number of desired cut points.

DMB Unsupervised Discretization

The method computes the number of samples in each interval and reduces the number of these intervals through the elimination of empty intervals and through unification of contiguous intervals that contain the same information.

To obtain an initial set of intervals for feature f_i we consider its mean μ_i and variance σ_i over the training items, and create a number of equal sized intervals symmetrical with respect to μ_i and proportional in size to σ_i. Once such intervals have been created, we iterate a set of steps that merge two adjacent classes if one of them is empty, if the distributions of elements in the classes is not altered (class entropy), and finally if the reduction obtained in the entropy of the feature is negligible.

For a given feature f_i, let K_i be the set of the intervals in which f_i is discretized; its entropy h_i is given by $-\sum_{k \in K_i} f_{ik} \log f_{ik}$, where $f_{lk} = p_{lk}/n$, and p_{lk} is the number of samples included in the interval k; since $h_i = 0$ if the number of intervals K_i is equal to 1, the goal is to obtain a good trade off between a high level of entropy and a small number of intervals. The procedure performs the following steps on the training data set:

1. For each feature f_i the mean value μ_i and the variance σ_i of the values of the feature over the items of the training set are computed;

2. N intervals around μ_i are computed, so that each interval width w_i is equal to $\sqrt{\sigma}_i/N$ (such intervals are indicated with C_{ik}, for $k = 1, .., N$);

3. For each interval C_{ik}, the total number p_{ik} of samples that are included in the interval is determined, together with their distribution in the classes;

4. The N intervals are reduced on the basis of the following three criteria:

   - if an interval is empty then it is unified with one of the smaller of its adjacent intervals;
   - if in two adjacent intervals samples of one class are strongly prevalent over samples of the other classes, the two intervals are unified;
   - if one interval is poorly populated it is unified with one of the two adjacent classes if the entropy level of the feature does not fall below a given threshold.

Given the final set of intervals, a binary representation of the values of the feature is obtained by mapping the rational value of that feature into its corresponding interval, and setting the corresponding binary variable to 1.

Supervised Discretization

In supervised discretization algorithms the class label is considered for discretizing the input data in order to effectively direct the process into a classification dependent task.

DMB Supervised Discretization

Instead of using mean and variance of the features values, when the values of a feature belonging to different classes are not overlapping, or just partially overlapping, we can use a simpler partitioning. The values are ordered and scanned in incremental way. An interval is defined as a sequence of consecutive values of the same class, once an element of a different class is found then another interval begins. Note that in this way there are not empty intervals. This alternative method performs well when it is possible to partition the values in few intervals, otherwise the reduction phase of intervals would not lead to a reduction. The worst case for this partitioning method is when the values are alternating.
Let $\mathcal{F}$ be the set of features, let $f \in \mathcal{F}$ be a feature. Let v_i be the i-th value of feature f. Every value $v_i \in f$ has a class label associated $c(f) = s$. Two intervals are merged if one of them has an amount of population lower than a given threshold. Two intervals are also merged according to the entropy values: if the entropy of the interval $K_i\ K_{i+2}$ is less than $K_i\ K_{i+1}$ then the intervals (K_i, K_{i+1}) and (K_{i+1}, K_{i+2}) are merged in one (K_i, K_{i+2}). See Figure 2.1 for the pseudo-code of the algorithm.

2.6 Noise reduction in discretization

Part of the following section was published in [FW12].

A new method based on recent experiments is presented, designed for properly dealing with cases where a certain amount of noise may be present in the data.

When intervals are defined over a continuous scale, we are interested in their purity; an interval is considered pure if all the elements that have that feature in that interval belong to the same class. The situation where all intervals for all features are pure may result in an excessive requirement: as a matter of fact, a separation among the classes can be achieved by having one pure interval for each element, or by the combination of more non-pure intervals. Given an

```
DMB supervised discretization Algorithm
1:  for every numeric feature f_i
2:    f' := orderAscending(f_i)
3:    K := ∅
4:    for every value v_j ∈ f'
5:      if v_j ≠ v_{j+1}
6:        define K_k := (v_j + v_{j+1}) / 2
7:      end if
8:    end for
9:    mergeminpop
10:   mergentro
11: end for
```

Figure 2.1: DMB supervised discretization algorithm.

interval, we then adopt as measure of its purity the normalized reciprocal of the class entropy of the elements that belong to that interval; as detailed later, we aim for a discretization where each element of the training set receives at least one interval with a good value of the purity measure.

Previous versions of the method were based on the iterative maximization of the purity of each interval; here we consider the additional problem of noisy data and its potential disturbance on the discretization process. A strict requirement on the purity of the intervals may in fact lead to the identification of too many small intervals; the problem then amounts in determining a proper balance between the number of intervals (that is to be minimized) and the purity of the intervals (that is to be maximized). Clearly, these two objectives are in contrast: the purity of the intervals cannot increase with the reduction of the number of intervals, and vice versa.

We define below a formal framework to deal with this problem.

We start by considering the projection of the data onto a single feature, say feature k, and f_{ik} for $i = 1, ..., n$ as the vector of the values that each element e_i assumes for feature k. In order to define the intervals for discretization we use cut points in the scale of the feature, e.g., values that determines the extreme of the intervals. Without loss of generality we assume that vector f_{ik} is in non-decreasing order and we mark the positions where a change in the class of two consecutive elements occurs. In these positions are the cut points that define only pure intervals. More formally, we define as nc_k the number of cut points for feature k, and with $c^k = \left\{c_1^k, ..., c_{nc_k}^k\right\}$ the values of such cut points,

where $c_j^k = 0.5 \times (f_{ik} + f_{i+1,k})$ whenever elements i and $i+1$ belong to different classes.

Now we build a measure of the discriminating power of a cut point. In order to do this for a given cut point c_j^k, we call a crossing any pair of elements $\{i_1, i_2\}$ that belong to different classes and lie on different sides of the cut point. With $NC(c_j^k)$ we indicate the set of all crossings of c_j^k. Then, for each crossing $\{i_1, i_2\}$ we define a distance $d(i_1, i_2)$ equal to the number of position that occur between i_1 and i_2 when the non decreasing order of the f_{ik} is considered. Finally, for a real parameter λ_k, we compute the discriminating power of cut point c_j^k as

$$DP(c_j^k, \lambda_k) = \sum_{(i_1,i_2) \in NC(c_j^k)} 1 \times e^{-\lambda_k \times d(i_1,i_2)}$$

$DP(c_j^k, \lambda_k)$ measures how good is a given cut point in discriminating elements of different classes. The parameter λ_k plays an important role in this measure: when $\lambda_k = 0$ each crossing has the same weight and therefore in the computation there is no account of locality: one single cut point will receive the largest value of $DP()$, and this cut point would be the best choice if we decide to choose only one cut point. Else, when λ_k becomes large, the measure becomes more local, and the value of $DP(c_j^k, \lambda_k)$ tends to be the same for all the cut points of the same feature.

Given a value λ_k, for obvious reasons we propose to choose as candidate cut points those that are associated with peaks in the values of the discriminating power. Below we discuss a sensible rule to pick the value of λ_k for all k and motivate its beneficial role in the presence of noisy data.

To do so, we now turn to consider all the m features. For each feature k it is easy to know (e.g. with binary search) the smallest value of λ_k for which all the cut points of k receive the same discriminating power. Call this value λ_k^{max}. Define now a m-dimensional vector $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ where, for each k, $0 \leq \lambda_k \leq \lambda_k^{max}$.

For a given multidimensional value of vector λ, we can compute:

- the total number of selected cut points over all the features (recall that a cut point is selected when its discriminating power is a local maximum with respect to its neighboring cut points); we refer to the total number of cut points associated with λ as $CP(\lambda)$;

- the largest purity value that is assigned to at least one interval for each element when the cut points are selected as above; we refer to this value

as $NPI(\lambda)$ (note that this value is the lower bound on the highest purity value of the intervals over all the elements of the training set).

From the values above we can compute two reference values: $\lambda^{min} = \{0, 0, \ldots, 0\}$ and $\lambda^{max} = \{\lambda_1^{max}, \lambda_2^{max}, \ldots, \lambda_m^{max}\}$ that are the two values of the k-dimensional vector λ to which correspond the extremes in the number of cut points $CP(\lambda)$ and in the lower bound on pure intervals $NPI(\lambda)$.

It is easy to show that:

$$CP(\lambda_{min}) = k; CP(\lambda_{max}) = \sum_k nc_k$$

and

$$NPI(\lambda_{min}) \geq 0; NPI(\lambda_{max}) = 1$$

Having at hand the range of variation for $CP(\lambda)$ and $NPI(\lambda)$ we can normalize their value to obtain a relative measure defined as below:

- Robustness: $R(\lambda) = \frac{(CP(\lambda^{max}) - CP(\lambda))}{(CP(\lambda^{max}) - CP(\lambda^{min}))}$;
- Precision: $P(\lambda) = \frac{(NPI(\lambda^{max}) - NPI(\lambda))}{(NPI(\lambda^{max}) - NPI(\lambda^{min}))}$.

We define the discretization quality (DQ) as the product of robustness and precision:

$$DQ(\lambda) = R(\lambda) \times P(\lambda)$$

and adopt it as a final measure of the quality of the discretization associated with a k-dimensional vector λ. We note that this measures increases monotonically both with the robustness and the precision of the discretization, and its maximization provides a very reasonable balancing of these two criteria.

The problem then amounts to solve the following optimization problem:

$$max_{(\lambda^{min} \leq \lambda \leq \lambda^{max})} DQ(\lambda).$$

The identification of a good solution for the problem above is determined by a search procedure over the k-dimensional compact sub space identified by $\lambda^{min} \leq \lambda \leq \lambda^{max}$. We choose to adopt λ^{max} (minimum robustness, maximum purity) as a starting point and gradually decrease λ component by component (see Figure 2.2):

The main step of the procedure described in Figure 2.1 is clearly step 7, where we select one of the features and decrease its λ value; this amounts to

Discretization for noise reduction	
1:	**for** every numeric feature f_i
2:	compute nc_k and λ_k^{max}
3:	**end for**
4:	compute $R(\lambda^{max})$, $P(lambda^{max})$, $DQ(\lambda^{max})$
5:	set $DQ_0 = 0, DQ_1 = DQ(\lambda^{max}), i = 1$
6:	**while** $(DQ_i > DQ_{i-1})$
7:	select k and δ, update $\lambda_k = max(0, \lambda_k - \delta))$
8:	$i = i + 1$; compute $DQ_i = DQ(\lambda)$
9:	**end while**
10:	output cut points associated with λ

Figure 2.2: Noise reduction discretization in DMB

reducing the number of cut points for feature k and thus increasing robustness and decreasing precision. We implement this step in a very straight forward fashion, adopting a greedy strategy: for each feature we select the value of δ that would imply the loss of only one cut point; then, we compute the related loss in precision implied by the loss of this cut point. The cut points that score the smallest loss in precision are candidate for the choice and we select among them at random. When this strategy is not able to improve the current solution we stop the search in a local minimum.

Once the procedure terminates, we are left with a possibly small number of cut points that can be used to discretize the features, and we can represent the discretization with a number of True-False or binary features associated with each interval. The procedure is designed in such a way that those features affected by noise would receive a less granular discretization: in a feature affected by noise the cut points will be very close to each other and will have a limited discriminating power; for this reason, they will be eliminated as their contribution to precision will be small as opposed to other cut points.

2.7 Discrete cluster analysis

A clustering procedure, called discrete cluster analysis (DCA), has been designed and integrated in DMB. The DCA is performed in the following way. Discretization is applied to the numeric features. First, a mapping into integer values of the real value of the real values is performed, according to a discretization procedure described in the previous section. Features with the same discretized profile over the samples are clustered and the resulting interval mapping is, for each feature, associated with an integer encoding. Two or more features are merged into the same cluster when their binarized expression are the same or are at a given distance for each sample. Finally, a binarized feature for each cluster is elected as its representative (clusters composed of a single features may also be present).
Beyond a grouping of the features, the clustering step performs a substantial reduction of the features to be analyzed by the next steps of the data mining process. The clustering procedure has been designed and developed in the object oriented programming language Java 6.0 using the design pattern Information Expert [BHS07].

2.8 Feature selection

In general feature selection (FS) in machine learning is defined as the identification of a small subset of relevant attributes or features in a large data set [BFL10]. It is a set of methods to identify those features that are best useful for the specific analysis task [BFFL08].
The main aim of feature selection is to identify and to remove irrelevant and redundant information from the data set. The size reduction allows the learning algorithm to work faster and more effectively. Therefore, FS is used to reduce the data to a treatable size before it can be processed by a data mining algorithm.
In order to concentrate only on important information different methods for extracting the relevant attributes of data have been studied and applied. The size of the selected subset has to be as small as possible and must retain the information that is most useful for the future data treatment. Often sophisticated data mining methods fail when they treat directly with a very large number of features, so the data has to be reduced to a small size. Feature selection is done during the preprocessing step of the knowledge discovery process. For unsupervised analysis the best features are the ones which present

the highest variability in all the data samples or in other words that are pairwise uncorrelated and able to differentiate each element from the others. In supervised analysis (e.g. classification) good features are the ones who have different values in objects that belong to different classes and have equal or similar values in the same class. Also in this case the features have to be pairwise uncorrelated (the discriminating power of a pair of feature that is strongly correlated is equivalent to that of only one feature in that pair). The principal obstacle for an effective feature selection is the dimension of the data set: the number of candidate subsets is exponential in size of the original data set. Therefore many methods adopt heuristic algorithms, which do not assure that the selected subset is the best one. Other methods use algorithms based on optimal problem formulation (minimization of a quality function), but also these need some approximation when evaluating the function.

Part of this subsection is based on [BFL10], [BFFL08], [DAFM06], [UA05] and [DL97] where the authors give a general overview of feature selection and identify the main steps of this process.

Feature selection methods

The Feature Selection process is based on four main steps [BFL10]:

1. Generation Procedure;
2. Evaluation Function;
3. Stopping Criterion;
4. Validation Procedure.

In the generation procedure the candidate subsets of features are computed. In a set with N features, the possible subsets are 2^N (e.g.: we can enumerate all the possible subsets by constructing a binary tree of size N and by counting the leaves we obtain 2^N). The goal of this step is to avoid the entire enumeration of the 2^N subsets, using heuristics or random strategies. Three approaches can be adopted: forward, backward and random strategy. The forward strategy starts with an empty set and iteratively adds a new feature according to a goodness measure. The backward strategy instead starts with all the features and removes the worst one in each iterative step. The random approach mixes forward and backward strategy and starts from a random set of features. The

evaluation function step measures the quality of a selected subset. The quality indicators can be founded on the following measures [DL97]:

- Distance (probability);
- Information (entropy);
- Dependence or correlation;
- Consistency.

The distance measure is defined as the conditional probability of one feature respect to another to discriminate between two classes: given two classes C_1 and C_2, feature X is preferred to Y if $P(C_1|X) - P(C_2|Y) > P(C_1|Y) - P(C_2|X)$. The information measure indicates the quantity of information contained in a given feature by comparing for example the entropy function obtained in the selected subsets. The dependence or correlation measure is the capability of a subset of features to predict the value of other features. The consistency measure evaluates the capacity of the selected features to separate the objects in different classes. The stopping criterion puts a constraint for avoiding the exhaustive search of the best features. This constraint can be on the number of selected features or on the improvement measure of the evaluation function. The validation procedure measures the quality of the selected subset by performing the classification on additional data (test set). Two approaches can be adopted: the filter methods and the wrapper methods (figure 2.3). In the filter approach the evaluation function is independent from the classification. In the wrapper approach each candidate subset is tested and evaluated on the basis of its performance for classification. The second approach performs better from the classification point of view, but it is clearly more time and computational resource consuming and therefore not suited for large data sets.

For the next methods only supervised learning, where a class variable/value is assigned to each object, is taken into account. The final task is to learn how to predict such class from the selected features.

Methods based on consistency

These methods search for the smallest subset of features that is as coherent as possible to the class variable. The authors in [AD91] propose the method FOCUS for Boolean data, which searches the solution space until the feature subset is such that each combination of feature values belongs to one and only one class. The main problem of this method is the explosion of the dimension of

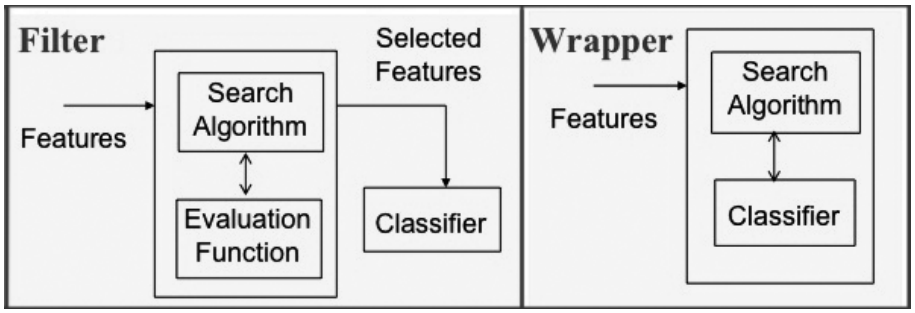


Figure 2.3: Filter and wrapper FS methods

the search space and the consequent computational time increase. For speeding up the search procedure they propose the following variant of the algorithm: given a subset S of features, the data is divided into a number of groups, each of them having the same values for the features in S. Let p_i and n_i be the number of positive and negative examples in the i-th group and N the number of individuals we have

$$E(S) = -\sum_{i=0}^{2^{|S|}-1} \frac{p_i + n_i}{N} [\frac{p_i}{p_i + n_i} log_2 \frac{p_i}{p_i + n_i} + \frac{n_i}{p_i + n_i} log_2 \frac{n_i}{p_i + n_i}]$$

that is used to evaluate all candidate features that could be added to the subset and then select the one that shows the minimum value of $E(S)$.

Another method based on consistency is described in [Nan11] where the Las Vegas Filter (LVF) algorithm is used. The inconsistency is measured with the following formula: $I(S) = \sum_{g=1}^{G_S} \frac{n_g - f_g}{n}$, where G_S is the number of different groups of objects defined by the features in S, n_g is the number of objects in group g that belong to the most frequent class, and n is the total number of objects in the training set. The algorithm is computed with these next steps:

1. The best subset B is composed of all the features and $I(B)$ is computed;
2. A random subset S of the features is chosen;
3. If $|S| \leq |B|$, then $I(S)$ is computed;
4. If $I(S) \leq I(B)$, then $B \leftarrow S$ and iterate.

This method has a good behavior in noisy data, but can perform bad in data sets composed of features with a large number of different values.

Methods based on information theory

In these methods the search of the final features is directed by a measure of the information conveyed by a subset. A typical measure of information is given by the entropy formula: $-\sum_i p_i \, log_2 p_i$. A simple idea is based on the computation of the entropy with respect to each feature and class: for each feature f and class c the entropy of the elements of f belonging to c is computed and the features with the highest entropy sum are chosen An example of these methods is in [KS97], where the authors affirm that a good subset of features must have class probability distribution as close as possible to the distribution of the original set of features: if C is the set of classes, V the set of the features, X a subset of V, $v = (v_1, \ldots, v_n)$ the values of the features in V and v_x the projection of v on X, the FS should aim to obtain a subset such that $Pr(C|X = v_x)$ is as close as possible to $Pr(C|V = v)$. The algorithm begins with all the features and applies backward elimination. At each step it removes the feature that minimize the distance between the original and the new class probability distribution. The distance is measured with the means of cross entropy defined as:
$D(Pr(C|V_i = v_i, V_j = v_j))$ where
$Pr(C|V_j = v_j)) = \sum_{c \in C} p(c|V_i = v_i, V_j = v_j) log_2 \frac{p(c|V_i = v_i, V_j = v_j)}{p(c|V_j = v_j)}$

Another method based on information theory which uses the minimum description length principle is described in [Tom09].

Methods based on correlation

These methods rely on the simple affirmation that a feature is useful if it is highly correlated with the class attribute [Hal00]. The correlation is obtained from the division of the co-variance of the two candidate variables by the product of their standard deviations σ: $corr(X, Y) = \frac{cov(X,Y)}{\sigma_x \sigma_y}$ These methods rely on the correlation among the features and to the correlation of the features with the class attribute, the author of [Hal00] demonstrated this concept, underlying that a feature is useful if it is highly correlated with the class attribute and is redundant if its value can be predicted from the values of other features. A good subset of features is composed of those features that are strongly correlated with the class attribute and very poorly correlated among themselves. An implementation of a method based on correlation is presented in [DAFM06],

where features are selected on the basis of the correlation among nominal attributes.

Combinatorial approaches to feature selection

The formulation of the feature selection problem as a mathematical optimization problem in which the cardinality of the set of the selected features has to be minimized under some constraints is proven to be effective in many real applications [BFFL08], [BFW09], [BFF10] and [ADB+11]. The DMB system uses a combinatorial approach to feature selection. In [CGK+00] the authors individuate a combinatorial problem, which can be mapped to the feature selection problem:

- Given a set S, select a subset K such that a number of properties Π_i, $i = 1, .., n$ held by S are maintained in K. The cardinality of K has to be minimized or maximized.

Other variants of the problem are:

- Subspace selection: S does not satisfy some Π_i; identify the largest subset $K \subseteq S$ such that $S_{|K}$ (S projected onto K) satisfies all Π_i;
- Dimension reduction: S satisfies all Π_i; identify the smallest subset $K \subseteq S$ such that S, $S_{|K}$ does not satisfy some Π.

With binary features we can map the problem of selecting a subset of features of minimal size that guarantees the separation between two sets can be formulated as an Integer Linear Programming Problem: the Set Covering Problem.

A formal definition of the FS problem (called test cover) presented in [GJ79] is: given a set of items 1,..,m and a collection F of features $f_1, \ldots, f_n$. The item set is divided in two classes (class A and class B). For each item h, each feature f_i takes a value in a given metric. In binary each feature has two possible values 1/0 representing the presence or the absence of a given characteristic, associated with that feature in item h (f_{ih}). A feature f_i covers (differentiates) item pair {k, h} if $f_{ik} \neq f_{ih}$. If we consider all the pairs of items {k, h} where k belongs to a class and h belongs to the other class, then a sub-collection $F' \subset F$ of features is a cover if each of such pairs {k, h} is covered by at least one element in F'. The number of pairs is equal to the product of the cardinalities of the two classes, that grows quadratically with m. The problem

of finding a subset of features of minimal size that covers all the pairs of distinct elements is called Combinatorial Feature Set or minimal test collection:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} x_i \\ s.t. \quad & \sum_{i=1}^{n} a_{ij} x_i \geqslant 1 \quad j = 1...M \\ & x_i \in 0,1 \qquad\quad i = 1...n \end{aligned} \tag{2.1}$$

where $x_i = 1$ if f_i is chosen and 0 otherwise; each of the M constraints is associated with a pair of items belonging to different classes. If row j is associated with the item pair {k, h} then we have that for feature $i\, a_{ij} = 1 \iff f_{ik} \neq f_{ih}$. The purpose of this formulation is to find a minimal set that can separate the given data. This is normally done on training data. When we apply the knowledge discovery process to test data maintaining a certain measure of redundancy of the selected features may be a good strategy to obtain better result in classification. This is done with the following formulation:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} x_i \\ s.t. \quad & \sum_{i=1}^{n} a_{ij} x_i \geqslant \alpha \quad j = 1...M \\ & x_i \in 0,1 \qquad\quad i = 1...n \end{aligned} \tag{2.2}$$

where α is an integer that indicates the degree of redundancy. This results in the selection of a larger set that brings more information in the classification step. The main drawback of these formulations proposed formulation in [BFFL08] is that the problem size grows quadratically with the objects in the data set and so they are computationally very expensive to solve.
An alternative formulation is given in [BFW09]. This formulation is more efficient (it grows linearly), but it is not guaranteed that the optimal solution individuates the best discriminating features. Given a feature f_j, define $P_A(j,k)$ and $P_B(j,k)$ be the proportion of objects where feature f_j has value k (for $k \in N$) in sets A and B, respectively. If $P_A(j,k) > P_B(j,k)$ (resp. $P_B(j,k) > P_A(j,k)$), then the presence of f_j with value k is likely to characterize objects that belong to class A (resp. B). To better qualify the strict inequality between $P_B(j,k)$ and $P_A(j,k)$, the authors introduce an additional

parameter $\lambda > 1$, and then define, for each feature j and for each object i in class A the vector d_{ij} as follows.

$$d_{ij} = \begin{cases} 1, & \text{if } f_{ij} = k \text{ and } P_A(j,k) \geq \lambda P_B(j,k); \\ 0, & \text{if } f_{ij} = k \text{ and } \lambda P_A(j,k) \leq P_B(j,k); \\ 1, & \text{if } f_{ij} \neq k \text{ and } \lambda P_A(j,k) \leq P_B(j,k); \\ 0, & \text{if } f_{ij} \neq k \text{ and } P_A(j,k) >\geq \lambda P_B(j,k); \end{cases}$$

While, for object i in class B, the value of d_{ij} will be:

$$d_{ij} = \begin{cases} 1, & \text{if } f_{ij} = k \text{ and } \lambda P_A(j,k) \leq P_B(j,k); \\ 0, & \text{if } f_{ij} = k \text{ and } P_A(j,k) \geq \lambda P_B(j,k); \\ 1, & \text{if } f_{ij} \neq k \text{ and } P_A(j,k) \geq \lambda P_B(j,k); \\ 0, & \text{if } f_{ij} \neq k \text{ and } \lambda P_A(j,k) \leq P_B(j,k); \end{cases}$$

In the practical application the parameter λ directly influences the density of the matrix composed of d_{ij} and can be adjusted to obtain a reasonable value for the density itself (say 20%). According to this definition, assume that the number of ones in vector d_j is positively correlated with the capability of feature f_j to discriminate between classes A and B. The problem is then to select a subset of the features that exhibits, as a set, a good discriminating power for all the objects considered, so that more features combined together are used to build rules that perform a complete separation between A and B. The purpose of the feature selection model is then to select a given and small number of features that, collectively, guarantee a good discriminating power for all the objects of the data sets. This can be formally stated asking to select a given number of features (say β) that maximize the minimum of the discriminating power over all the objects. The binary integer optimization problem can then be defined as follows:

$$\begin{aligned} \max \quad & \alpha \\ & \sum_{j=1}^{m} a_{ij} x_j - \alpha \geq 0 \quad i = 1 \dots n \\ & \sum_{j=1}^{m} x_j \leq \beta \\ & x_j \in \{0,1\} \qquad\qquad j = 1 \dots m \end{aligned} \tag{2.3}$$

where β is a parameter of the problem, and not a variable. The optimal solution of the above problem would then select the features that guarantee

the largest discriminating power over all the objects in the data. Despite the problem has been described with straight-forward arguments, it can easily be shown that its optimal solution amounts to identify the feature set of a given size that maximize the additive class entropy of its objects. Besides, the number of variables of the problem is given by the number of features (m), and the number of rows by the number of objects (n), keeping the size of the problem in a linear relation with the size of the data. The problem is anyway difficult to solve, and for large sizes approximate solution methods may be needed if one is not to resort to heavy and often expensive commercial solvers for integer programming. In supervised cases (classifications) a constraint is associated to each pair of objects that belong to different classes, in unsupervised cases a constraint is generated for each pair of objects.
All these formulations have been implemented in the DMB system.
The principal disadvantage of the combinatorial approach is the rapid growth of the problem dimensions due to the constraints associated to each pair of objects (belonging to different classes). Moreover, the problem is Non Polynomial (NP), it is intractable with optimum algorithms and has to be solved using heuristics.
For the resolution of the above formulation the authors of [BFFL08] propose an heuristic and non-deterministic algorithm named GRASP (Greedy Randomized Adaptive Search Procedure). The GRASP heuristic algorithm that solves this problem is described with its extensions in the next subsection.

GRASP

To solve the feature selection problem a Greedy Randomized Adaptive Search Procedure (GRASP) is proposed by Feo and Resende [FR89, FR95]. The GRASP procedure has been redesigned and improved in this dissertation. GRASP is a multi-start or iterative method, in which each iteration consists of two phases: construction of a solution and local search.

The construction phase builds a solution x. If x is not feasible, a repair procedure is invoked to obtain feasibility. Once a feasible solution x is obtained, its neighborhood is explored by the local search until a local minimum is found. The best overall local optimum solution is kept as the result. An extensive survey of the literature is given in Festa et al. [FR09a, FR09b]. The pseudo-code in Figure 2.4 illustrates the main blocks of a GRASP procedure for minimization, in which `MaxIterations` iterations are performed and `Seed` is used as the initial seed for the pseudo-random number generator.

```
algorithm GRASP(f(·), g(·), MaxIterations, Seed)
1    x_best:=∅; f(x_best):=+∞;
2    for k = 1, 2, ...,MaxIterations→
3       x:= ConstructGreedyRandomizedSolution( Seed, g(·));
4        if (x not feasible) then
5           x:= repair(x);
6        endif
7       x:= LocalSearch(x, f(·));
8        if (f(x) < f(x_best)) then
9           x_best:=x;
10       endif
11    endfor;
12    return(x_best);
end GRASP
```

Figure 2.4: Pseudo-code of a basic GRASP for a minimization problem.

Starting from an empty solution, a complete solution is iteratively constructed in the construction phase, one element at a time (see Figure 2.5). At each construction iteration, the choice of the next element to be added is determined by ordering all candidate elements (i.e. those that can be added to the solution) in a candidate list C with respect to a greedy function $g : C \to \mathbb{R}$ that measures the benefit of selecting each element. The heuristic is adaptive because the benefits associated with every element are updated at each iteration of the construction phase to reflect the changes brought on by the selection of the previous element. The probabilistic component of a GRASP is characterized by randomly choosing one of the best candidates in the list, but not necessarily the top candidate. The list of best candidates is called the *restricted candidate list* (RCL). This choice technique allows for different solutions to be obtained at each GRASP iteration, but does not necessarily compromise the power of the adaptive greedy component of the method.
In the following, we say that a column *covers* a row of a binary matrix when that row exhibits a 1 in correspondence of that column. Since the selection involves candidate columns, it is intuitive to relate the greedy function to the number of rows still to be fully covered that a column not yet chosen would cover, if selected.

As it is the case for many deterministic methods, the solutions generated by a GRASP construction are not guaranteed to be locally optimal with re-

```
procedure ConstructGreedyRandomizedSolution( Seed, g(·))
1    x:=∅;
2    Sort the candidate elements i according to their incremental costs
g(i);
3    while (x is not a complete solution)→
4        RCL:= MakeRCL();
5        v:= SelectIndex(RCL, Seed);
6        x := x ∪ {v};
7        Resort remaining candidate elements j according to their
         incremental costs g(j);
8    endwhile;
9   return(x);
end ConstructGreedyRandomizedSolution;
```

Figure 2.5: Basic GRASP construction phase pseudo-code.

```
procedure LocalSearch(x, f(·))
1    Let N(x) be the neighborhood of x;
2    H:={y ∈ N(x) | f(y) < f(x)};
3    while (|H| > 0)→
4        x:=Select(H);
5        H:={y ∈ N(x) | f(y) < f(x)};
6    endwhile
7    return(x);
end LocalSearch
```

Figure 2.6: Pseudo-code of a generic local search procedure.

spect to simple neighborhood definitions. Hence, it is almost always beneficial to apply a local search to attempt to improve each constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution in the neighborhood of the current solution. It terminates when no better solution is found in the neighborhood. The *neighborhood structure* N for a problem relates a solution s of the problem to a subset of solutions $N(s)$. A solution s is said to be *locally optimal* if in $N(s)$ there is no better solution in terms of objective function value. The key to success for a local search algorithm consists of the suitable choice of a neigh-

borhood structure, efficient neighborhood search techniques, and the starting solution. Figure 2.6 illustrates the pseudo-code of a generic local search procedure for a minimization problem.
A local search has been designed that, starting from a just built solution, tries to find a better quality solution, i.e. a new set of columns with lower cardinality (removal of redundant columns) and/or corresponding to a higher coverage.
It is difficult to formally analyze the quality of solution values found by using the GRASP methodology. However, there is an intuitive justification that views GRASP as a repetitive sampling technique. Each GRASP iteration produces a sample solution from an unknown distribution of all obtainable results. The mean and variance of the distribution are functions of the restrictive nature of the candidate list. For example, if the cardinality of the restricted candidate list is limited to one, then only one solution will be produced and the variance of the distribution will be zero. Given an effective greedy function, the mean solution value in this case should be good, but probably sub-optimal. If a less restrictive cardinality limit is imposed, many different solutions will be produced implying a larger variance. Since the greedy function is more compromised in this case, the mean solution value should degrade. Intuitively, however, by order statistics and the fact that the samples are randomly produced, the best value found should outperform the mean value. Indeed, often the best solutions sampled are optimal.
An especially appealing characteristic of GRASP is the ease with which it can be implemented. Few parameters need to be set and tuned, and therefore development can focus on implementing efficient data structures to assure quick GRASP iterations.

2.9 Noise reduction in feature selection

Part of the following section was published in [FW12].

When dealing with binary features, the problem of selecting a subset of features of minimal size that guarantees the separation between two sets can be formulated as an Integer Linear Programming Problem, more specifically as a variant of the Set Covering problem. A mathematical formulation of the problem is given in 2.1 and explained in the previous section.

In [BLL+02] a branch-and-bound procedure based on a new definition of branching rules and lower bounds for the above problem is presented. Nevertheless, when the problem size is large, the use of optimization algorithms

to produce guaranteed optimal solutions becomes impractical, and one has to resort to heuristics schemes.

The above approach to the FS problem presents also additional drawbacks. Its purpose is to find a minimal set that can separate the given data, that plays the role of training data in the general process. The features associated with the minimal set are then used to project the training data and to derive classification rules, that are then applied to test data, once the latter has been projected using the same feature set.

When the data is noisy or not well sampled, it may happen that the effect of good features - e.g., features that should belong to the final model - is masked by the noise in some data and the thrifty representation adopted with the previous model does not leave space to fix this error; for this reason, maintaining a certain measure of redundancy in the information that is retained by the FS selection results into good strategy for the production of rules with good predictive power in the presence of noisy data.

This is particularly true when FS is followed by classification algorithms that can perform an additional selection of the features based on the separating model. For this reason, we want to consider cases where the right hand side of the covering constraints is greater than 1. This would result in the selection of a larger set that brings more information to the formula extraction step; more precisely, we propose to adopt an optimization model where the number of features to be selected is fixed in advance by the parameter β, and the level of redundancy α is maximized in the objective function in 2.3.

To solve the generalized set covering problems we pursue a non-deterministic and heuristic method known in literature as GRASP, described in the previous section.

The construction and the local search phases of the GRASP heuristic are applied repeatedly, finally returning the best solution found.

The proposed feature selection method is designed to achieve robust results at the price of a certain level of redundancy, in order to absorb the potential bias on the results induced by noisy data. The quality of the results would although depend on the choice of the parameter β. An additional refinement of the method takes into account a strategy for the right choice of β, based on a more extensive analysis of the relations between the values of α and β.

We first note that the value of the solution α cannot decrease when the parameter β is increased; therefore, we solve a initial instance of the problem with a sufficiently large value of β_0, obtaining a corresponding solution $\alpha_0 > 0$. Then, we reduce β gradually and obtain corresponding α values, until we reach the smallest value β_h for which $\alpha = 1$; from these runs we derive the values of

β needed to obtain $\alpha = 0, 1, 2, \ldots, \alpha_0$. We can now inspect these values and measure the increase in β that is needed to obtain an increase of 1 in the value of α. For ease of reference, call β_h the value of β needed to obtain $\alpha = h$, and call $\hat{\beta}_h = \beta_h - \beta_{h-1}$; then, when $\hat{\beta}_{h+1} >> \hat{\beta}_h$ we stick to the value β_h, and adopt the corresponding solution of value h to identify the features to be selected and passed on to the next step.

2.10 Logic formulas extraction and classification

The feature selection step provides the characterizing attributes for each class in the training set. In the formula extraction step DMB has to extract the logic separating formulas of each class. The aim of this step is to produce logic formulas or logic rules for each class.
After the output of the candidate features from the GRASP algorithm DMB adopts a MinSat approach [FT02] for learning the logic classification formulas. The Lsquare [FT06, Tru04] method is part of DMB for computing the model of the analyzed data set, e.g. the separating "if-then" formulas or rules. Lsquare approaches this challenge as a sequence of Minimum Cost Satisfiability Problems (MinSat), a well studied combinatorial optimization problem that is NP-Hard. Therefore it solves the problem with an algorithm based on decomposition techniques. The reader may refer to [FT02, FT06, Tru04] for a detailed description of the method and problems.
The logic formulas extracted by this method are in Disjunctive Normal Form (DNF) over the literals, i.e. Lsquare computes for every class of the experimental samples the logic classification formulas in Disjunctive Normal Form. The literals of the formula represent inequalities in the form of, e.g. "A01232423 $\geq$ 0.5", and are conjucted in "AND" and "OR" clauses.
DMB is able to fine tune the formula extraction computation by increasing the cost of the negative literals in the MinSat problem formulations in order to output a majority positive literals, this feature improves the clarity of the model and simplifies the comprehension of the logic formula associated to each class. Positive literals are also more specialized and have therefore a better predictive power. In case of negative literals, DMB also integrates an automatic transformation procedure of the logic formula, which outputs the propositional rules in positive form, using a conjunctive normal form. This is done for simplifying the user experience, who can deal with more readable formulas.
The above step produces logic formulas or logic rules for each class. At this point, an additional evaluation step is performed on the training set to assign

proper weights to them. Each logic formula is applied to each object of the training set and:

- if the formula recognizes the object and it is associated to the same class, the number of correct classified elements (true positives TP) of the formula is increased;
- if the formula wrongly recognizes the object and it is associated to a different class, the number of wrong classified elements (false positives FP) of the formula is increased;
- if the formula does not recognize the object but it is associated to the same class, then the number of not recognized elements (false negatives FN) of the formula is increased;
- if the formula does not recognize the object and it is associated to a different species, then the number of true negatives (TN) of the formula is increased.

These quantities are then normalized and the true positive rate (%TP), false positive rate (%FP) and true negative rate (%TN) are computed; the Laplace score ($LapS = \frac{TP+1}{TP+FP+m}$) and the false positive / true positive rate ($\frac{FP}{TP}$) are also considered.

The classification of the test data is done in the following way:

- if an element is recognized by only one formula then it is classified in the species associated to that formula;
- if an element is recognized by two or more formulas, then the formula with an higher Laplace Score is chosen;
- if the Laplace Scores are equal, we consider the false positive value and select the formula with the lower false positive value;
- if also the FP value is equal then the element is not classified.

Additional cut offs of formulas with sub-optimal coverage are done by considering the false positive / true positive rate (FP/TP) and the true positive rate. If FP/TP is greater than a given threshold or the TP is 0 the formula is considered unreliable and therefore discarded.

2.11 Noise reduction in the extraction of logic formulas

Part of the following section was published in [FW12].

In this section a variant of the formula extraction step is described, which is designed to refine the solution in order to spot the presence of noise in the data and to mitigate its potentially negative effects.

The identified DNF formulas have the property of being created by conjunctive clauses that are searched for in order of coverage of the training set. Therefore, they usually are formed by few clauses that explain a large portion of the data (clauses with large coverage, that synthesize the interpretation of the trends present in the data) and several clauses that explains few of the examples in the training set (smaller coverage: the interpretation of the outliers).

At this stage of the process we may assume that part of the noise may have been eliminated in the previous steps; nevertheless, being the method exposed to over-fitting by its very nature, we adopt an iterative pruning approach that refines the formulas getting rid of variables that appear in the model but do not show a relevant contribution.

We recall that an explanatory model determined by *Lsquare* is composed of a logic formula in DNF, referred to as F, designed to have value *True* on one of the classes (say, class A) and value *False* on the other one (class B; the role of the classes could be inverted if needed). Such a formula is composed of a number of conjunctive clauses, combined with disjunctions. Each clause, in turn, is composed of a finite number of literals, e.g., occurrence of a logic variable or its negation. We indicate a logic variable that appears in F as v_i.

Once F has been determined, we define a pruning procedure that removes a variable at a time according to the rules described below, until a certain stopping criterion is met.

First we need to define, for every logic variable v_i that is present in F, three measures:

- $ex(v_i)$, or *exclusive coverage* of v_i: the number of couples of training elements belonging to different classes that are differentiated only by v_i in F;
- $sc(v_i)$, or *score* of v_i: we eliminate the variable v_i from the formula F and then evaluate the reduced formula on training data. Such evaluation will result in number of errors (false negatives $fn(v_i)$ and false positives $fp(v_i)$). The score $sc(v_i)$ is exactly the number of such errors;

- $ds(v_i)$, or *decreasing speed* of v_i: it is defined as the difference between the score obtained by v_i at the previous iteration and the one obtained at the current one.

These measures are the main ingredients of the proposed pruning procedure. When $ex(v_i)$ is sufficiently large we know that the v_i cannot be substituted by another one, and we are not inclined to prune it. The score measure $sc(v_i)$ tells with higher precision what would happen if we remove this variable.

At each iteration, we select the variable with minimum score, breaking ties arbitrarily; assume v_j to be such variable.

Then, we consider its exclusive coverage $ex(v_j)$; if $ex(v_j) \leq 0.05 * n_A * n_B$ (with n_A and n_B we indicate the number of elements in classes A and B, respectively) we allow to prune v_j knowing that at most 5% of the couples of elements of different class may not be correctly distinguished when a model without v_j is calculated by *Lsquare*.

A special role is assigned to the the decreasing speed $ds(v_i)$. Features whose score tends to decreases with the pruning iterations are in fact less important that those for which the score increases; as fewer features are left at each iteration, we know that the contribution of good features will increase, and vice versa. The decreasing speed captures this behavior: when high, it indicates that a feature is eligible for pruning; when small, it indicates that the feature is to be kept. We use the $ds(v_i)$ to determine a stopping criterion: if no features have $ds() > 0$ we interrupt the pruning.

The procedure is synthesized in the scheme of figure 2.7.

2.12 Potential limitations of DMB

DMB is a new classification workflow framework dedicated to the analysis of life science data. Its strength are the customization potentiality to a wide number of biological problem and the strong stress on the computation of a human interpretable model.
The potential limitations of DMB may be individuated in the following:

- the formula extraction problem (MinSat) is limited to a certain number of logic variables in Lsquare;
- an optimal solution of the MinSat problem is not provided in Lsquare, i.e. no optimal logic classification formulas;
- the non output of alternative classification models;

Pruning the Logic Model	
0:	**while**
1:	**for** all logic variables v_i compute $ex(v_i)$, $sc(v_i)$, $ds(v_i)$
2:	**end for**
3:	select v_j such that $sc(v_j)$ is minimum;
4:	$if(ex(v_j) \leq 0.05 * n_A * n_B)$
5:	AND (there exists v_k such that $ds(v_k) > 0$)
6:	remove v_j from F
7:	run *Lsquare*;
8:	*else* break
9:	**end while**

Figure 2.7: Pruning procedure

- the missing integration of other discretization, feature selection and formula extraction algorithms;
- the single thread architecture.

To address these issues a novel algorithm for the formula extraction problem is planned which provides both approximate and optimal solutions. It is also a goal in the near future to develop new methods for computing alternative classification models when performing the data mining analysis, so that not just a single classification model is provided. This leads to a wider description of the data and is very important in biology, when scientist are interested in knowing all the differentiating variables of the analyzed classes. Finally, the integration of existing alternative algorithms for discretization and feature selection are considered and performances of the methods are going to be improved with parallelism.

2.13 Releases

All the different steps of the data mining process of DMB have been engineered in different modules written in ANSI C and compilable on Windows, Linux and MacOs operating systems. DMB relies on the software architectural pattern

Pipes and Filters [BHS07] for computing streams of data. The DMB software system is organized in flows. The modules can be combined for the specific analysis that has to be performed. Three flows are currently available:

- Barcoding with Logic Formulas (BLOG);
- MicroArray Analyzer (MALA);
- Data Mining In Big (DMIB).

Diverse versions have been released:

Barcoding with Logic Formulas - BLOG

BLOG is dedicated to DNA Barcode sequences classification. For further details the reader may refer to section 3.2.

Microarray Logic Analyzer - MALA

MALA is specifically designed for Microarray data analysis. For additional details the reader may refer to section 3.3.

Data Mining in Big - DMiB

DMiB is a general flow, which can be configured for various types of analysis.

The different releases and applications to biological data analysis are described in chapter 3 of the dissertation.

2.14 Conclusions

In this chapter the logic data mining system DMB was presented, its approach to the knowledge discovery process was described in detail, and its computational steps were illustrated: sampling, discretization, clustering, feature selection, logic formulas extraction, noise reduction and classification.
DMB supports two types of sampling: percentage split and cross validation.
The discretization procedures implemented by DMB are supervised and unsupervised.
A novel attribute clustering method based on discretization has been presented in section 2.7.

DMB approaches the feature selection problem with a combinatorial approach, called minimum test collection problem and solves it with a greedy heuristic named GRASP.
For the logic formulas extraction DMB adopts a MinSat approach [FT02] and solves it with the Lsquare heuristic.
All the steps have been integrated with a noise reduction procedure.
The classification is performed according to a weighting of the logic formulas with statistical rates.
Finally the potential limitation of DMB were delineated and the different software releases were listed.

Chapter 3

Applications to biological data analysis

3.1 Introduction

In this chapter the experimental results obtained by applying the logic data mining system DMB to a number of problems arising in biological contexts and the different software releases resulted from these applications to different life science problems are described.
The first application of the DMB system was for performing species classification with DNA Barcode sequences. A special customization of the system was necessary and resulted in the software BLOG described in the next section. It has been showed that BLOG is the best performing method currently available for classifying species with DNA Barcode sequences [vVWFB12].
DMB was also customized for gene expression profiles analysis resulting in the MALA software release, a complete description is available in section 3.3 and in [WFB12]. Gene expression data is numeric (real) and characterized by a large number of features. MALA strengths are the clustering and feature selection capabilities, which lead to compact models in terms of logic formulas. MALA was applied to an important Alzheimer Disease mice experiment [ADB+11] resulting in the identification of a subset of genes able to distinguish between the healthy and diseased samples, these genes are clear biomarkers of the Alzheimer disease. MALA was also compared with other standard classification methods, as Support Vector Machines (SVM) and trees, for performing classification on public available gene expression profile data [WFB12].

Another application of DMB was the classification of the actually known five different human polyomaviruses, described in section 3.2 and in [WLPD+12]. DMB was used to analyze a large set of clinical patient trial data resulting in the identification of a subset of characterizing features for every class of disease (section 3.4).
Finally, DMB was also applied to the analysis of Single Nucleotide Polymorphism (SNP) simulated noisy patient data to validate the noise reduction procedure described in chapter 2 and to compare it to other rule based classification methods (section 3.5). DMB was able to compete with these methods and resulted very promising.

3.2 DNA Barcodes species classification: A comparative analysis

Part of the following section was published in [WvVFB13, vVWFB12, BFW09].

BLOG (Barcoding with LOGic) is a logic data mining diagnostic and character-based DNA Barcode analysis method. Its aim is to classify specimens to species based on DNA Barcode sequences with a supervised machine learning approach.

BLOG 2.0 software design, fundamental modules, online/offline user interfaces and the major recent improvements are described. These improvements affect both methodology and software design, and lead to higher accuracy and recognition rates over a large test bed of empirical and simulated data sets. On average BLOG 2.0 outperforms previous versions as well as other DNA Barcode analysis methods. Finally the power of BLOG 2.0 which consists in its output is pointed out: classification rules that compactly characterize species in terms of DNA Barcode and can be applied outside the scope of DNA Barcoding.

Introduction and background

A new specimen classification technique named DNA Barcoding was proposed in 2003 by Hebert et al. [HCBd03]. A short DNA sequence from a small portion of the mitochondrial DNA, the gene cytochrome c oxidase subunit I (COI), was chosen as Barcode for animals and, more recently a combination of two different gene regions (rbcL and matK) was defined as Barcode for plants (CBOL Plant Working Group 2009), and the internal transcribed spacer (ITS) gene

region was proposed as a universal Barcode marker for fungi [SSH+12]. These small portions of the DNA, most of them specifically chosen because of their easiness to align, present high variability, also between closely related species, and are considered to contain the needed information to classify a specimen to species.

Several data analysis methods have been developed and adopted to automatically classify a DNA Barcode sequence to a predefined species [DKMS05, AG07, NM06, MBWN08, SPD08, BFW09, Lit11]. The classification problem may be formulated in the following way: given a reference library composed of DNA Barcode specimen sequences of known species, an unknown DNA Barcode sequence has to be recognized in a species of the library. Various DNA Barcode data analysis methods that solve this problem have been proposed: tree-based methods [SN87, MBWN08, EC63, HR01]; similarity-based methods [FAPB02, ADS+09, MSVP06, AMS+97]; statistical methods [NM06, MN05, AG07] and diagnostic methods [SPD08, DKMS05, BFW09]. Tree-based methods assign unidentified (query) Barcodes to species based on their membership of clusters (or clades) in a DNA Barcode tree. Similarity-based methods assign query Barcodes to species based on how much DNA Barcode characters they have in common. Statistical methods estimate confidence measures on DNA Barcode matches for species identification. Diagnostic methods (character-based methods) rely on the presence/absence of particular characters in DNA Barcode sequences for identification, instead of using them all. A collection of online methods is available on `bol.uvm.edu` [ST11] and on `www.boldsystems.org` [RH07].

In this section version 2.0 of the character-based diagnostic DNA Barcode analysis method BLOG, which is an evolution of the logic data mining method described in [BFW09], is presented. BLOG identifies for each species in the reference library the distinctive nucleotide positions of the DNA Barcode sequences, and assigns to each species logic classification formulas, small rules in the form of "if-then", that are able to characterize a species in a compact way. A distinctive advantage of BLOG compared to other available methods is that such logic formulas offer additional species-level information that can be used outside the scope of DNA Barcoding, e.g. in species description and in molecular detection [vVWFB12]. The main evolutions of BLOG 2.0 reside in the availability of enhanced user interfaces, in a new classification algorithm, in software engineering, in its output and in an optimized criteria for the selection of the candidate distinctive nucleotide positions.

Methods: BLOG - Barcoding with LOGic formulas

BLOG is a DNA Barcode analysis software, with the aim to classify a set of given specimens to species. The two main computational steps of BLOG are:

1. Feature selection: BLOG selects positions of the DNA Barcode sequences that are potentially suited to distinguish species;

2. Formula extraction: BLOG computes the logic formulas that classify each species present in the reference library.

Input and output

BLOG uses a supervised machine learning approach: the user has to provide as input a training set containing specimens with a priori known species membership. Based on this training set the software selects suitable nucleotide positions (feature selection) and computes the logic formulas for species classification (formula extraction). Subsequently, the logic formulas can be applied to a testing set which contains specimens that require classification. Such a testing set can contain query specimens with unknown species membership or, alternatively, specimens with a priori known species membership, allowing verification of the specimen classifications. Input files are DNA Barcode sequence alignments in standard FASTA format [Pea90]. The heading line of each sequence is composed by the starting character ">", the "specimen id" and the "species name field" separated by a vertical bar "|" (e.g.: ">E3434243 | squalus edmundsi"). In general, users will provide training and testing sets separately. It is also possible to provide only one data set, which is automatically randomly divided over a training and testing data. The ratio of the number of specimens in the training and testing data set can be specified. Output of BLOG are logic formulas for species classification, classification rates, and confusion matrices. The logic formulas consist of an antecedent and a consequent, with the antecedents composed of conjunctions and disjunctions of nucleotide assignments to a specific position in the DNA Barcode sequence, while the consequents are the different species names that are present in the data set. An example of such a logic classification formula is "if pos40=T and pos265=T then the specimen is classified as Ompok bimaculatus". Classification rates are given as number and percentage of correct, incorrect and not classified specimens in the training set and in the test set (if a priori known species memberships are available). Confusion matrices give in detail statistics for each species that is present in the data sets, with each element e_{ij} of this matrix representing

the number of specimen from species i predicted to be of species j. Correctly classified elements are on the main diagonal of the confusion matrix.

Feature selection

The first main computational step of BLOG is the extraction of species-specific positions of the DNA Barcode sequences from the training set. In general, feature selection in machine learning is defined as the selection of relevant attributes or features of the data samples [BFL10]. The feature selection approach of BLOG is based on the mathematical optimization formulation "minimal test collection" described in chapter 2 section 2.7. This mathematical formulation is hard to solve with optimal values, because a constraint is defined for each pair of specimens belonging to diverse species, increasing its size quadratically. It is therefore computationally demanding to solve with exact algorithms for a significant dimension of the reference library. To circumvent quadratic growth of the mathematical formulation a linear approximation for the problem is proposed by [BFW09]. This approximation keeps the problem size low and solvable without substantially affecting quality of the results [BFW09]. In addition, BLOG implements an effective heuristic algorithm to solve the feature selection problem while at the same time limit the computation time and keep the problem tractable (see section 2.7 for further details). This algorithm is based on the Greedy Randomized Adaptive Search Procedure (GRASP) proposed by Feo and Resende [FR95], a non-deterministic randomized multistart iterative meta-heuristic algorithm that does not provide a proof of the optimality of the solution, but has proven efficient and effective in many applications such as DNA Barcoding [BFW09, vVWFB12] and microarray analysis [ADB+11]. Previous versions of BLOG [BFW09] applied the feature selection step simultaneously on all species in the reference database. However, features that allow separation of one species are not necessarily useful for separating another. BLOG 2.0 therefore has the option to apply the feature selection step separately to each species in the reference library. In each feature selection step the considered species is assigned class A and all the other species class B. Consequently, m different instances of the feature selection problem have to be solved for each analysis run, where m is the number of species in the training set. A large computation time would be needed with optimal algorithms which further justifies the use of the GRASP heuristic described in chapter 2 section 2.7.

Formula extraction

In the formula extraction step BLOG extracts the formulas separating each species, i.e. the aim of this step is to produce a logic formula (or logic rule) for each species. For extracting the logic formulas BLOG adopts the method Lsquare described in chapter 2 section 2.8. Each formula is represented as a disjunction of conjunctive clauses. Each literal of a formula represents an assignment of a nucleotide (i.e. A,T,G, or C) to a particular position in the DNA Barcode sequence. Previous versions commonly produced negative literals (e.g. pos40=NOT T) to ensure shorter formulas. However, negative literals recognize three different nucleotides making them potentially less precise than positive literals (e.g. pos40=G OR pos40=C would be a more precise formula than pos40= NOT T). Therefore, BLOG 2.0 allows increasing the cost of the negative literals in the formula extraction problem formulations in order to prevalently output positive literals.

Classification

Before evaluating the test set, BLOG 2.0 performs an evaluation of the training set with the aim to assign relative weights to the logic formulas: the Laplace Score [TSK05], the false positive and true positive rates are computed for every logic formula over the reference library, these scores are then considered in the test set for performing the classification assignments. The reader may refer to chapter 2 section 2.10 for further details.

Workflow and modules

The flow chart of BLOG 2.0 is represented in figure 3.1, pointing out all the different software modules. A complete description of the main modules is reported in chapter 2; differences in BLOG 2.0 flow are the for each species iterated features selection and the reengineered classification components using the procedures described above (Feature selection and Classification sections). BLOG uses the Pipes and Filters software architectural pattern for computing streams of data [BHS07] and it is written in ANSI C. Three releases of BLOG 2.0 are available:

Graphical user interface

An offline graphical user interface release is available for download on `dmb.iasi.cnr.it/blog-downloads.php`. This release of BLOG 2.0 is in-

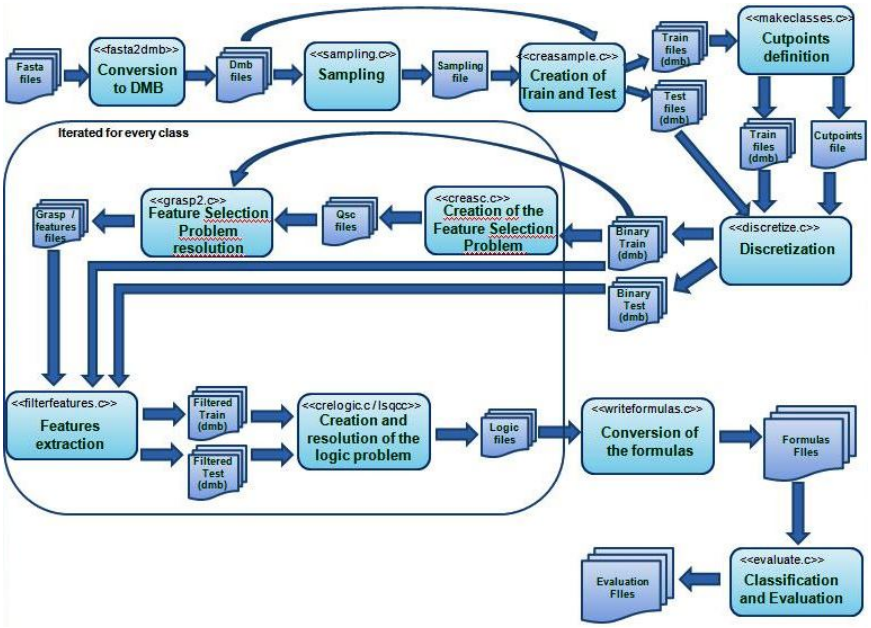


Figure 3.1: BLOG flow diagram

tended for most users, who wish to fine tune the analysis and run the software on their own computers (Linux and Windows) as it has the most user-friendly interface. Users can graphically view the DNA Barcode sequences, load training and test files, execute BLOG 2.0 and view the classification results and the logic formulas for each species present in the data set. The offline graphic user interface has been implemented with the Java Swing framework. A complete user manual for this version is provided on the website in the BLOG-2-GUI-manual.pdf. A screenshot of the offline graphic user interface is available in figure 3.2.

Command-line interface

For performing intensive experimentations we suggest to use the offline command-line version, which is available for download at
`dmb.iasi.cnr.it/blog-downloads.php`. With this version the user can organize experiments in batches and read the output in different files for each

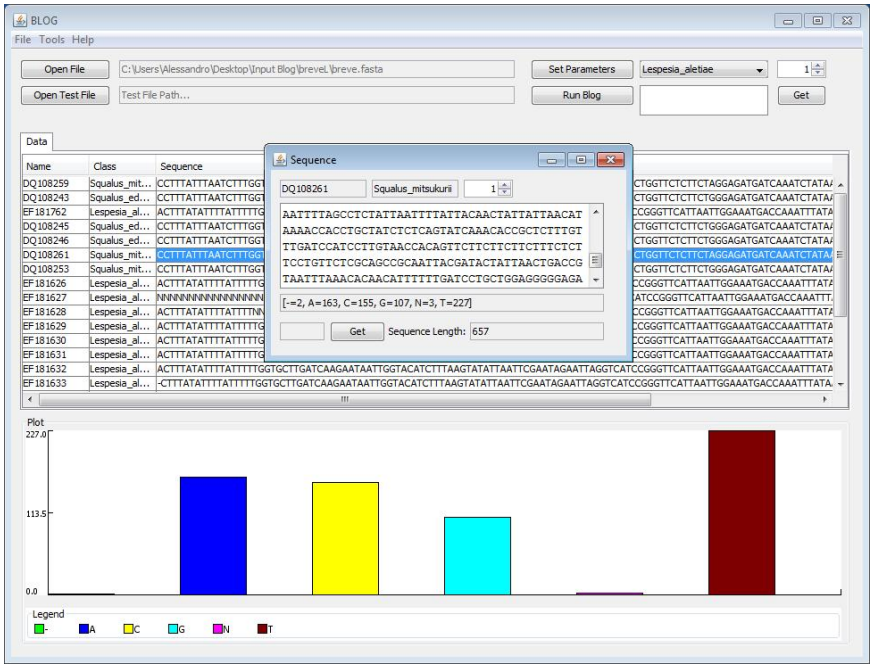


Figure 3.2: BLOG graphical user interface

run. Executables of the BLOG software are available for Linux and Windows and the C source code is released for compilation on other operating systems. A complete user manual for this version is provided on the website in the BLOG2-COMMAND-LINE-README.txt.

Web release

A simple web user interface of BLOG is available at `dmb.iasi.cnr.it/blog.php`. Data (training and test sets) can be uploaded through an input form and results the (classification rates, logic formulas and confusion matrices) are returned in CSV (Comma Separated Values) text files, which are easily readable by all common spreadsheet software. In addition, a compressed archive containing all results is sent to the user via email. We direct the users to `dmb.iasi.cnr.it/blog.php` for additional information and usage instruction for this release. The BLOG web service has been released

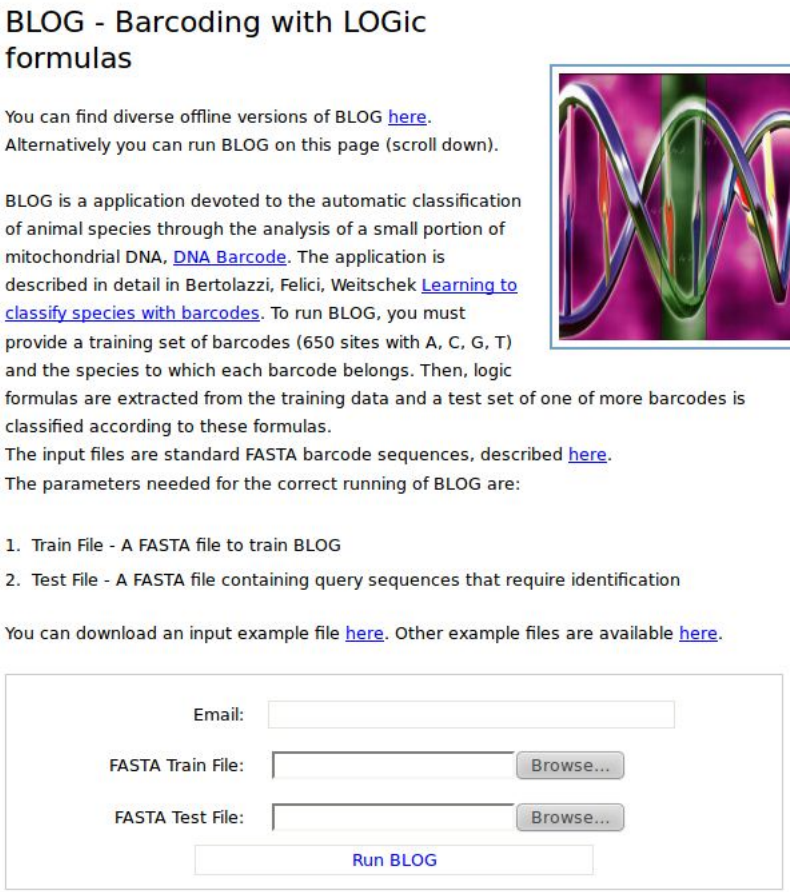


Figure 3.3: BLOG webinterface

on a Linux server (Ubuntu Server distribution), using a LAMP (Linux Apache MySql and PHP) platform (Linux Kernel 2.6.32, Apache 2.2.14, PHP 5.2) with a Java job queuing system that relies on a MySQL database (version 5.1.41). In figure 3.3 a screenshot of the BLOG web page is provided.

		Birds				Bats				Fishes			
		BLOG 1		BLOG 2		BLOG 1		BLOG 2		BLOG 1		BLOG 2	
# of chosen	Test %	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing	Training	Testing
features		% error	% error	% error	% error	% error	% error	% error	% error	% error	% error	% error	% error
10	10	9.08	18.5	0.23	14.47	15.93	15.45	0	3.18	7.58	12.17	0.34	6.04
10	20	12.4	20.03	0.4	15.3	20.04	21.67	0	3.08	10.37	13.09	0.34	6.32
15	10	1.4	11.75	0	12.65	5.84	7.47	0	2.96	4.78	8.06	0.25	5.2
15	20	1.75	11.52	0	13.62	9.03	10.42	0	3.12	6.83	13.74	0.21	5.26
20	10	0.8	9.05	0	12.13	1.11	3.23	0	3.05	2.56	7.83	0.23	5.2
20	20	0.47	8.94	0	12.9	1.77	5.3	0	3.13	9.76	15.31	0.23	5.2
25	10	0.3	7.73	0	11.3	1.56	3.18	0	3.01	0.86	5.57	0.23	5.2
25	20	0.24	7.46	0	11.59	1.83	3.05	0	2.85	1.49	6.03	0.23	5.2
avg		**3.3**	**11.8**	**0.08**	**13**	**7.13**	**8.72**	**0**	**3.05**	**5.53**	**10.22**	**0.26**	**5.45**
std		4.7	4.8	0.15	1.39	7.3	6.78	0	0.11	3.68	3.78	0.05	0.46

Figure 3.4: Average classification performance of BLOG 2.0 versus BLOG 1.0

Results and Discussion

The main purpose of this section is to describe the software tool BLOG 2.0, its methodological background, some guidelines on its use, its interfaces and to compare it to other classification methods for DNA Barcode sequences. The system has been experimentally tested on various datasets and accurately compared with other competing methods [WVF11] and in [vVWFB12]. For completeness a brief revision of the experimental results is reported, pointing the reader to the referred papers for additional details. The methodological changes implemented in BLOG 2.0 constitute major improvements compared to previous versions. Indeed BLOG 2.0 outperformed BLOG 1.0 based on three empirical DNA Barcode data sets. Data sets comprised DNA Barcodes of bats (Chiroptera, 840 specimens from 82 species), fishes (Craniata, 626 specimens from 82 species) and birds (1623 specimens from 150 species) and were made available by the Consortium for the Barcode of Life
(accessible on `dmb.iasi.cnr.it/blog-downloads.php`). Within each data set specimens were randomly distributed over a train and a test data set with the percentage test sequences being from 10% to 20% and using four different settings of BLOG for the maximum number of features to select (10, 15, 20, and 25), amounting to a total of eight different analyses per data set. A synthesis of the results from [BFW09] and [WVF11] is given in figure 3.4 , showing that BLOG 2.0 has better classification results overall. Where the logic formulas as calculated by the previous version had considerable error rates even on the training sets (3.37.1%) those calculated by BLOG 2.0 fit the training sets nearly

perfectly (average error rates < 0.23%). Average error rates on the testing sets decreased substantially based on the data from bats (from 8.72% to 3.05%) and fishes (from 10.22% to 5.45%). The only increase in error rates was for the testing set based on the Birds data (from 11.8% to 13.0%). In addition, most literals of the logic classification formulas are outputted in positive form, making them more precise. As an example of BLOG output, we provide the formulas produced for the bats data sets in table 3.1.

Species	*Formula*	*Coverage*
Ametrida centurio	182=G AND 215=C AND 554=A	1.00
Anoura caudifer	320=A AND 539=G	1.00
Anoura geoffroyi	215=G AND 320=G AND 542=A	1.00
Anoura latidens	56=C AND 215=A AND 554=A	1.00

Table 3.1: Examples of logic formulas

To additionally show the efficacy of BLOG 2.0 a comparison of the relative performance of DNA Barcode data analysis methods in identifying recently diverged species was performed. Two tree-based methods Neighbour Joining (NJ) [SN87] and Parsimony (PAR) [EC63], two similarity-based methods Nearest Neighbour (NN) [MSVP06] and BLAST [AMS$^+$97], and two diagnostic methods DNA-BAR [DKMS05] and BLOG 2.0. These methods were applied to simulated data (available on `dmb.iasi.cnr.it/blog-downloads.php`) as well as to empirical DNA Barcode data sets published by [MP05], [LG10] and [DPC10].
Realistic DNA Barcode datasets were simulated using Mesquite version 2.75 [MM09] by considering time of species divergence and effective population size (Ne), the number of individuals in a population (of a species) that are contributing genes to the succeeding generations. Gene trees with 1000, 10000 and 50000 individuals of effective population size were simulated, generating data sets composed of 50 species each of 20 individuals. Each simulation was replicated 100-fold. The simulated data sets are available on
`dmb.iasi.cnr.it/blog-downloads.php`. The methods were also tested on three empirical data sets published in previous studies and available from GenBank: Drosophila [LG10], 615 DNA Barcodes sequences from 19 species (insects); Inga [DPC10], 913 multi-locus DNA Barcode sequences from 56 species (genus of tropical leguminous trees - Fabaceae); and Cypraeidae [MP05], 2008 mtDNA COI sequences of 211 species (Mollusca). The experimental results can be quickly summarized: for the real data sets BLOG 2.0 classified, on aver-

age, the query sequences with better recognition results (93.1% average success score), DNA-BAR followed with 90.4% correct rate and had the best score in two of three empirical data sets. Neighbour Joining was the best of the two similarity based methods. Detailed classification results are reported in table 3.2. BLOG had also, on average, the highest correct query sequences classification

	Drosophila	**Inga**	**Cypaeidae**	**Avg**
NJ	83.9	91.3	91.5	88.9
PAR	83.9	81.4	85.3	83.5
NN	82.2	88.4	91.2	87.3
BLAST	83.9	82.6	92.7	86.4
DNA-BAR	83.9	**94.2**	**93.2**	90.4
BLOG	**96.6**	90.1	92.7	**93.13**

Table 3.2: Classification results in %

score in simulated data (86.2%). Also DNA-BAR (diagnostic method), Nearest Neighbor and BLAST (similarity based methods) obtained very promising results, with 85.7% and 85.6% respectively, underperforming only slightly BLOG and beating significantly the tree-based methods. Detailed classification results are reported in table 3.3.

	Ne1000	**Ne10000**	**Ne50000**	**Avg**
NJ	83.7	85.5	84.2	84.5
PAR	73.3	79.8	79.5	77.5
NN	86.2	86.1	84.8	85.7
BLAST	86.2	86.1	84.6	85.6
DNA-BAR	**86.3**	86.8	**85.2**	86.1
BLOG	86.0	**88.2**	84.6	**86.2**

Table 3.3: Classification results in %

In figure 3.5 a bar plot is reported that shows the average correct rates of the different methods on the real and simulated data sets. Results showed that similarity-based and diagnostic methods such as BLOG 2.0 significantly outperform other methods, when applied to simulated DNA Barcode data. The diagnostic method BLOG had highest correct query identification rate (see figure 3.5) based on simulated as well as empirical data, indicating that it is a consistently better method overall [vVWFB12]. To consolidate the perfor-

mance of BLOG, the software was tested on two new data sets, the first composed of internal transcribed spacer (ITS) gene region Barcode fungi sequences and the second containing ribulose-bisphosphate carboxylase gene (rbcl) region green algae Barcode sequences (both available on http://dmb.iasi.cnr.it/blog-downloads.php). In particular 50 fungi sequences belonging to eight different species in the Dikarya subkingdom and 26 green algae sequences of five different species in the Haematococcaceae family were extracted from BOLD [RH07]. The results were in line with the classification rates obtained with previous experiments on COI and ITS sequences: for fungi 92% correct classification rates (sensitivity 0.923, specificity 1), for algae 100% correct classification rates (sensitivity 1, specificity 1) and compact classification formulas composed of one or two nucleotides locations.
Beyond the promising classification results, the distinctive advantage of BLOG is the output of the model, which gives a compact and precise description of species in the reference library. BLOG offers additional species-level information - the logic classification formulas - that may also be used outside the scope of DNA barcoding, in species description or in molecular detection. For example, they could potentially be used for designing detection assays based on species-specific nucleotides such as DNA chips or microarrays [ADB+11, vVWFB12].

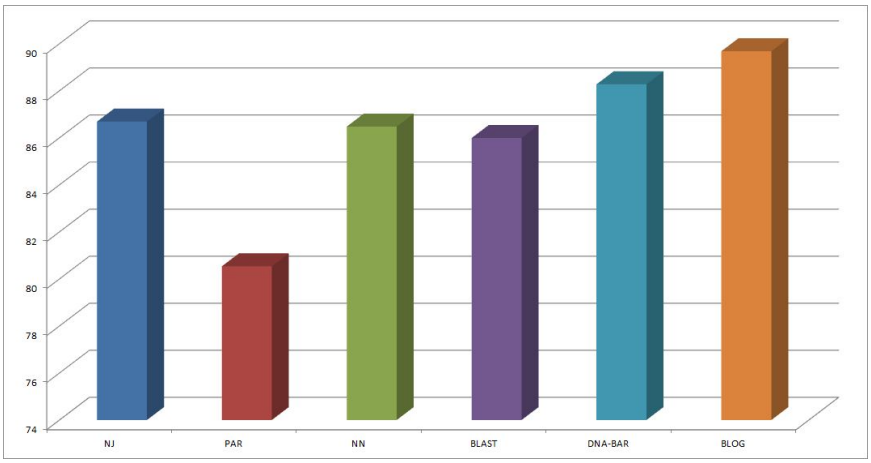


Figure 3.5: BLOG comparative results (correct percentage rates)

Conclusions

The DNA Barcode analysis software tool BLOG 2.0 is available in various user releases. The aim of BLOG is to generate logic formulas that can be used to accurately identify different species and the version 2.0 includes a number of improvements over the previous versions in terms of feature selection, formula extraction, classification, software engineering and enhanced user interfaces. It is available at `dmb.iasi.cnr.it/blog.php` in three versions: an offline user interface, an offline command line release for all common operating systems and a web service. Results indicate that BLOG 2.0 outperforms other DNA Barcode classification methods and that it is not only a major improvement over its previous version but also a valuable addition to the existing suite of methods for analyzing DNA Barcodes. Another distinctive advantage of BLOG is the output of the model, in terms of logic formulas, which gives a compact and precise description of species in the reference library. BLOG offers additional species-level information, the logic classification formulas, that can be used outside the scope of DNA Barcoding, e.g. in species description and in molecular detection [vVWFB12].

3.3 Gene expression profiles analysis

Part of the following section was published in [WFB12, ADB+11].

Introduction

A microarray or DNA array is a semiconductor device, whose aim is to determine the expression level of a large set of genes in one experimental sample with a unique parallel experiment [SSDB95, CYH+96]. By combining many microarrays of different experimental samples a grid of multiple rows and columns is obtained. Each row represents a gene and each column an experimental sample. A cell of the array is associated to a probe DNA sequence, hybridized by Watson-Crick complementarity to the DNA of a target gene, e.g. the mRNA sequences. mRNA sequences contain the information for the amino acids to form a particular protein. The microarray experimental process composed of the following steps:

- the mRNA sequences are amplified;
- fluorescent tagged;

- poured on the array;
- the array is so hybridized;
- the array is scanned with a laser that measures the quantity of fluorescent light in each cell.

This measures are the expression levels of the current gene that is represented in the row of the given experiment.
The microarray experiments contain a large amount of data, that can be stored in form of a matrix, where each row is associated to a gene expression level and each column to an experimental sample. The set of rows is normally quite larger than the set of columns. It is on average composed by more than ten thousand of rows (genes), in the size of twenty thousand. The set of columns (experimental samples) is in the size of hundred. Microarray data sets are therefore under-sampled. The matrix is very informative and needs to be properly analyzed to obtain the desired biological knowledge.
The DMB system has been customized in a flow for gene expression profiles analysis, resulting in MALA. Microarray Logic Analyzer (MALA) is a clustering and classification software, particularly engineered for microarray and RNA-SEQ gene expression profile analysis. The aims of MALA are to cluster the microarray gene expression profiles in order to reduce the amount of data to be analyzed and to classify the microarray experiments. To fulfill this objective MALA uses a machine learning process based methodology, that relies on 1) Discretization, 2) Gene clustering, 3) Feature selection, 4) Formulas computation, 5) Classification. In this section the methodology, the software design, the different releases and user interfaces of MALA are described. Its strengths are also empathized: the identification of classification formulas that are able to precisely describe in a compact way the different classes of the microarray experiments. Finally the experimental results obtained on a real microarray data set coming from Alzheimer diseased versus control mice microarray probes and on two public available data sets (Psoriasis and Multiple Sclerosis) are shown, and it is proven that MALA is a powerful and reliable software for microarray gene expression analysis.

Methods

New advances of microarray technology and next generation sequencing (RNA-SEQ) lead to a large amount of gene expression data available to biological scientists and bioinformaticians. The necessity to effectively analyze the gene

expression profiles of the microarray experimental samples resulted in the development of different software tools and methods [HLP+06, LZO04, RLG+06, SBB+06]. In this section two particular types of microarray data analysis are taken into consideration:

1. gene clustering;

2. experiments classification.

Gene clustering is the detection of gene groups that manifest similar patterns [RSA10]. Several clustering methods can be applied to group similar genes in the microarray experiment; for a survey on clustering methods the reader could refer to [JTZ04], [XI05] and [MO04, AEH12], where the authors describe another common analysis on microarrays: biclustering, a technique where the genes and the experimental samples are clustered simultaneously.
The aim of experiments classification is to distinguish between two or more classes, to which the different samples belong, e.g. different cell types or tumor vs non tumor tissues [JUA05]. In this section a microarray gene clustering and experiments classification software is proposed: MALA. This software integrates an alternative clustering method and an effective classification approach to distinguish the different experimental samples.

MALA: Microarray Logic Analyzer

MALA is a clustering and classification software, particularly engineered for microarray gene expression analysis. The aims of MALA are to cluster the microarray gene expression profiles, in order to reduce the amount of data to be analyzed, and to classify the microarray experiments. To fulfill this objective MALA uses a machine learning process based methodology, that relies on the computational steps described below. This methodology has been applied to different biological problems, such as species classification with DNA Barcode sequences [BFW09, vVWFB12], Polyomaviruses identification [WLPD+12] and microarray analysis [ADB+11]. For further details on the methodology the reader may refer to chapter 2. The flow of MALA is composed of three main steps:

1. the optional application of discrete cluster analysis (DCA), an efficient gene expression clustering method;

2. the selection of the most relevant (clusters of) genes (feature selection);

3. the identification of the logic formulas that best characterize the microarray samples (formula extraction).

The continuous values representing the gene expressions are discretized into a limited number of intervals for each cell of the array; the obtained discrete variables are then used to select a small subset of the genes that have strong discriminating power for the considered classes; finally the logic classification formulas are extracted.

Input - output

MALA relies on the supervised machine learning paradigm of training and testing: the input data is divided in two disjoint sets, one for training the software, which is used to extract the classification model, and one for testing the accuracy of the computed model. For dividing the data in training and testing percentage split or cross validation sampling methods are supported. The MALA software accepts as input format a comma separated values (csv) file, that can be easily produced by a standard spreadsheet editor and that reflects the intrinsic structure of a microarray experiment: every row of the file contains the expression profile of a gene, every column represents an experimental sample. The heading line should list the class labels of the experimental samples. The complete syntax is reported on the MALA website at `dmb.iasi.cnr.it/faq.php#dmbformat`. MALA main outputs are:

- the gene clusters and its frequencies;
- the experiments classification models as logic formulas (rules in the form of "if-then");
- the classification rates;
- the confusion matrices.

An example of logic classification formula is "IF Aph1b < 0.47 then the experimental sample is CONTROL".

Computational steps

MALA is based on several computational steps, which have been integrated in the software: Discretization, Gene clustering, Feature selection, Formulas computation, Classification and Supplementary Analysis. These steps are described in the following sections.

Discretization

MALA relies on a classification algorithm specifically designed to deal with binary domains; the gene expressions have to be transformed into discrete values and to be discretized. MALA applies a transformation and converts each gene expression into a new discrete variable that is suitable for treatment into a logic framework. MALA supports two types of discretization, that differ on the rule adopted to select the first set of intervals. The first type uses an unsupervised rules, the second a supervised. For further details see section 2.4 in chapter 2 of the dissertation.

Gene clustering

Microarray data sets are characterized by a large number of genes in every sample (in the range of tens of thousands); it is therefore very important to adopt methods to extract a subset of genes able to characterize the model among the exponential number of potential ones. The gene clustering step aim is to group together similar genes, whose expression or co-expression is related. MALA implements a method named Discrete Cluster Analysis (DCA) to obtain this goal, described in chapter 2 sections 2.4, 2.5 and 2.6.
After the discretization step an integer mapping is computed for every gene expression, that represents the interval in which an experiment falls. This integer mapping can be represented in a binary form. Two or more genes are merged into the same cluster when their binary representation over the intervals is equal. Finally, a gene for each cluster is elected as its representative. Clusters composed of a single gene may also be present and are considered as non clustered genes. For further details see section 2.5 and 2.6 in chapter 2.

Feature selection

Feature selection is the identification of a small subset of important attributes or features in a large data set. In order to obtain another substantial reduction of the genes [UA05], for performing the classification of the experiments, MALA applies a feature selection step. It consists in the choice of a small number of (clusters of) genes, that are candidate to distinguish the different classes of the experimental samples. MALA approaches feature selection as a combinatorial problem; it adopts a modified formulation of the test cover optimization model.

To solve the feature selection problem MALA uses an efficient heuristic named GRASP. For further details see section 2.7 in chapter 2.

Formulas computation

After the output of the candidate (cluster of) genes from the GRASP algorithm MALA adopts a MinSat approach [FT02] for learning the logic classification formulas. The Lsquare [FT06] method is part of MALA for computing the model of the microarray data, e.g. the separating "if-then" formulas or rules. MALA computes for every class of the experimental samples the logic classification formulas in Disjunctive Normal Form. The literals of the formula represent inequalities in the form of, e.g. "A01232423 $\geq$ 0.5", and are conjuncted in "AND" and "OR" clauses. For further details see section 2.7 in chapter 2.

Classification

The evaluation of the logic formulas and the classification of the samples to the right class is performed according to the algorithm described in [WVF11]. To summarize the process the formulas are firstly weighted with the Laplace Score [TSK05] on the training set and then applied on the test set for performing the classification assignments. Additional cut offs of logic formulas with suboptimal coverage are done by considering the false positive and true positive rates. For further details see section 2.9 in chapter 2.

Supplementary analysis

Two additional useful microarray statistical tests have been integrated in the software MALA:

- The Pearson correlation;
- The Principal Component Analysis (PCA) [Jol02].

The widely adopted Pearson correlation analysis is available on the gene expression profiles, while the PCA can be computed both on the experiments and on the gene expression profiles. PCA and Pearson correlation may give an alternative grouping of the genes to Discrete Cluster Analysis (DCA), described above. The supplementary analysis are integrated in the graphic user interface of MALA, that is described below.

Software engineering and releases

All the different computational steps have been engineered in an integrated software named MALA, written in ANSI C and compilable on Windows, Linux and Mac OS operating systems. MALA relies on the software architectural pattern Pipes and Filters [BHS07] for computing streams of data. A component diagram is given in figure 3.6. Diverse versions are released and described below.

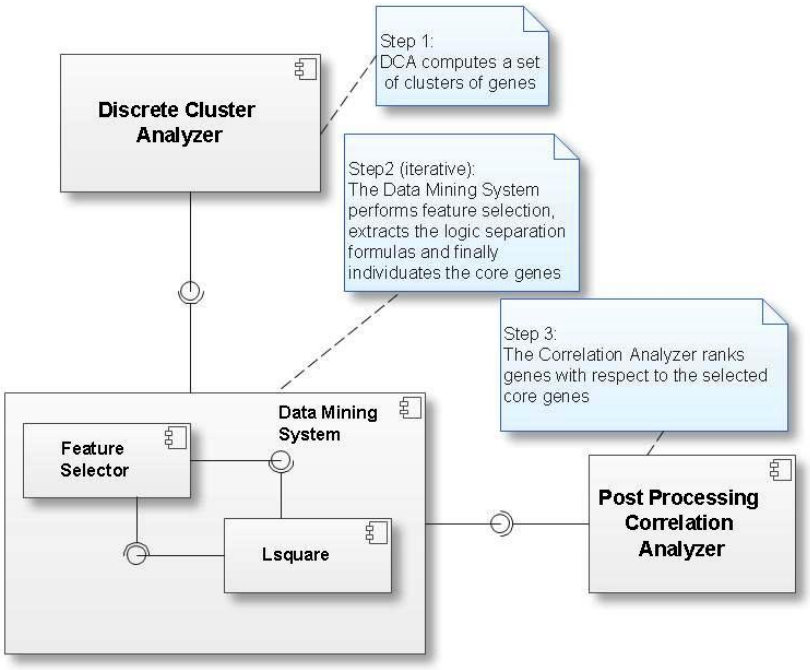


Figure 3.6: MALA component diagram

Graphic user interface

A graphic user interface, downloadable on
`dmb.iasi.cnr.it/mala-downloads.php`, guides the user in the microarray

clustering and classification process. This version uses a Java Swing framework for visualizing the data set, running the software, performing the additional analysis and viewing the clusters, the classification results and the logic separating formulas. A complete user manual is available on the MALA website. A screenshot of the offline graphic user interface is provided in figure 3.7.

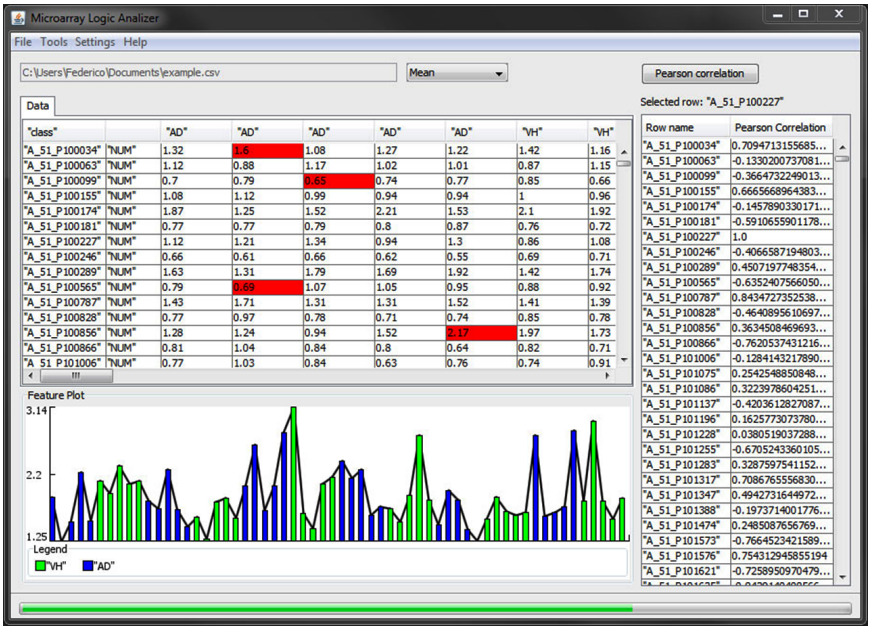


Figure 3.7: MALA graphic offline user interface

Command line version

The command line version is dedicated to the users who want to perform long experiments and batch the entire analysis process. This version is compiled and tested on Linux and Windows operating systems and available on `dmb.iasi.cnr.it/mala-downloads.php`. The source code is also released on the same web site for compiling MALA on alternative systems and a complete user manual is published.

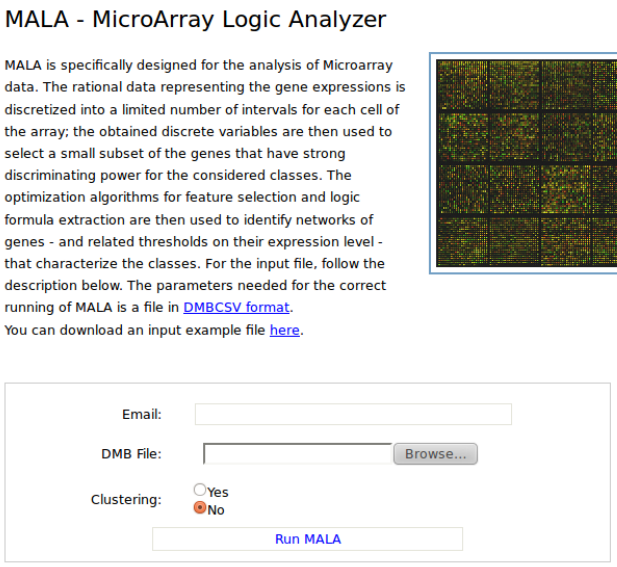


Figure 3.8: MALA web interface

Web service

MALA has been released also as a web service at `dmb.iasi.cnr.it/mala.php`. The user can upload the microarray data via an input form. After the computation, all outputs of MALA are provided in CSV (Comma Separated Values) format, which is easily readable by all common spreadsheet software. A compressed archive containing the computation results is sent via email to the user. The MALA web service has been released on a Linux server (Ubuntu Server distribution), using a LAMP (Linux Apache MySql and PHP) platform (Linux Kernel 2.6.32, Apache 2.2.14, PHP 5.3.2) with a Java job queuing system that relies on a MySQL database (version 5.1.61).
A screenshot of the MALA web service is reported in figure 3.8.

Results and discussion

In this section the experimental results obtained in [ADB+11] to show the effectiveness and the efficacy of MALA on real microarrays gene expression profile

analysis are summarized. Additional tests and comparative analysis with other classification methods have been performed on the same data set and on public available data.
In [ADB+11] MALA was used for the classification of control versus Alzheimer diseased mice, represented in early (1-3 months) and late stage (6-15 months) expression data. A small number of classification formulas was computed, encompassing mRNAs whose expression levels were able to discriminate between diseased and control mice. The purpose of the work was to discover genes whose expression or co-expression strongly characterizes the Alzheimer disease. A small number of genes capable to separate effectively between control and diseased mice was identified. The data set was composed of 119 experimental samples and 16,515 gene expression profiles and was provided from the European Brain Research Institute (EBRI). The application of the Discrete Clustering method DCA (see above) shrinked the whole gene set down to 3656 for 1-3 months and to 3615 for 6-15 months. MALA was capable to identify a few subset of genes and to compute the logic separating formulas for each class. The logic separating formulas for 1–3 and 6–15 months are reported respectively in table 3.4 and table 3.5 as examples. Every disjunctive clause reported in the table is capable to distinguish alone the two different classes of samples. The logic formulas have been validated both using a 30-fold cross validation and a holdout validation (90% train and 10% test), resulting in 99% of correct classification rate.
To reinforce the validity of the results here presented, some of the most commonly used classification algorithms were tested on the same data sets. The WEKA [HFH+09] implementations of K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Random Forest (RF) and C4.5 Classification Tree (C4.5) were adopted. Reasonable parameter tuning was performed, when necessary, for all these methods; the best settings and the correct recognition rates obtained are summarized in the following table 3.6. Experiments were run using a 30-fold cross validation scheme. From the results it can be seen that MALA performs at a comparable level to (slightly dominates) the other methods ; the second best is SVM, that unfortunately produces classification models whose interpretation is very difficult for human beings. As anticipated, the advantages of MALA reside in its ability to extract meaningful and compact models, in its clustering capabilities and in its availability as an integrated tool.
Other tests have been performed on data sets downloaded from public repositories ArrayExpress and GEO: Psoriasis and Multiple Sclerosis Diagnostic. The Psoriasis data set was composed of 54,613 gene expression profiles of 176

	early stage
AD	(Nudt19 $<$ 0.76) OR (Arl16 $\geq$ 1.31) OR (Aph1b $\geq$ 0.47) OR (Slc15a2 $\geq$ 0.55) OR (Agpat5 $\geq$ 0.73) OR (Sox2ot $<$ 0.58 OR Sox2ot $\geq$ 1.53) OR (2210015D19Rik $\geq$ 0.86) OR (Wdfy1 $\geq$ 1.37)
Control	(Nudt19 $\geq$ 0.76) OR (Arl16 $<$ 1.31) OR (Aph1b $<$ 0.47) OR (Slc15a2 $<$ 0.55) OR (Agpat5 $<$ 0.73) OR (0.58 $\geq$ Sox2ot AND Sox2ot $<$ 1.53) OR (2210015D19Rik $<$ 0.86) OR (Wdfy1 $<$ 1.37)

Table 3.4: Logic formulas in early stage

experimental samples (85 control and 91 diseased) and was provided from the National Psoriasis Foundation. The Multiple Sclerosis Diagnostic data set contained 22,215 gene expression profiles of 178 experimental samples (44 control and 134 diseased) and was released from the National Institute of Neurological Disorders and Stroke (NINDS). All gene expression profile values were normalized using the standard Affymetrix Expression Console software (ver 1.2), by the MAS5 algorithm. The results are reported in table 3.7 Also in this case MALA performs at a comparable level to (slightly dominates) the other methods. The second bests are SVM and RF, that unfortunately produce classification models that are difficult to understand for humans.

6-15 months	
AD	(Slc15a2 $\geq$ 0.62) OR (Agpat5 $<$ 0.26 OR Agpat5 $\geq$ 0.55) OR (Sox2ot $\geq$ 1.78) OR (2210015D19Rik $\geq$ 0.82) OR (Wdfy1 $<$ 0.75 OR Wdfy1 $\geq$ 1.29) OR (D14Ertd449e $<$ 0.33 OR D14Ertd449e $\geq$ 0.52) OR (Tia1 $<$ 0.17 OR Tia1 $\geq$ 0.49) OR (Txnl4 $<$ 0.74) OR (1810014B01Rik $<$ 0.71 OR 1810014B01Rik $\geq$ 1.17) OR (Snhg3 $<$ 0.16 OR Snhg3 $\geq$ 0.35) OR [(1.12 $\geq$ Actl6a AND Actl6a $<$ 1.42) OR Actl6a $\geq$ 1.48] OR (Rnf25 $<$ 0.67 OR Rnf25 $\geq$ 1.26)
Control	(Slc15a2 $<$ 0.62) OR (0.26 $\geq$ Agpat5 AND Agpat5 $<$ 0.55) OR (Sox2ot $<$ 1.78) OR (2210015D19Rik $<$ 0.82) OR (0.75 $\geq$ Wdfy1 AND Wdfy1 $<$ 1.29) OR (0.33 $\geq$ D14Ertd449e AND D14Ertd449e $<$ 0.52) OR (0.17 $\geq$ Tia1 AND Tia1 $<$ 0.49) OR (Txnl4 $\geq$ 0.74) OR (0.71 $\geq$ 1810014B01Rik AND 1810014B01Rik $<$ 1.17) OR (0.16 $\geq$ Snhg3 AND Snhg3 $<$ 0.35) OR [(0.81 $<$ Actl6a AND Actl6a $<$ 1.12) OR (1.42 $<$ Actl6a AND Actl6a $<$ 1.48)] OR (0.67 $\geq$ Rnf25 AND Rnf25 $<$ 1.26)

Table 3.5: Logic formulas in late stage.

method	settings	early stage	late stage	model
MALA	no settings	100.0	100.0	yes
SVM	polykernel=2	96.66	100.0	no
RF	trees=100	96.66	94.91	no
C4.5	unpruned, minobj=2	98.33	98.30	yes
KNN	k=2	70.00	86.44	no

Table 3.6: Classification results in %

method	settings	MsDiagnostic	Psoriasis	model
MALA	no settings	94.94	100.0	yes
SVM	polykernel=2	90.45	98.86	no
RF	trees=100	91.57	98.86	no
C4.5	unpruned, minobj=2	87.08	97.16	yes
KNN	k=2	87.64	99.43	no

Table 3.7: Classification results in %

3.4 DMiB: Data Mining in Big

DMiB is a general flow, which can be configured for various types of biological data analysis. DMiB integrates all the computational steps described in chapter 2 of the dissertation:

- sampling;
- discretization;
- clustering;
- feature selection;
- formula extraction;
- noise reduction;
- classification.

DMiB has been successfully applied on Human Polyomaviruses identification, clinical patient trial characterization and on noisy Tag SNP classification.
All these applications are described in the next sections of this chapter (3.5, 3.6, 3.7).

3.5 Polyomaviruses identification

Part of the following section was published in [WLPD+12].

Differences in genomic sequences are crucial for the classification of viruses into different species. The logic data mining method DMB was used to analyze viral DNA sequences in order to identify the nucleotides which are able to distinguish the five different human polyomaviruses. Human polyomaviruses are small double stranded DNA viruses of about 5 kb in length which belong to the Polyomaviridae family. Up to 2008, five species of human polyomaviruses have been identified and characterized: BK (BKPyV), JC (JCPyV), KI (KIPyV), WU (WUPyV) and MC (MCPyV) polyomaviruses. The available gene regions of each species were Small t antigen (ST), Large t antigen (LT), VP1, VP2 and VP3.
A total number of 1982 of sequences has been analyzed in order to find patterns that characterize each species. A pattern is defined as a set of positions of the DNA sequence whose corresponding nucleotides completely characterize the species.
DMB was applied to the 1982 sequences beloging to the three particular gene regions. The sequences required to be aligned before the application of the method. Three types of virus classification were performed considering only the same gene regions, all gene regions together and the 21 virus-gene regions as classes.
When considering only the sequences belonging to the same gene regions DMB was able to distinguish each virus species with a perfect classification rate (100% both in training and testing set in a 100-fold cross validation) and with compact logic formulas. The formulas in table 3.8 indicate the positions and the nucleotide assignments in a particular gene region of the DNA sequence for each polyomavirus species (the different nucleotide positions are calculated from the ATG starting codon). These positions are now under study by experts in order to discover biological meaning and the role played by these nucleotides.

The second analysis was performed by considering all the genes regions together. The main purpose of this analysis was to distinguish the five different polyomaviruses in all the 1982 sequences (here all the genes were considered together). The classification had a perfect recognition rate (100% in 100-fold cross validation). The logic formulas were composed by few literals (from three to five) and all formulas had only one clause in conjunctive normal form (i.e., a sequence of conjunctions).
Another classification analysis was done by distinguishing the different 21 gene

Species	Genes	Formulas
BK	VP1,VP2,VP3	(pos437=A) AND (pos486=C)
JCV	VP1,VP2,VP3	not(pos338=C) AND (pos532=C)
KIV	ST, LT,VP1,VP2,VP3	not(pos294=T) AND not(pos358=T) AND not(pos521=T) AND not(pos532=G)
MCV	ST,LT	(pos199=A) AND not(pos286=T)
WUV	ST, LT, VP1, VP2	not(pos286=T) AND pos425=A AND not(pos474=G)

Table 3.8: Logic formulas for virus classification

regions and polyomaviruses in all the 1982 sequences. For performing the analysis and to validate the results a 100 fold cross validation sampling of the different sequences was applied. Also in this analysis the recognition rate was very high (99% both in training and testing set).
In all cases the models in terms of logic formulas were able to distinguish the different types of viruses. Overall, this is a promising approach that needs to be validated on a larger sample size. For instance, it will be interesting to verify how mutations (polymorphisms, deletions and insertions) in different genomic regions can affect this analysis.
For additional details to this experimental analysis the reader may refer to [WLPD$^+$12].

3.6 Logic mining on clinical patient data trials

In this work the logic mining system DMB was adopted to analyze a large data set of clinical variables of different nature: binary, categorical, integer, real. The focus was on the analysis of the four classes of patients (Normal, Mild Cognitive Impairment, Demented and Depressed), without entering into the details of the different dementia. The adopted flow was composed by two

main modules: feature selection and formula extraction using a subset of these features.

The aim of the analysis was to design a new diagnostic model and possibly a work-flow for the early diagnosis of dementia. Collected data included demographic characteristics, medical history, pharmacological treatments, clinical and neurological examination, psychometric tests, laboratory blood tests, imaging and various other clinical patient trials.

The challenge of an early and precise diagnosis of dementia is only partially solved, in particular at the Mild Cognitive Impairment (MCI) stage, where sensitivity and specificity are still quite low [MME+07]. In fact, current protocols not always lead to an accurate diagnosis, especially in the early stage of disease. The goal was to find a new diagnostic model by mining a large database of clinical variables of MCI, Alzheimer Disease (AD), other demented patients and healthy subjects. The proposed model should be able to discriminate between the different classes of patients.

The source database is a standardized curated collection of psychometric and blood tests, imaging and other clinical data belonging to over 4000 patients, from several Geriatrics and Alzheimer's departments in Italy. The input data come from the ReGAl project involving 36 geriatric memory clinics throughout Italy, coordinated by the University of Perugia. In each centre, data were recorded, then sent to the coordinating centre and processed for quality control and statistical analysis. Collected data included demographic characteristics, medical history, pharmacological treatments, clinical and neurological examination, psychometric tests, laboratory blood tests, imaging (MRI and CT). The data matrix is composed of 748 variables and 4727 samples. The missing data is approximately 30%.

The analysis of the database was based on the DMB system, able to extract salient variables from the whole set and processing together continuous, discrete and categorical data. The DMB system was effective: it was able to identify a subset of characterizing features for every class of disease reported in figure 3.9 and these variables were confirmed by several runs of the method on many random subsets using 80% of data (patients). The model creation was more challenging. A subset of the selected variables was used to build the model. Demented patients were the most easily predicted and the models were usually less complex, Normal subjects were predicted with more difficulties and the task was even harder with the MCI and Depressed classes. The "noise" in MCI and depressed classes might be originated by: less specific criteria to define the MCI state, compared to Dementia one; overlapping value intervals for similar variables in the MCI and Depressed classes. The final model will be

Variables that discriminate between one class form the others			
Normal	**MCI**	**Dementia**	**Depression**
Albumin	Albumin	Albumin	Babcock_1
Anxiety_symptoms	Anxiety_symptoms	Babcock_1	CIRS_cognitive_psychiatric
Azotemia	Azotemia	CIRS_cognitive_psychiatric	CIRS_hypertension_artery
Babcock_1	Babcock_1	CIRS_hypertension_artery	CIRS_skeletal_muscle
CIRS_cognitive_psychiatric	CIRS_cognitive_psychiatric	CIRS_skeletal_muscle	Copy_drawing_corrected
CIRS_hypertension_artery	CIRS_hypertension_artery	Copy_drawing_corrected	Delayed_Recall_corrected
Copy_drawing_corrected	Copy_drawing_corrected		Diffused_hypodensity_CT
Cortical_CT	Cortical_CT	Frontal_horn_hypodensity_CT	FAS
Delayed_Recall_corrected	Delayed_Recall_corrected	How_lives_alone_not	How_lives_alone_not
ECG_patological	ECG_patological	How_long_Stop_drink	How_long_drink
How_lives_alone_not	How_lives_alone_not	IADL_Total	IADL_Total
IADL_Total	How_long_drink	MMSE_Total	Main_job
Main_job	IADL_Total	NPI_Depression	Marital_status
NPI_Depression	Main_job	Years_education	MMSE_Total
Token_test	NPI_Depression		NPI_Depression
	Token_test		Son_daughter_living
			Token_test

Figure 3.9: Characterizing features for every disease

composed by logic formulas connecting few clinical variables, able to classify each patient into a diagnostic category. In conclusion the logic data mining analysis allows to focus just on essential clinical variables and to eventually obtain a good differential diagnosis, being a potential outline for a new faster and cheaper diagnostic work-flow.

3.7 Noisy simulated clinical patients (Tag SNPs)

Single Nucleotide Polymorphism (SNPs) are positions of the DNA sequences where the differences among individuals are embedded [BFF10]. Every individual of the same species presents its genetic variation in only few positions of its genome. Many variations consists in nucleotide changes at specific positions. These variations of single nucleotides are known as single nucleotide polymorphisms (SNPs).
Several experiments with the DMB system were performed injecting a controlled amount of noise in simulated ad hoc noisy data sets of SNPs. In particular, the new noise reduction variants of DMB (presented in chapter 2 sections 2.5, 2.8 and 2.11) were tested on the data provided by Thorsten Lehr and described in [LYZ+11]. The data set aims to simulate complex genetic association studies in terms of SNP (single-nucleotide polymorphism) from different

patients samples. 42 data sets have been generated with distinct patients size (300 and 600), number of attributes (SNP) size (500, 1500, 3000) and with three embedded case-control models (A, B and C). Random noise was injected in each data set leading to different percentages of false positive and false negatives (5%, 10% and 20%). Three different case - control models were used for the data simulation.

Model A
if rs5 = AA and rs10 = AB OR
rs5 = AB and rs10 = AA OR
rs5 = AB and rs10 = BB OR
rs5 = BB and rs10 = AB
then CASE else CONTROL
Model B
if rs5 = BB and rs10 = AA OR
rs15 = AA and AUC > 105
then CASE else CONTROL
Model C
if random number > threshold then CASE else CONTROL

The investigated data sets are reported in table 3.9 (courtesy of Thorsten Lehr). Lehr evaluated the performance of three ruled based classifier algorithms RIPPER (Repeated Incremental Pruning to Produce Error Reduction) [Coh95], RIDOR (RIpple-DOwn Rule) [GC95] and PART [FW98] and ranked them according to the original model similarity (model level). The grading system was the following:
Model Level
Grade A: 100% accordance
Grade B: one attribute missing or in addition
Grade C: two attributes were different from the true model
Grade D: three attributes were different from the true model
Thorsten Lehr defined the following testing procedure. He performed on each of data set 18 experiments with Ripper (18 different paramenter settings), 9 experiments with RIDOR (9 different paramenter settings) and 30 experiments with RIDOR (30 different paramenter settings). For each classifier, if at least one of the different numbers of experiments on one data set reached an A grade, he assigned an A to the classifier. This is a very optimistic grade assignment. According to Thorsten Lehr RIPPER outperformed the other two classifiers. In this experimentation we investigated how DMB with the new noise reduc-

tion procedures is able to perform respect to the methods analyzed by Thorsten Lehr. We point out that our system does not need parameters setting tuning so we performed just one experiment for each data set and assigned the score according to Lehr's grading system. In table 3.10 we show the results of the four ruled based classifiers computed on average. From table 3.10 we see a very good behavior of DMB for the model level experiments. The DMB systems got 12 A, 6 B, 12 C, 6 D and 0 0 scores, on a scale from 0 to 5 the score is 2.66. The best performing method [Coh95] gets similar results (16 A, 5 B, 4 C, 2 D, 9 0), on a scale from 0 to 5 the score is 2.47. The system gets rid of different overfitting effects due to the introduction of noise.
We conclude with two additional observations. The proposed noise reduction methods (described in chapter 2 of the dissertation) appear to be very robust and stable with respect to different data sets, and they do not rely on the use of verification data to tune parameters or evaluate the effect of the pruning; the refinements proposed for error control rely solely on the analysis of the training set. The methods do require an additional computational effort, as in the three phases the problem needs to be solved several times with different parameters; although, such effort is limited and can be assumed be bounded by a small multiple of the effort needed to compute a single iteration. Proper implementation of the discretization and feature selection steps would anyway allow to run the new procedure without additional effort. In the formula extraction phase the cost of running the algorithm several time has to be taken into account, although such cost is typically contained, as the problem has already been drastically reduced in size by the two previous steps.

3.8 Conclusions

In this chapter the main software releases and applications of the DMB system for the classification of biological data have been presented.
Three different software releases have been described. The first software, named BLOG, is devoted to the classification of species with DNA Barcode sequences. It has been applied to several data sets and compared to current available competing methods, resulting in one of the best performing methods.
The second, called MALA, is developed for performing analysis of gene expression profiles and was applied to three different data sets, comparing it to other classification methods. Also in this case MALA is able to compete with the other methods (slightly outperforming them).
The third, DMiB - a general purpose tool, was applied also to real and sim-

ulated biological data sets: human polyomaviruses, clinical patient trials and Tag SNP. DMiB distinguished the known human polyomaviruses, it found the characteristic trials for every patient class in the clinical data set and it was tested with noisy Tag SNP, obtaining in all cases promising results.
The main strength of DMB is the output of the classification model, in terms of logic formulas, which gives a compact and precise description of the analyzed data. DMB offers additional class-level information, the logic classification formulas, that can be used outside the scope of classification - knowledge extraction.

DSet	Md	# SNPs	# Pat	R C/C	FP%	FN%
1	A	500	300	2	5	5
2	A	1500	300	2	5	5
3	A	3000	300	2	5	5
4	A	500	600	2	5	5
5	A	1500	600	2	5	5
6	A	3000	600	2	5	5
7	A	500	300	2	10	10
8	A	1500	300	2	10	10
9	A	3000	300	2	10	10
10	A	500	600	2	10	10
11	A	1500	600	2	10	10
12	A	3000	600	2	10	10
13	A	500	300	2	20	20
14	A	1500	300	2	20	20
15	A	3000	300	2	20	20
16	A	500	600	2	20	20
17	A	1500	600	2	20	20
18	A	3000	600	2	20	20
19	B	500	150+150	2	5	5
20	B	1500	150+150	2	5	5
21	B	3000	150+150	2	5	5
22	B	500	300+300	2	5	5
23	B	1500	300+300	2	5	5
24	B	3000	300+300	2	5	5
25	B	500	150+150	2	10	10
26	B	1500	150+150	2	10	10
27	B	3000	150+150	2	10	10
28	B	500	300+300	2	10	10
29	B	1500	300+300	2	10	10
30	B	3000	300+300	2	10	10
31	B	500	150+150	2	20	20
32	B	1500	150+150	2	20	20
33	B	3000	150+150	2	20	20
34	B	500	300+300	2	20	20
35	B	1500	300+300	2	20	20
36	B	3000	300+300	2	20	20
37	C	500	300	2	n.a.	n.a.
38	C	1500	300	2	n.a.	n.a.
39	C	3000	300	2	n.a.	n.a.
40	C	500	600	2	n.a.	n.a.
41	C	1500	600	2	n.a.	n.a.
42	C	3000	600	2	n.a.	n.a.

Table 3.9: The SNP noisy data sets

Dset	Model	DMB sc	Ripper sc	Ridor sc	Part sc
1	A	B	B	0	0
2	A	A	0	0	D
3	A	D	0	0	0
4	A	A	A	0	0
5	A	B	A	0	0
6	A	C	A	0	0
7	A	B	0	0	0
8	A	A	B	0	0
9	A	C	0	0	0
10	A	A	A	0	0
11	A	A	D	0	0
12	A	A	A	0	0
13	A	A	0	0	0
14	A	A	0	0	0
15	A	A	0	0	0
16	A	A	A	0	0
17	A	A	0	0	0
18	A	A	0	0	0
19	B	B	A	A	A
20	B	B	A	B	A
21	B	C	A	D	C
22	B	C	A	0	A
23	B	B	A	0	B
24	B	C	A	0	A
25	B	D	A	0	A
26	B	C	B	D	B
27	B	D	B	0	0
28	B	D	A	A	A
29	B	C	A	0	B
30	B	C	A	0	0
31	B	C	C	D	0
32	B	D	C	0	0
33	B	C	C	0	0
34	B	C	B	0	0
35	B	C	C	0	0
36	B	D	D	0	0
37	C	0	0	0	0
38	C	0	0	0	0
39	C	0	0	0	0
40	C	0	0	0	0
41	C	0	0	0	0
42	C	0	0	0	0

Table 3.10: Experimental results on the model level

Chapter 4

Alignment free classification of biological sequences

4.1 Introduction

The increasing availability of biological sequences from massive experiments lead to the growth of the field of sequence analysis. Analysis algorithms include methods and techniques from statistics and computer science, like Markov chains and global optimization models. The similarity of sequences is used to prove related biological functions or detect common organisms. Most current sequence analysis methods are based on alignment, i.e. align areas of the sequences sharing common properties. Each computed alignment is evaluated with a score, that depends on the number of same and contiguous characters in the sequences. Optimal methods for sequence alignments rely on dynamic programming techniques, the most adopted optimal sequence alignment algorithms are Needleman and Wunsch [NW70] and Smith Waterman [Pea91].These algorithms are computational demanding and the complexity is exponential in the length of the sequences. Heuristics have been proposed that solve the sequence alignment problem, e.g. BLAST [AMS+97] and FASTA [Pea90]. For performing the alignment of multiple sequences more efficiently several algorithms have been proposed that address this issue like ClustalW [TGH+02], Muscle [Edg04], and Mafft [KMKM02].
The main problems of alignment based methods are the high computational requirements, especially when dealing with a large number of sequences [VA03]. Another issue, when analyzing biological sequences with alignment based meth-

ods, is the non consideration of recombinations and shuffling of the sequences. Aligning the sequences that have to be analyzed, has many disadvantages, as the high computationally requirements, the loss of information; moreover, often sequences are also not alignable. For example in the case of non coding regions, particular segments of the DNA sequence that are not part of a given gene, the computation of an alignment is not possible, because of their noncoding nature and due to their sparsity in the genomic regions.
To solve these limitations alignment free methods for the analysis of biological sequences have been introduced during the last decade, that are particularly suited for sequences that are not easily alignable or for which there is a lot of ambiguity (e.g., non-coding regions or GC-rich regions of DNA)

4.2 General presentation of alignment free methods

Alignment free techniques have been proven to be successful in phylogeny and sequence analysis [VA03]. In alignment free methods the similarity of two sequences is assessed based only on the dictionary of subsequences that apper in the strings, irrespective of their relative position [AC11]. Alignment free methods for sequence comparison are classified in two main groups: methods that rely on subsequences (oligomers) frequencies and methods based on sequence compression [VA03].

Methods based on oligomers frequencies

The methods based on oligomers frequencies are based on the computation of the substrings frequencies of a given length k in the original sequences, called k-mers. A promising alignment free method is the k-mer frequency count analysis [KP09]. The k-mer frequencies of a sequence are represented in a frequency vector, where each component of the vector is associated with the frequency of a particular k-mer [VA03]. Let S be a sequence of n characters over an alphabet A, e.g. $A = A, C, G, T$, and let $k \in I$, $k < n$, $k > 0$. If K is a generic subsequence of S of length k, K is called k-mer. Let the set $V = km_1, km_2, ..., km_r$ be all possible k-mers over A, V has size $r = |A|^k$. The k-mers are computed by counting the occurrences of the substrings in S with a sliding window of length k over S, starting at position 1 and ending at position $n - k + 1$. A vector C contains for each k-mer the counts $C = c_{km1}, c_{km2}, ..c_{kmr}$. The frequencies are then computed accordingly and stored in a vector $F = f_{km_1}, f_{km_2}, ...f_{km_r}$, for a k-mer i the frequency is defined as $f_i = \frac{c_{kmi}}{n-k+1}$. The sequences are so

represented in a coordinate space, that is mathematically tractable with linear algebra and statistics, e.g. by considering the vector representation of the sequences it is possible to compute different distance measures between two sequences or to give the vector represention as input to a classifier.
In current approaches the sequences are then compared according to their vector representations by computing a distance between them, using a distance measure as those described in chapter 1 section 1.3 (Euclidean distance, Manhattan distance, Minkowski distance, Mahalanobis distance, Correlation distance...). A simple and effective distance measure is the Euclidean distance, another very used distance measure is the d2 distance [AC11].

Methods based on sequence compression

The methods based on sequence compression (e.g. Kolmogorov complexity [LV08]) aim is to find the shortest possible description of the sequence and rely on computing the similarity of the sequences by analyzing their compressed representations. These methods are out of the scope of this section, the Kolomogorov complexity and the Universal Sequence Maps [AV02] are cited as current available methods.

Advantages of alignment free methods

The main advantages in aligment free methods are their speed and scalability, they are time linear in the size of the input. In alignment free methods there is no parameter setting, no training and no learning, leading to a parameter-free data mining [AC11].

4.3 Alignment free and logic mining for sequences classification

As a novel technique for biological sequence analysis, logic data mining has been proposed in chapter 3 of the dissertation, in particular for classifying species with DNA Barcode sequences. When analyzing biological sequences with a classical logic data mining approach an alignment between them or an overlapping gene region is necessary, due to the fact that an analysis of the characteristics nucleotides present in a determined position for every class is performed, leading to logic formulas of the type, e.g. "if in pos90=A then the sequence belongs to class X". The alignment is necessary, because a positional

analysis is only possible when the sequences come from the same gene regions and are aligned on a reference position.
As discussed previously computing an alignment has some drawbacks and it is not always possible to align biological sequences (e.g. non coding regions). To overcome the limits of an alignment based approach a technique, that does not take care of the positions, is necessary to perform an effective sequence analysis for sequences that are not easily alignable or for which there is a lot of ambiguity.
This technique, that combines alignment free methods and logic data mining, is called **logic alignment free (laf)**. The technique is based on an alternative method of treating the k-mer frequency vector representation of the sequences (see section 4.1) for performing their classification: it is taken as input for logic data mining methods in order to perform the classification of the biological sequences. A new classification technique for biological sequences is introduced: the combination of alignment free k-mer frequency counts and logic data mining allows the analysis of biological sequences without the strict requirement of an alignment or of an overlapping DNA gene region. This leads to the possibility of performing classification of non coding DNA, which is not alignable, and of whole genomes, which are very hard to align, as the problem of whole genome alignment is computationally hard. The performed experiments show that this technique is very promising in distinguishing functional versus non functional elements inside the same organism and in classifying diverse organisms's whole genomes at different levels of the phylogenetic tree.
In the logic alignment free analysis every sequence is associated to a class, e.g. Vertebrate, Invertebrate; Amniotic, Mammalian. The sequences of a given average length, e.g. 30, are filtered and considered for a classification experiment. For every sequence s in a class data set, e.g. Vertebrate, the following computational steps are performed:

1. The reverse complement of the sequence is computed;
2. The counts of the k-mers (k=3...6) are calculated on the sequence s and on its reverse complement;
3. All counts are normalized obtaining the frequencies of each k-mer;
4. A data set is obtained, where each column represents a sequence and each row a k-mer of nucleotide frequency, e.g. table 4.1;
5. The obtained data sets are given as input to four logic data mining algorithms: RIPPER [Coh95], RIDOR [GC95], PART [FW98] and DMB;

6. The numeric data sets are discretized, i.e. the frequencies are converted from numerical to nominal by the definition of intervals, according to Fayyad and Irani's MDL method [HFH+09] or according to the methods described in chapter 2 section 2.4; the discretization procedure improves the performance of logic data mining algorithms;

7. The classification algorithms are run in 10 fold cross validation mode;

8. The best model in terms of correct classification rates is taken, e.g.
if $freq(AAAC) < 0.195$ then the organism is Vertebrate
if $freq(AAAC) > 0.195$ then the organism is Invertebrate.

	Seq1	**Seq2**	**...**	**SeqM**	**SeqN**
	Vertebrate	**Vertebrate**	...	**Invertebrate**	**Invertebrate**
AAA	0.46	0.26	...	0.24	0.26
AAC	0.12	0.16	...	0.23	0.24
AAG	0.13	0.23	...	0.23	0.22
...	...	...	...	...	...

Table 4.1: Data set construction

The k-mer counts on every sequence are extracted with the Jellyfish software [MK11]. Scripts for filtering, reverse complementing, joining the data sets and calculating the frequencies have been implemented. The Weka [HFH+09] and DMB implementations of the logic data mining algorithms are adopted for performing the classification analysis.

Experimentation

Conserved non coding sequences

In this study a particular class of non coding sequences, conserved non encoding elements sequences (CNEs), were investigated as particular biomarkers for performing the logical analysis and distinguish the different classes of sequences. CNEs are not able to be aligned, because of their noncoding nature and due to their sparsity in the genomic regions.
Therefore a technique, that does not take care of the positions, is necessary to perform an effective comparative analysis. In the present analysis conserved noncoding elements sequences that are already available in the literature

[KP07, BPM+04, SPMM08, VWG+07, GPM+05, Mat09] were used. The experiments have been performed according to the method described in previous section 4.3. The following data sets have been analyzed:

1. 3440 human coding vs human CNEs sequences;
2. 3339 amniotic vs mammalian sequences;
3. 3669 human CNEs vs invertebrate CNEs sequences.

The correct classification percentage rates by applying the laf technique (alignment free k-mer frequency count and four logic data analysis algorithms, RIPPER [Coh95], RIDOR [GC95], PART [FW98], and DMB) are reported in table 4.2. An example of an extracted logic classification formula is the following (the

DataSet	**JRip**	**Ridor**	**Part**	**DMB**	**Average**
1	91.68	90.55	91.25	91.37	91.21
2	93.89	93.93	91.20	91.46	92.62
3	93.88	93.48	94.87	93.05	93.82
Average	93.15	92.65	92.44	91.96	92.55

Table 4.2: CNEs classification rates in % (10-fold cross validation)

frequencies are multiplied by 10^5 for helping the reader):
if 719.462 <= $freq(CGCG)$ < 1075.395 OR 719.462 <= $freq(CTAG)$ < 1075.395 then the sequence is "human coding" else the sequence is a "human CNEs".
From the table we can see that the method performed with very high classification results when analyzing CNEs: in the three experiments the correct classification rates were higher than 90%.
The results were compared with a consolidated bioinformatics method for analyzing sequences, the Genomic Signature analysis (also called GC content analysis), proposed by Karlin [KB95]. Genomic signatures are the most widely used approach to assess compositional preferences; in the past it has been shown that the method is able to efficiently reconstitute taxonomic differences in oligonucleotide usage. Genomic signatures were calculated at the level of dinucleotides and an identical process of classification (using JRip, Ridor, Part and DMB) was performed. The EMBOSS suite [RLB00] was adopted to calculate the GC content of sequences. The correct percentage classification rates obtained by applying the Genomic Signature analysis in combination with logic

DataSet	JRip	Ridor	Part	DMB	Average
1	65.48	61.38	63.63	64.75	63.81
2	74.79	71.94	73.34	72.68	73.18
3	60.10	60.18	58.99	59.97	59.81
Average	66.79	64.50	65.32	65.80	65.60

Table 4.3: GC classification rates in % (10-fold cross validation)

data mining methods are reported in table 4.3. From the results it is shown that the classification efficiency with the use of genomic signatures is much inferior to the one obtained with the proposed methodology.

Whole genome analysis: Bacteria classification

In this section the proposed laf technique (alignment free k-mer counts frequency analysis combined with logic data mining) is applied for classifying bacteria in the different levels of the phylogenetic tree (phylum, class, order, genus, species) by analyzing their whole genome. Whole genomes are very difficult to be aligned, because of their length (more than 2 milion base pairs on average in bacteria, bilions in other more complex organisms). The problem of multiple alignment is computationally demanding and grows exponentially in the size of the input (length and number of sequences) [HME12]. An alignment free technique simplifies considerably the analysis process, permitting an increase of the speed and an effective biological classification of the sequences. 1964 bacteria genome sequences were downloaded from the NCBI database (`ftp.ncbi.nih.gov`). The proposed laf method (alignment free k-mer count analysis in combination with four logic data analysis algorithms, RIPPER [Coh95], RIDOR [GC95], PART [FW98] and DMB), was applied to the bacteria sequences, obtaining also in this case very promising classification results, reported in table 4.4. An example of logic classification formula is the following (the frequencies are multiplied by 10^5 for helping the reader):
if $5558.475 <= freq(AAAA) < 6248.76$ then the sample is a "Campylobacter jejuni".
The results show that the method was able to correctly classify the bacteria genomes in their right taxa. The best performances were obtained at species level with an average correct classification rate of 98%. Species is the lowest level in the phylogenetic tree and it is clearly easier to classify precisely an organism at this level, in higher levels the distinctive characteristics of every

Level	JRip	Ridor	Part	DMB	Average
Species	98.14	97.21	98.71	98.90	98.24
Genus	88.77	86.34	85.53	85.84	86.62
Order	78.44	75.28	76.32	76.08	76.53
Class	80.81	79.86	81.08	80.72	80.61
Phylum	83.80	80.99	80.59	81.54	81.73
Average	85.99	83.93	84.44	84.61	84.74

Table 4.4: Bacteria classification rates (10-fold cross validation)

element are shuffled together, leading to less precise classification models. Although also at higher levels of the phylogenetic tree the classification rates were effective to distinguish the different taxa of bacteria (80% on average).

4.4 Next generation sequencing reads classification

Part of the following section was published in [DCFSW12].

The process of composing a whole DNA genome of an organism from a large set of short sequence partially overlapping fragments, called reads, is defined as *de novo assembly*. Modern sequencing instruments for technological and cost limits produce reads, whose size is very limited respect to the whole genome. The length of a read is commonly between 40 - 200 base pairs (characters), the length of a simple genome, e.g. bacteria, is 2 milion base pairs.
Three major next generation sequencing (NGS) technologies are currently widely adopted: Roche 454, Illumina and Solid. The challenge of these technologies is producing longer reads at a minor cost, because they are easier to assemble [NVP12]. The length of a read is strongly correlated with the cost and speed. Illumina technology actual performances are 40 gigabase pairs per day at a low cost per base pair [`illumina.com`] with reads average length of 70; Roche 454 performances are 1 gigabase pairs per day at a high cost with reads average length of 250 [`454.com`].

The output of a high throughput next generation sequencing (NGS) machine is a collection of short reads, which have to be properly assembled in order to reconstruct the original DNA sequence of the analyzed organism [Met10]. The DNA sequence assembly process is based on aligning and merging these reads for effectively reconstructing the real primary structure of the DNA sam-

ple sequence or reference genome [AEBSJ+11]. The use of NGS machines results in much larger sets of reads to be assembled, posing new problems for computer scientists and bioinformaticians. The current evolving of NGS is characterized by a huge amount of reads available in a single sequencing experiment, all of them having to be processed during the assembly process. Faster methods for filtering the promising read pairs in order to reduce the input of the main assembly algorithm are urgently needed to speed up the reconstruction of the original whole DNA sample sequence [BBF+09]. In particular, a relevant issue is related with the trade-off between precision of the assembly process and its computational time, stating the need for faster methods that can keep pace with the speed and volume of reads that are generated with NGS. An important step in DNA assembly is the identification of a subset of read pairs that have a high probability of being aligned sequentially in the reconstruction. Such a step is often referred to as filtering, and amounts in selecting a significantly smaller subset of the initial set of read pairs (whose dimension is quadratic in the number of initial reads) that can be then processed by an alignment algorithm, usually quite time consuming. The desired effect of filtering is then to quickly filter out from the candidate set of read pairs those that would not provide a good alignment in the following phase. The computation cost of filtering should then be balanced by the speedup obtained when a smaller set of read pairs is considered for alignment.
In this section the use of alignment free distances is proposed and tested to evaluate the similarity between two short reads as a technique for filtering good read pairs to be assembled. The method operates in constant time in the string length and is tested in its ability to emulate, with a proper level of precision, much more time consuming methods to evaluate the similarity between short DNA sequences, such as the established alignment based Needleman-Wunsch edit distance [NW70], often used in the final step of the assembly procedure. These preliminary experiments show the efficacy of this approach for filtering the promising read pairs - eligible candidates to successfully assemble the entire genome of a given organism. Therefore, the alignment free reads filtering may significantly accelerate the assembly process without a substantial loss in accuracy of the DNA sample sequence reconstruction.

DNA sequence assembly

The DNA sequence assembly process is based on the alignment and merging of reads (stretch of sequences) in order to reconstruct the original primary structure of the DNA sample sequences. Reads come from random parts of

the genome and are partially overlapping. The quantity of reads is defined during the sequencing experiment and depends from the adopted coverage for a single base. The coverage is defined as the average number of times a base pair belonging to the genome occurs within the total set of reads [NVP12]. Given a set of sequences $S = \{s_1, s_2, \ldots, s_n\}$, where s ∈ S is a fragment of the primary structure of DNA (read) (e.g. s = ATTCGA...CTGACT), assembly is in charge of building the longest sequence from the set S where each pair of consequent reads obey certain similarity conditions. The assembly problem is proven to be NP-hard [NP09] and several heuristic algorithms, like [BBF+09], have been proposed for effectively solving this problem. The current heuristic algorithms for assembly are based on two main approaches: overlap graphs and De Bruijn Graphs [AEBSJ+11]. In the overlap graphs approach each read and its complement correspond to a node, the overlaps between pairs of reads are calculated with alignment methods, e.g. Needleman and Wunsch, and the weight of the arcs between nodes is determined [BBF+09]. A hamiltonian path in the graph is a good assembly. The drawbacks of this approach are that the alignment algorithm takes $O(kl)$, where k and l are the lengths of the sequences, and the number of possible alignments is $O(n^2)$ where n is the number of sequences. Also most of the sequences do not overlap with each other in a satisfying manner. In the De Bruijn Graphs approach reads are represented on a graph whose nodes and arcs are nucleotides subsequences [CPT11]. The assembly is found searching for an eulerian cycle in this graph and is represented by a sequence of arcs.

DNA read pairs filtering and Alignment Free Distance

This step identifies the promising read pairs in order to reduce the amount of input data given to the real assembly algorithm. A very quick measure of the similarity between two reads, Alignment Free (AF) based distance [VA03], was adopted. AF computes the similarity of two strings based only on the dictionary of their substrings, irrespective of their relative position. As a dictionary the set of 4-mer (sequences composed of 4 different nucleotides) was considered and then a profile for each read composed by the relative frequencies of each 4-mer in the read was built. The Euclidean distance between the profiles of two reads was taken as an inverse measure of the similarity of the two reads and thus as an indication that the two reads formed a promising pair to be considered in the assembly phase. AF filtering was then used defining a proper threshold on the AF distance and discarding all the pairs that exhibited an AF distance above the threshold. Computational complexity of AF distance is a

constant linearly bounded by the number of k-mers adopted and the length of the strings to be compared.

Comparing with other distances: Needleman-Wunsch and "Bowtie" distance

Along with AF the well-established Needleman-Wunsch edit distance (NW) was considered and they were compared in their ability to identify significant pairs. This comparison was based on the computation of a sort of perfect distance computed after an alignment over an already known sequence has been performed. Such distance, referred to as Bowtie distance (BT), was obtained as follows:
a. a large number of reads coming from a known sequence were considered;
b. these reads were aligned over the known sequence using the standard Bowtie algorithm [LTPS09];
c. any two reads received a maximum BT distance if their alignment did not intersect over the reference sequence, else they received a distance inversely proportional to their intersection over the sequence (e.g., they would have BT distance equal to 0 if they were aligned one on top (or inside) of the other by the Bowtie algorithm).
By construction, the BT distance is assumed to be the reference distance, e.g., the distance that expressed the best possible alignments - being based on the knowledge of the reference sequence - and the correlation of AF and NW with BT was tested; moreover, the ability of AF and NW to predict that a given read pair had BT distance above or below a given threshold was verified.

Results and Discussion

For our test the *Escherichia coli* genome and a set of reads from this genome obtained by Roche 454 sequencing machine were considered. Reads have average length of 235 nucleotides and standard deviation of approx. 10 (the large majority of them having length in the interval 225-245). Reads were aligned with the reference sequence with Bowtie and then 100,000 were sampled at random according to their alignment along the sequence. Reads were considered both forward and reverse complemented, giving rise to a total of 200,000 read pairs. All 620,798 read pairs with BT distance < 1 were considered for the experiments; then, out of the remaining pairs, 233,099 were sampled at random. A total of 853,897 read pairs composed the working data set. For all these reads, NW distance and AF distance over the 4-mer were computed. AF,

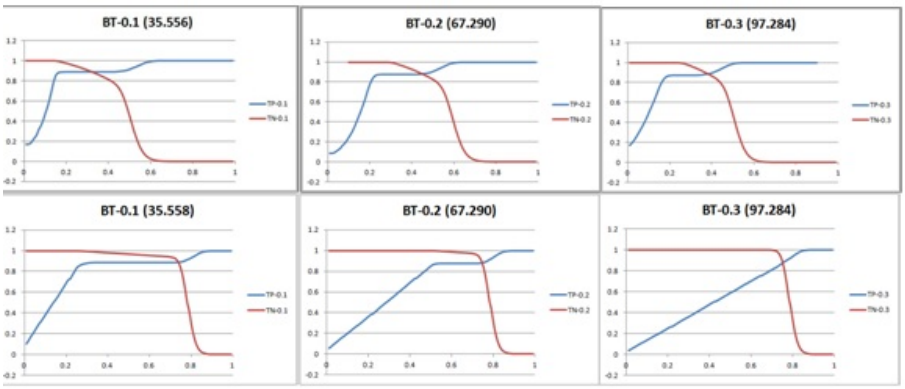


Figure 4.1: Error curves for predictors of BT. Error rates of threshold predictors for BT based on AF are plotted in the charts of the first row; predictors for BT based on NW are in the second row; blue lines represent true positive rates, red lines represent true negative rates.

NW and BT distances were all normalized between 0 (maximal similarity) and 1 (maximal dissimilarity).
The first interesting result was that the correlation between distances showed that AF approximates BT somehow better than NW: a correlation coefficient of 0.761 for AF and BT was obtained, compared with a smaller 0.706 when NW and BT were considered (coherently, correlation between AF and NW was 0,721). The second interesting result was obtained when the ability of AF and NW to predict whether BT was above or below a given threshold was compared. A threshold predictor for a given function F_2 based on function F_1 and on a given pair (α_1, α_2) was defined as follows:
if $(F_1 < \alpha_1)$ then predict $(F_2 < \alpha_2)$, else predict $(F_2 \geq \alpha_2)$.
To a given pair (α_1, α_2), the measure of True Positive rate (TP) (percentage of cases where $(F_1 < \alpha_1)$ and $(F_2 < \alpha_2)$ and of True Negative rate (TN) (percentage of cases where $(F_1 \geq \alpha_1) and (F_2 \geq \alpha_2)$ were associated; analogously False Positive rate (FP) and False Negative rate (FN) were defined . For each (α_1, α_2) with both values ranging from 0 to 1, the positive and negative error rates taking AF as a predictor of BT and NW as a predictor of BT were then computed, with step 0.05. Part of the results are summarized in the charts of figure 4.1, that show for 3 different levels of α_2 (0.1, 0.2, and 0.3) the precision of the predictors (y-axis) when the value of α_1 is changed (x-axis), both when

AF is used as a predictor of BT (charts in the first row) and when NW is used as a predictor of BT (charts in second row). Similar results were obtained also for other levels of α_2, here omitted for brevity. The curves bring to light very clearly how AF is a very good threshold predictor for BT for the considered data; despite its light computational complexity, it appears to perform significantly better than the more complex NW edit distance when its ability to support a threshold predictor is considered.

4.5 Conclusions

In this chapter a new technique based on alignment free sequence analysis and logic data mining called *logic alignment free (laf)*, that classifies biological sequences without the strict requirement of alignment or of an overlapping gene region was illustrated. Two experimental studies have been shown with this technique: whole genome bacteria classification and conserved non encoding regions characterization. Both studies shed to light the power laf: promising classification results of biological sequences, no necessity to align them and identification of common subsequences (kmers) for each class present in the data set.
Additionally, an application of an alignment free metric to next generation sequencing reads filtering was described and compared to other more time consuming alignment based distances.

Conclusions

In this dissertation the field of data mining applied to biological problems and bioinformatics has been investigated. The results were the design, the development and the improvement of a knowledge extraction logic data mining system called DMB. In particular discretization procedures, a clustering method, a feature selection method, a noise reduction method, a classification procedure, and integrated software were designed.
The biological applications of the DMB data mining system lead to the development of three different software releases: BLOG, MALA and DMiB. The first dedicated to the classification of the living species, the second to the analysis of gene expression profiles and the third for multipurpose use.
The model extracted from several experimental analysis were logic formulas able to characterize the different classes of the data set in a clear and compact way. The classification model is a strong plus for the domain expert, that gains a precious and directly interpretable knowledge.
In particular, the DMB system was tested on DNA Barcode sequences, gene expression profiles, polyomaviruses, clinical patient trials, simulated Tag SNP, and whole genome sequences. DMB was proven to perform better than several DNA Barcode species classification methods. Also on gene expression profiles microarray data the system was tested on four different data sets and compared to other standard classification methods, obtaining excellent results. To further prove the efficacy of DMB, it was applied to other real and simulated biological data sets: human polyomaviruses, clinical patient trials and Tag SNP. On polyomaviruses DMB was able to distinguish completely the actually know five human viruses, on clinical patient trials it extracted the characteristic at-

tributes for every class in the data set and on Tag SNPs it was shown that DMB is able to deal with noisy data sets and to obtain slightly better results than other available classification methods.
The DMB system performances are very promising on many different biological data sets. The distinctive advantage of DMB is the output of the classification model, in terms of logic formulas, which gives a compact and precise description of the analyzed data. It offers additional class-level information, the logic classification formulas, that can be used outside the scope of classification - knowledge extraction.
Finally a new method for biological sequence analysis, called *logic alignment free (laf)* has been proposed: the combination of an alignment free method - the k-mer frequency counts - and of logic data mining permits the classification of biological sequences that cannot or are difficult to be aligned, as non encoding DNA regions and whole genomes. This new method has been proven to be successful in correctly classifying functional non encoding regions of the same organism and in distinguishing diverse organisms whole bacteria genomes at different levels of the phylogenetic tree.
In this work the goal of biological knowledge discovery has been proven: it has been shown, with several real experiments, that logic data mining methods - as DMB - are suitable to perform biological knowledge extraction.
Future directions for the work described in this dissertation can be identified as:

- the creation of customized flows of the DMB system for new biological data analysis issues;
- the development methods for listing equivalent alternative models for solving classification problems;
- the design of novel algorithms for the resolution of the feature selection and formula extraction, focusing on exact and probabilistic ones;
- the improvements of the performances of the algorithms with new programming paradigms, e.g. parallelism.

Publications

Journal publications

BLOG 2.0: a software system for character-based species classification with DNA Barcode sequences. What it does, how to use it
E. Weitschek, R. van Velzen, G. Felici and P. Bertolazzi
Molecular Ecology Resources 2013 (doi: 10.1111/1755-0998.12073)

Human polyomaviruses identification by logic mining techniques
E. Weitschek, A. Lo Presti, G. Felici, G. Drovandi, M. Ciccozzi, M. Ciotti and P. Bertolazzi
Virology Journal 58(9), 2012
www.virologyj.com/content/9/1/58

DNA Barcoding of recently diverged species: relative performance of matching methods
R. Van Velzen, E. Weitschek, G. Felici and F.T.Bakker
Plos One 7(1):e30490, 2012
www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0030490

Gene expression biomarkers in the brain of a mouse model for Alzheimer's disease: mining of microarray data by logic classification and feature selection
I. Arisi, M. D'Onofrio, R. Brandi, A. Felsani, S. Capsoni, G. Drovandi, G. Felici, E. Weitschek, P. Bertolazzi and A. Cattaneo
Journal of Alzheimer's Disease 24(4):72138, 2011

Learning to classify species with Barcodes
P. Bertolazzi, G. Felici and E. Weitschek
BMC Bioinformatics 10(S-14):7, 2009
www.biomedcentral.com/1471-2105/10/S14/S7

Conference publications

Filtering with alignment free distances for high throughput DNA reads assembly
M.C. De Cola, G. Felici, D. Santoni, E. Weitschek
EMBnet.journal 2012 (volume S18.B)
http://journal.embnet.org/index.php/embnetjournal/article/view/538

MALA: A microarray clustering and classification software
E. Weitschek, G. Felici and P. Bertolazzi
Biological Knowledge Discovery Workshop, DEXA 2012:201-205

Mining Logic Models in the presence of noisy data
G. Felici and E. Weitschek
Proceedings of Fort Lauderdale FL ISAIM - International Symposium on Artificial Intelligence and Mathematics 2012
www.cs.uic.edu/pub/Isaim2012/WebPreferences/ISAIM2012/Boolean/Felici/Weitschek.pdf

Conference abstracts publications

Logic data mining in the presence of noisy data
G. Felici and E. Weitschek
21st International Symposium on Mathematical Programming 2012
http://ismp2012.mathopt.org/show-abs?abs=1662

A Two Level Randomized Greedy Algorithm For Large-Scale Combinatorial Optimization Problems
P. Festa, G. Felici and E. Weitschek
43rd Annual Conference of the Italian Operational Research Society Graph Algorithms and Optimization 2012
www.airo2012.it/statiche/airo2012/abstracts.pdf

Clustering And Classication Of Microarray Data
E. Weitschek, G. Felici and P. Bertolazzi

43rd Annual Conference of the Italian Operational Research Society 2012
www.airo2012.it/statiche/airo2012/abstracts.pdf

Updates in logic mining for bioinformatics
P. Bertolazzi, G. Felici, G.Drovandi, and E. Weitschek
AIROWinter International Conference 2011
www.iasi.cnr.it/ felici/aw11/aw11/abstracts.pdf

Greedy randomized algorithms with probability learning for classification in bioinformatics
P. Festa, G. Felici and E. Weitschek
41st Annual Conference of the Italian Operational Research Society 2010
http://airo2010.unical.it/AIRO+EWGT_2010-Book_of_abstracts.pdf

Posters

Alignment free high throughput DNA reads filtering with GPGPU computing
E. Weitschek, M.C. De Cola, G. Drovandi, G. Felici and P. Bertolazzi
Intelligent Systems for Molecular Biology (ISMB) 2012

DMB: novel software tools for logic data mining in bioinformatics
E. Weitschek, G. Felici and P. Bertolazzi
Intelligent Systems for Molecular Biology (ISMB) 2011

New diagnostic model for the early diagnosis of Alzheimer's Disease and other dementias, based on Logic Mining of clinical variables
I. Arisi, M. D'Onofrio, R. Brandi, A. Cattaneo, G. Drovandi, G. Felici, E. Weitschek, P. Bertolazzi, S. Brancorsini, S. Ercolani, F. Mangialasche, P. Mecocci
10th International Conference on Alzheimer's and Parkinson's Diseases 2011

Technical Reports

Species classification using DNA Barcode sequences: A comparative analysis
E. Weitschek, R. van Velzen and G. Felici
IASI-CNR, R. 11-07 2011
www.iasi.cnr.it/reports/R11007/R11007.pdf

Human Polyomaviruses genome analysis by logic mining techniques
E. Weitschek, A. Lopresti, G. Felici, G. Drovandi, M. Ciccozzi, M. Ciotti and P. Bertolazzi
IASI-CNR, R. 10-23 2010
www.iasi.cnr.it/reports/R10023/R10023.pdf

Logic Classification and Feature Selection from Gene Expression Profile in the Brain of the AD11 ANTI-NGF Mice Model of Alzheimer's Disease at Different Stages of Neurodegeneration
I. Arisi, M. DOnofrio, R. Brandi, A. Felsani, S. Capsoni, G. Drovandi, G. Felici, E. Weitschek, P. Bertolazzi and A. Cattaneo
IASI-CNR, R. 10-02 2010
www.iasi.cnr.it/reports/R10002/R10002.pdf

Abstract

Advances in molecular biology lead to an exponential growth of biological data, also thanks to the support of computer science. The primary sequences data base GenBank is doubling its size every 18 months, actually consisting in more than 160 billions sequences. The 1000 genomes project released whole DNA sequences of a large number of individuals, producing more than 3000 billions DNA base pairs. Analyzing these enormous amount of data is becoming very important in order to shed light on biological and biomedical questions. The challenges are in managing this huge amount of data, in discovering its interactions and in the integration of the biological know-how. The analysis of biological data requires new methods to extract compact and relevant information; effective and efficient computer science algorithms are needed to support the analysis of complex biological data sets. The interdisciplinary field of data mining, which guides the automated knowledge discovery process, is a natural way to approach the task of biological data analysis. In this dissertation new data mining methods are presented and proven to be effective in many biological data analysis problems. The particular field of logic data mining, where a data classification model is extracted in form of propositional logic formulas, is investigated and a new system for performing a complete knowledge discovery process is described. The system presents new methods for discretization, clustering, feature selection and classification. All methods have been integrated in three different tools: BLOG, MALA and DMiB, the first dedicated to the classification of species, the second to the analysis of gene expression profiles, and the third for multipurpose use. These tools were applied to species classification with DNA Barcode sequences, viruses identifi-

Abstract

cation, gene expression profiles analysis, clinical patient characterization, tag snp classification, non coding DNA identification, and whole genome analysis. A comparison with other data mining methods was performed. The analysis results were all very positive and contributed to the gain of important additional knowledge in biology and medicine, like the detection of nine core genes able to distinguish Alzheimer diseased versus control experimental samples, or the identification of the distinguishing nucleotides positions in the five actually known human polyomaviruses. Moreover, a new technique based on alignment free sequence analysis and logic data mining, that is able to perform the classification of sequences without the strict requirement of computing an alignment between them, is presented. This is a major advantage as the problem of alignment is computationally hard and many biological sequences are not alignable, because of their intrinsic nature, e.g. non coding regions. Also in this case the performed experiments on whole genomes and on conserved non encoding elements show the success of this approach. The models extracted from several analysis are logic formulas able to characterize the different classes of the data set in a clear and compact way. The model is a strong plus for the domain expert, that gains a precious and directly interpretable knowledge.

List of keywords

logic data mining, classification, biological knowledge discovery, bioinformatics

Bibliography

[AC11] Alberto Apostolico and Fabio Cunial. Sequence similarity by gapped lzw. In *DCC*, pages 343–352, 2011.

[AD91] H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proc. 9th National Conference on Artificial Intelligence*, volume 2, pages 547–552. AAAI Press, 1991.

[ADB+11] I. Arisi, M. D'Onofrio, R. Brandi, A. Felsani, S. Capsoni, G. Drovandi, G. Felici, E. Weitschek, P. Bertolazzi, and A. Cattaneo. Gene expression biomarkers in the brain of a mouse model for alzheimer's disease: mining of microarray data by logic classification and feature selection. *Journal of Alzheimer's Disease*, 24(4):721–38, 2011.

[ADS+09] Frederic Austerlitz, Olivier David, Brigitte Schaeffer, Kevin Bleakley, Madalina Olteanu, Raphael Leblois, Michel Veuille, and Catherine Laredo. Dna barcode analysis: a comparison of phylogenetic and statistical classification methods. *BMC Bioinformatics*, 10 Suppl 14:S10, 2009.

[AEBSJ+11] Dent A. Earl, Keith Bradnam, John St. John, Aaron Darling, Dawei Lin, Joseph Faas, Hung On Ken Yu, Buffalo Vince, Daniel R. Zerbino, Mark Diekhans, Ngan Nguyen, Pramila Nuwantha, Ariyaratne Wing-Kin Sung, Zemin Ning, Matthias Haimel, Jared T. Simpson, Nuno A. Fronseca, Inanç Birol, T. Roderick Docking, Isaac Y. Ho, Daniel S Rokhsar, Rayan Chikhi, Dominique

Lavenier, Guillaume Chapuis, Delphine Naquin, Nicolas Maillet, Michael C. Schatz, David R. Kelly, Adam M. Phillippy, Sergey Koren, Shiaw-Pyng Yang, Wei Wu, Wen-Chi Chou, Anuj Srivastava, Timothy I. Shaw, J. Graham Ruby, Peter Skewes-Cox, Miguel Betegon, Michelle T. Dimon, Victor Solovyev, Petr Kosarev, Denis Vorobyev, Ricardo Ramirez-Gonzalez, Richard Leggett, Dan Maclean, Fangfang Xia, Ruibang Luo, Zhenyu L., Yinlong Xie, Binghang Liu, Sante Gnerre, Iain Maccallum, Dariusz Przybylski, Filipe J. Ribeiro, Shuangye Yin, Ted Sharpe, Giles Hall, Paul J. Kersey, Richard Durbin, Shaun D. Jackman, Jarrod A. Chapman, Xiaoqiu Huang, Joseph L. Derisi, Mario Caccamo, Yingrui Li, David B. Jaffe, M. Green, Richard, David Haussler, Ian Korf, and Benedict Paten. Assemblathon 1: A competitive assessment of de novo short read assembly methods. *Genome Research*, September 2011. International competition of de novo genome assembly. The Symbiose team (IRISA/CNRS/ENS Cachan Brittany) participated to this competition.

[AEH12] Wassim Ayadi, Mourad Elloumi, and Jin-Kao Hao. Pattern-driven neighborhood search for biclustering of microarray data. *BMC Bioinformatics*, 13(S-7):S11, 2012.

[AG07] Zaid Abdo and G Brian Golding. A step toward barcoding life: a model-based, decision-theoretic method to assign genes to pre-existing species groups. *Syst Biol*, 56(1):44–56, Feb 2007.

[AMS+97] S. F. Altschul, T. L. Madden, A. A. Schffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–3402, Sep 1997.

[AV02] Jonas S. Almeida and Susana Vinga. Universal sequence map (usm) of arbitrary discrete sequences. *BMC Bioinformatics*, 3:6, 2002.

[BBF+09] Jacek Blazewicz, Marcin Bryja, Marek Figlerowicz, Piotr Gawron, Marta Kasprzak, Edward Kirton, Darren Platt, Jakub Przybytek, Aleksandra Swiercz, and Lukasz Szajkowski. Whole genome assembly from 454 sequencing output via modified dna graph concept. *Comput. Biol. Chem.*, 33(3):224–230, June 2009.

[BFF10] Paola Bertolazzi, Giovanni Felici, and Paola Festa. Logic based methods for snps tagging and reconstruction. *Comput. Oper. Res.*, 37:1419–1426, August 2010.

[BFFL08] Paola Bertolazzi, Giovanni Felici, Paola Festa, and Giuseppe Lancia. Logic classification and feature selection for biomedical data. *Comput. Math. Appl.*, 55:889–899, March 2008.

[BFL10] Paola Bertolazzi, Giovanni Felici, and Giuseppe Lancia. Application of feature selection and classification to computational molecular biology. In S. Lonardi eds. J.K. Chen, editor, *Biological Data Mining*, pages 257–294. Chapman & Hall, 2010.

[BFW09] Paola Bertolazzi, Giovanni Felici, and Emanuel Weitschek. Learning to classify species with barcodes. *BMC Bioinformatics*, 10(S-14):7, 2009.

[BHS07] F. Buschmann, K. Henney, and D. Schmidt. *Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing*. Volume 4. Wiley Chichester, UK, 2007.

[BIK+96] E. Boros, T. Ibaraki, A. Kogan, E. Mayoraz, and I. Muchnik. An implementation of logical analysis of data. Technical Report 29-96, Rutgers University, NJ, 1996.

[BIM99] E. Boros, T. Ibaraki, and K. Makino. Logical analysis of binary data with missing bits. *Artificial Intelligence*, 107:219–263, 1999.

[BLL+02] Koen M. J. De Bontridder, B. J. Lageweg, Jan Karel Lenstra, James B. Orlin, and Leen Stougie. Branch-and-bound algorithms for the test cover problem. In *ESA*, volume 2461 of *Lecture Notes in Computer Science*, pages 223–233. Springer, 2002.

[BPM+04] Gill Bejerano, Michael Pheasant, Igor Makunin, Stuart Stephen, W James Kent, John S. Mattick, and David Haussler. Ultraconserved elements in the human genome. *Science*, 304(5675):1321–1325, May 2004.

[CGK+00] Moses Charikar, Venkatesan Guruswami, Ravi Kumar, Sridhar Rajagopalan, and Amit Sahai. Combinatorial feature selection problems. In *FOCS*, pages 631–640, 2000.

[Coh95] William W. Cohen. Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[CPT11] P.E.C. Compeau, P.A. Pevzner, and G. Tesler. How to apply de bruijn graphs to genome assembly. *Nature biotechnology*, 29(11):987–991, 2011.

[CST00] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[CYH+96] M. Chee, R. Yang, E. Hubbell, A. Berno, X. C. Huang, D. Stern, J. Winkler, D. J. Lockhart, M. S. Morris, and S. P. Fodor. Accessing genetic information with high-density dna arrays. *Science*, 274(5287):610–614, Oct 1996.

[CZBS+12] Laura Clarke, Xiangqun Zheng-Bradley, Richard Smith, Eugene Kulesha, Chunlin Xiao, Iliana Toneva, Brendan Vaughan, Don Preuss, Rasko Leinonen, Martin Shumway, et al. The 1000 genomes project: data management and community access. *nature methods*, 9(5):459–462, 2012.

[DAFM06] Vanda De Angelis, Giovanni Felici, and Giuseppe Mancinelli. Feature selection for data mining. In Triantaphyllou E. eds Felici G., editor, *Data Mining and Knowledge Discovery Apporaches Based on Rule Induction Techniques*, pages 227–252. Massive Computing Series, Springer, 2006.

[Das90] B. V. Dasarathy. *Nearest neighbor (NN) norms: NN pattern classification techniques*. 1990.

[DB95] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *J. Artif. Int. Res.*, 2:263–286, January 1995.

[DCFSW12] Maria C De Cola, Giovanni Felici, Daniele Santoni, and Emanuel Weitschek. Filtering with alignment free distances for high throughput dna reads assembly. *EMBnet. journal*, 18(B):pp–23, 2012.

[DD01] J E Dayhoff and J M DeLeo. Artificial neural networks: opening the black box. *Cancer*, 91(8):1615–1635, 2001.

[DFE09] S. Dulli, S. Furini, and Peron E. *Data Mining*. Springer, 2009.

[DKMS05] B. DasGupta, K. M. Konwar, I. I. Mandoiu, and A. A. Shvartsman. Dna-bar: distinguisher selection for dna barcoding. *Bioinformatics*, 21(16):3424–3426, Aug 2005.

[DL97] M Dash and H Liu. Feature selection for classification. *Intelligent Data Analysis*, 1:131–156, 1997.

[DPC10] KG Dexter, TD Pennington, and CW Cunningham. Using dna to assess errors in tropical tree identifications: How often are ecologists wrong and when does it matter? *Ecol Monogr*, 80:267–286, 2010.

[EC63] AWF Edwards and L.L. CavalliSforza. The reconstruction of evolution. *Annals of Human Genetics*, 27:105–106, 1963.

[Edg04] Robert C Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–1797, 2004.

[Ell12] Mourad Elloumi. 3rd international workshop on biological knowledge discovery and data mining -biokdd 2012- preface. In *23rd International Workshop on DEXA BIOKDD*, 2012.

[FAPB02] Robin Floyd, Eyualem Abebe, Artemis Papert, and Mark Blaxter. Molecular barcodes for soil nematode identification. *Mol Ecol*, 11(4):839–850, Apr 2002.

[fC00] Consortium for CRISP. Cross-industry standard process for data mining. `http://www.crisp-dm.org`, 2000.

[FPSS96] U.C. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37–54, 1996.

[FR89] TA Feo and MGC Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.

[FR95] TA Feo and MGC Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[FR09a] P Festa and MGC Resende. An annotated bibliography of grasp – part i: Algorithms. *International Transactions in Operational Research*, 16(1):1–24, 2009.

[FR09b] P. Festa and M.G.C. Resende. An annotated bibliography of grasp – part ii: Applications. *International Transactions in Operational Research*, 16(2):131–172, 2009.

[FS97] Y. Freund and R. E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[FT02] G. Felici and K. Truemper. A minsat approach for learning in logic domains. *INFORMS Journal on Computing*, 13(3):1–17, 2002.

[FT06] Giovanni Felici and Klaus Truemper. *Encyclopedia of Data Warehousing and Mining, J. Wang (ed.)*, volume 2, chapter The Lsquare System for Mining Logic Data, pages 693–697. Idea Group Inc., 2006.

[FW98] Eibe Frank and Ian H. Witten. Generating accurate rule sets without global optimization. In *IN: PROC. OF THE 15TH INT. CONFERENCE ON MACHINE LEARNING*. Morgan Kaufmann, 1998.

[FW12] Giovanni Felici and Emanuel Weitschek. Mining logic models in the presence of noisy data. In *ISAIM*, 2012.

[GC95] Brian R. Gaines and Paul Compton. Induction of ripple-down rules applied to modeling large databases. *Journal of Intelligent Information Systems*, 1995.

[GJ79] M.R. Garey and D.S. Johnson. *Computer and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[GPM+05] Evgeny A. Glazov, Michael Pheasant, Elizabeth A. McGraw, Gill Bejerano, and John S. Mattick. Ultraconserved elements in insect

genomes: a highly conserved intronic sequence implicated in the control of homothorax mrna splicing. *Genome Res*, 15(6):800–808, Jun 2005.

[Hal00] Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *ICML*, pages 359–366, 2000.

[HCBd03] Paul D N. Hebert, Alina Cywinska, Shelley L. Ball, and Jeremy R. deWaard. Biological identifications through dna barcodes. *Proc Biol Sci*, 270(1512):313–321, Feb 2003.

[HDL+12] James R Hennell, Paul M D'Agostino, Samiuela Lee, Cheang S Khoo, Nikolaus J Sucher, et al. Using genbank® for genomic authentication: a tutorial. *Methods in molecular biology (Clifton, NJ)*, 862:181, 2012.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[HLP+06] Hong Hu, Jiuyong Li, Ashley W. Plank, Hua Wang, and Grant Daggard. A comparative study of classification methods for microarray data analysis. In *AusDM*, pages 33–37, 2006.

[HME12] S. Hosni, A. Mokaddem, and M. Elloumi. A new progressive multiple sequence alignment algorithm. In *Database and Expert Systems Applications (DEXA), BIOKDD 2012*, pages 195 –198, sept. 2012.

[HR01] J. P. Huelsenbeck and F. Ronquist. Mrbayes: Bayesian inference of phylogenetic trees. *Bioinformatics*, 17(8):754–755, Aug 2001.

[Jol02] I.T. Jolliffe. *Principal component analysis*. Springer series in statistics. Springer-Verlag, 2002.

[JTZ04] Daxin Jiang, Chun Tang, and Aidong Zhang. Cluster analysis for gene expression data: a survey. *Knowledge and Data Engineering, IEEE Transactions on*, 16(11):1370 – 1386, 2004.

[JUA05] Thanyaluk Jirapech-Umpai and Stuart Aitken. Feature selection and classification for microarray data analysis: Evolutionary

methods for identifying predictive genes. *BMC Bioinformatics*, 148, 2005.

[KB95] S. Karlin and C. Burge. Dinucleotide relative abundance extremes: a genomic signature. *Trends Genet*, 11(7):283–290, Jul 1995.

[KC04] Lukasz A. Kurgan and Krzysztof J. Cios. Caim discretization algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 16:145–153, 2004.

[Kin98] K. Kincade. Data mining: digging for healthcare gold. *Insurance and Technology*, 23(2):2–7, 1998.

[KMKM02] Kazutaka Katoh, Kazuharu Misawa, Kei-ichi Kuma, and Takashi Miyata. Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic acids research*, 30(14):3059–3066, 2002.

[KP07] Su Yeon Kim and Jonathan K. Pritchard. Adaptive evolution of conserved noncoding elements in mammals. *PLoS Genet*, 3(9):1572–1586, Sep 2007.

[KP09] Pavel Kuksa and Vladimir Pavlovic. Efficient alignment-free dna barcode analytics. *BMC Bioinformatics*, 10 Suppl 14:S9, 2009.

[KR90] L. Kaufman and P.J. Rousseeuw. *Finding Groups in Data An Introduction to Cluster Analysis*. Wiley Interscience, New York, 1990.

[KS97] Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. Technical Report 1997-75, Stanford InfoLab, 1997.

[KT10] H.C. Koh and G. Tan. Data mining applications in healthcare. *Journal of Healthcare Information Management*, 19(2):64–72, 2010.

[LG10] Melanie Lou and G Brian Golding. Assigning sequences to species in the absence of large interspecific differences. *Mol Phylogenet Evol*, 56(1):187–194, Jul 2010.

[LGG+01] Nicholas M Luscombe, Dov Greenbaum, Mark Gerstein, et al. What is bioinformatics? a proposed definition and overview of the field. *Methods of information in medicine*, 40(4):346–358, 2001.

[Lit11] Damon P. Little. Dna barcode sequence identification incorporating taxonomic hierarchy and within taxon variability. *PLoS One*, 6(8):e20552, 2011.

[LTPS09] B. Langmead, C. Trapnell, M. Pop, and S.L. Salzberg. Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biol*, 10(3):R25, 2009.

[LV08] Ming Li and Paul M.B. Vitnyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Publishing Company, Incorporated, 3 edition, 2008.

[LYZ+11] Thorsten Lehr, Jing Yuan, Dirk Zeumer, Supriya Jayadev, and Marylyn Ritchie. Rule based classifier for the analysis of gene-gene and gene-environment interactions in genetic association studies. *BioData Mining*, 4(1):4, 2011.

[LZO04] Tao Li, Chengliang Zhang, and Mitsunori Ogihara. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, 20(15):2429–2437, Oct 2004.

[Mac67] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[Mat09] John S. Mattick. Deconstructing the dogma: a new view of the evolution and genetic programming of complex organisms. *Ann N Y Acad Sci*, 1178:29–46, Oct 2009.

[MBWN08] Kasper Munch, Wouter Boomsma, Eske Willerslev, and Rasmus Nielsen. Fast phylogenetic dna barcoding. *Philos Trans R Soc Lond B Biol Sci*, 363(1512):3997–4002, Dec 2008.

[Met10] Michael L. Metzker. Sequencing technologies - the next generation. *Nat Rev Genet*, 11(1):31–46, Jan 2010.

[Mil00] A. Milley. Healthcare and data mining. *Health Management Technology*, 21(8):44–47, 2000.

[MK11] Guillaume Marais and Carl Kingsford. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764–770, Mar 2011.

[MM09] W.P. Maddison and D.R. Maddison. Mesquite: a modular system for evolutionary analysis. http://mesquiteproject.org/, 2009.

[MME+07] E. Mariani, R. Monastero, S. Ercolani, F. Mangialasche, M. Caputo, F.T. Feliziani, D.F. Vitale, U. Senin, and P. Mecocci. Vascular risk factors in mild cognitive impairment subtypes. findings from the regal project. *Dement Geriatr Cogn Disord*, 24(6):448–456, 2007.

[MN05] Mikhail V. Matz and Rasmus Nielsen. A likelihood ratio test for species membership based on dna sequence data. *Philos Trans R Soc Lond B Biol Sci*, 360(1462):1969–1974, Oct 2005.

[MO04] S.C. Madeira and A.L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 1(1):24 –45, 2004.

[MP05] Christopher P. Meyer and Gustav Paulay. Dna barcoding: error rates based on comprehensive sampling. *PLoS Biol*, 3(12):e422, Dec 2005.

[MSVP06] R Meier, K Shiyang, G Vaidya, and Klng Peter. Dna barcoding and taxonomy in diptera: A tale of high intraspecific variability and low identification success. *Systematic Biology*, 55:715–728, 2006.

[Nan11] G. Nandi. An enhanced approach to las vegas filter (lvf) feature selection algorithm. In *Emerging Trends and Applications in Computer Science (NCETACS), 2011 2nd National Conference on*, pages 1 –3, 2011.

[NM06] Rasmus Nielsen and Mikhail Matz. Statistical approaches for dna barcoding. *Syst Biol*, 55(1):162–169, Feb 2006.

[NP09] Niranjan Nagarajan and Mihai Pop. Parametric complexity of sequence assembly: theory and applications to next generation sequencing. *J Comput Biol*, 16(7):897–908, Jul 2009.

[NVP12] Francesca Nadalin, Francesco Vezzi, and Alberto Policriti. Gapfiller: a de novo assembly approach to fill the gap within paired reads. *BMC Bioinformatics*, 13 Suppl 14:S8, 2012.

[NW70] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, March 1970.

[Pea90] W. R. Pearson. Rapid and sensitive sequence comparison with fastp and fasta. *Methods Enzymol*, 183:63–98, 1990.

[Pea91] W. R. Pearson. Searching protein sequence libraries: comparison of the sensitivity and selectivity of the smith-waterman and fasta algorithms. *Genomics*, 11(3):635–650, Nov 1991.

[Qui93] J. Ross Quinlan. *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, 1 edition, January 1993.

[Qui96] J. R. Quinlan. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[RH07] Sujeevan Ratnasingham and Paul D N. Hebert. bold: The barcode of life data system (http://www.barcodinglife.org). *Mol Ecol Notes*, 7(3):355–364, May 2007.

[RLB00] P. Rice, I. Longden, and A. Bleasby. Emboss: the european molecular biology open software suite. *Trends Genet*, 16(6):276–277, Jun 2000.

[RLG+06] Michael Reich, Ted Liefeld, Joshua Gould, Jim Lerner, Pablo Tamayo, and Jill P. Mesirov. Genepattern 2.0. *Nat Genet*, 38(5):500–501, May 2006.

[RSA10] L. Romdhane, H. Shili, and B. Ayeb. Mining microarray gene expression data with unsupervised possibilistic clustering and proximity graphs. *Applied Intelligence*, 33:220–231, 2010.

[SBB+06] Alexander I. Saeed, Nirmal K. Bhagabati, John C. Braisted, Wei Liang, Vasily Sharov, Eleanor A. Howe, Jianwei Li, Mathangi Thiagarajan, Joseph A. White, and John Quackenbush. Tm4 microarray software suite. *Methods Enzymol*, 411:134–193, 2006.

[She00] C. Shearer. The crisp-dm model: the new blueprint for data mining. *Journal of Data Warehousing*, 5(4):13–22, 2000.

[SN87] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 1987.

[SPD08] Indra Neil Sarkar, Paul J. Planet, and Rob Desalle. caos software for use in character-based dna barcoding. *Mol Ecol Resour*, 8(6):1256–1259, Nov 2008.

[SPMM08] Stuart Stephen, Michael Pheasant, Igor V. Makunin, and John S. Mattick. Large-scale appearance of ultraconserved elements in tetrapod genomes and slowdown of the molecular clock. *Mol Biol Evol*, 25(2):402–408, Feb 2008.

[SSDB95] M. Schena, D. Shalon, R. W. Davis, and P. O. Brown. Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270(5235):467–470, Oct 1995.

[SSH+12] Conrad L. Schoch, Keith A. Seifert, Sabine Huhndorf, Vincent Robert, John L. Spouge, C Andr Levesque, Wen Chen, Fungal Barcoding Consortium , and Fungal Barcoding Consortium Author List . Nuclear ribosomal internal transcribed spacer (its) region as a universal dna barcode marker for fungi. *Proc Natl Acad Sci U S A*, 109(16):6241–6246, Apr 2012.

[ST11] Indra Neil Sarkar and Michael Trizna. The barcode of life data portal: bridging the biodiversity informatics divide for dna barcoding. *PLoS One*, 6(7):e14689, 2011.

[TGH+02] Julie D Thompson, Toby Gibson, Des G Higgins, et al. Multiple sequence alignment using clustalw and clustalx. *Current protocols in bioinformatics*, pages 2–3, 2002.

[Tom09] Brian Tomasik. A minimum description length approach to multitask feature selection. *CoRR*, abs/0906.0052, 2009.

[Tru04] K. Truemper. *Design of Logic-Based Intelligent Systems.* Wiley-Interscience, 2004.

[TSK05] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining.* Addison Wesley, 2005.

[UA05] Thanyaluk J. Umpai and Stuart Aitken. Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes. *BMC Bioinformatics*, 6(1):1–11, 2005.

[VA03] Susana Vinga and Jonas Almeida. Alignment-free sequence comparison-a review. *Bioinformatics*, 19(4):513–523, Mar 2003.

[Vap98] V. N. Vapnik. *Statistical learning theory.* Wiley, 1998.

[vVWFB12] Robin van Velzen, Emanuel Weitschek, Giovanni Felici, and Freek T. Bakker. Dna barcoding of recently diverged species: Relative performance of matching methods. *PLoS ONE*, 7(1):e30490, 01 2012.

[VWG+07] Tanya Vavouri, Klaudia Walter, Walter R. Gilks, Ben Lehner, and Greg Elgar. Parallel evolution of conserved non-coding elements that target a common set of developmental regulatory genes from worms to humans. *Genome Biol*, 8(2):R15, 2007.

[WF05] I.H. Witten and E. Frank. *Data Mining.* Morgan Kaufmann, 2005.

[WFB12] Emanuel Weitschek, Giovanni Felici, and Paola Bertolazzi. Mala: A microarray clustering and classification software. In *23rd International Workshop on DEXA, BIOKDD*, 2012.

[WL90] Bernard Widrow and Michael A Lehr. 30 years of adaptive neural networks: Perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990.

[WLPD+12] Emanuel Weitschek, Alessandra Lo Presti, Guido Drovandi, Giovanni Felici, Massimo Ciccozzi, Marco Ciotti, and Paola Bertolazzi. Human polyomaviruses identification by logic mining techniques. *BMC Virology Journal*, 58(9), 2012.

[WVF11] Emanuel Weitschek, Robin VanVelzen, and Giovanni Felici. Species classification using dna barcode sequences: A comparative analysis. Technical Report 11-07, IASI CNR, 2011.

[WvVFB13] Emanuel Weitschek, Robin van Velzen, Giovanni Felici, and Paola Bertolazzi. Blog 2.0: a software system for character-based species classification with dna barcode sequences. what it does, how to use it. *Molecular Ecology Resources*, 2013.

[XI05] Rui Xu and Donald C. Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.