Machine learning in bioinformatics

Pedro Larrañaga, Borja Calvo, Roberto Santana, Concha Bielza, Josu Galdiano, Iñaki Inza, José A. Lozano, Rubén Armañanzas, Guzmán Santafé, Aritz Pérez and Victor Robles

Submitted: 29th July 2005; Received (in revised form): 21st October 2005

Abstract

This article reviews machine learning methods for bioinformatics. It presents modelling methods, such as supervised classification, clustering and probabilistic graphical models for knowledge discovery, as well as deterministic and stochastic heuristics for optimization. Applications in genomics, proteomics, systems biology, evolution and text mining are also shown.

Keywords: *machine learning; bioinformatics; supervised classification; clustering; probabilistic graphical models; optimisation; heuristic; genomics; proteomics; microarray; system biology; evolution; text mining*

Corresponding author. Pedro Larrañaga, Intelligent Systems Group, Department of Computer Science and Artificial Intelligence, University of the Basque Country, Paseo Manuel de Lardizabal, 1, 20018 San Sebastian, Spain. Tel: +34943018045; Fax: +34934015590; E-mail: pedro.larranaga@ehu.es

Pedro Larrañaga is Professor of Computer Science and Artificial Intelligence at the University of the Basque Country. He received MS degree in mathematics from the University of Valladolid in 1981, and PhD in computer science from the University of the Basque Country in 1995. He has published over 40 refereed journal papers. His main research interests are in the areas of evolutionary computation, machine learning, probabilistic graphical models and bioinformatics.

Borja Calvo received MS in Biochemistry in 1999 and Bachelor degree in Computer Science in 2004, both from the University of the Basque Country. Currently he is a PhD student at the University of the Basque Country and a member of the Intelligent Systems Group. His research interests include machine learning methods applied to bioinformatics.

Roberto Santana received PhD in Mathematics from the University of Havana in 2005. At present, he is at the University of the Basque Country as a member of the Intelligent Systems Group. His research interests include estimation of distribution algorithms and bioinformatics.

Concha Bielza received her MS degree in Mathematics in 1989 from Complutense University, Madrid, and PhD in Computer Science in 1996 from Technical University of Madrid, Madrid. She is an Associate Professor of Statistics and Operation Research in the School of Computer Science at Madrid Technical University. Her research interests are primarily in the areas of probabilistic graphical models, decision analysis, metaheuristics for optimization, data mining, classification models and real applications. Her research has appeared in journals like Management Science, Computers and Operations Research, Statistics and Computing, Naval Research Logistics, Journal of the Operational Research Society and as chapters of many books.

Josu Galdiano is currently doing his MS in Computer Science at the University of the Basque Country. His research interests include machine learning methods applied to bioinformatics.

Iñaki Inza is a Lecturer at the Intelligent Systems Group of the University of the Basque Country. His research interests include data mining and search heuristics in general, with special focus on probabilistic graphical models and bioinformatic applications.

José A. Lozano received his BS degrees in Mathematics and Computer Science and the PhD degree from the University of the Basque Country, Spain in 1991, 1992 and 1998, respectively. Since 1999, he has been an Associate Professor of Computer Science at the University of the Basque Country. He has edited three books and has published over 25 refereed journal papers. His main research interests are evolutionary computation, machine learning, probabilistic graphical models and bioinformatics.

Rubén Armañanzas received his MS in Computer Science from the University of the Basque Country in 2004. At present, he is a PhD student and member of the Intelligent Systems Group. His research interests include feature selection, computational biology and bioinformatics.

Guzmán Santafé received his MS in Computer Science from the University of the Basque Country in 2002. At present, he is a PhD student at the University of the Basque Country and member of the Intelligent Systems Group. His research interests include machine learning techniques applied to bioinformatics.

Aritz Pérez received her Computer Science degree from the University of the Basque Country. He is currently pursuing PhD in Computer Science in the Department of Computer Science and Artificial Intelligence. His research interests include machine learning, data mining and bioinformatics. Currently, he is working on supervised classification using Bayesian networks, variable selection and density estimation, focused for continuous domains.

Victor Robles received the MS degree in Computer Engineering and PhD from the Universidad Politécnica de Madrid, in 1998 and 2003, respectively. During 2004, he was a postdoctoral researcher at Harvard Medical School. He is currently an associate professor in the Department of Computer Systems Architecture and Technology at the Universidad Politécnica de Madrid. His research interests include bioinformatics, data mining and optimization. Dr Robles has been involved in the organization of several workshops and publications, as well as in several books on proceedings.

INTRODUCTION

The exponential growth of the amount of biological data available raises two problems: on one hand, efficient information storage and management and, on the other hand, the extraction of useful information from these data. The second problem is one of the main challenges in computational biology, which requires the development of tools and methods capable of transforming all these heterogeneous data into biological knowledge about the underlying mechanism. These tools and methods should allow us to go beyond a mere description of the data and provide knowledge in the form of testable models. By this simplifying abstraction that constitutes a model, we will be able to obtain predictions of the system.

There are several biological domains where machine learning techniques are applied for knowledge extraction from data. Figure 1 shows a scheme of the main biological problems where computational methods are being applied. We have classified these problems into six different domains: genomics, proteomics, microarrays, systems biology, evolution and text mining. The category named 'other applications' groups together the remaining problems. These categories should be understood in a very general way, especially genomics and proteomics, which in this review are considered as the study of nucleotide chains and proteins, respectively.

Genomics is one of the most important domains in bioinformatics. The number of sequences available is increasing exponentially, as shown in Figure 2. These data need to be processed in order to obtain useful information. As a first step, from genome sequences, we can extract the location and structure of the genes [1]. More recently, the identification of regulatory elements [2–4] and non-coding RNA genes [5] is also being tackled from a computational point of view. Sequence information is also used for gene function and RNA secondary structure prediction.

If the genes contain the information, proteins are the workers that transform this information into life. Proteins play a very important role in the life process, and their three-dimensional (3D) structure is a key feature in their functionality. In the *proteomic* domain, the main application of computational methods is protein structure prediction. Proteins are very complex macromolecules with thousands of atoms and bounds. Hence, the number of possible structures is huge. This makes protein structure prediction a very complicated combinatorial problem where optimization techniques are required. In proteomics, as in the case of genomics, machine learning techniques are applied for protein function prediction.

Another interesting application of computational methods in biology is the management of complex experimental data. *Microarray* essays are the best known (but not the only) domain where this kind of data is collected. Complex experimental data raise two different problems. First, data need to be pre-processed, i.e. modified to be suitably used by machine learning algorithms. Second, the analysis of the data, which depends on what we are looking for. In the case of microarray data, the most typical applications are expression pattern identification, classification and genetic network induction.

Systems biology is another domain where biology and machine learning work together. It is very complex to model the life processes that take place inside the cell. Thus, computational techniques are extremely helpful when modelling biological networks [6], especially genetic networks, signal transduction networks and metabolic pathways.

Evolution and, especially phylogenetic tree reconstruction also take advantage of machine learning techniques. Phylogenetic trees are schematic representations of organisms' evolution. Traditionally, they were constructed according to different features (morphological features, metabolic features, etc.) but, nowadays, with the great amount of genome sequences available, phylogenetic tree construction algorithms are based on the comparison between different genomes [7]. This comparison is made by means of multiple sequence alignment, where optimization techniques are very useful.

A side effect of the application of computational techniques to the increasing amount of data is an increase in available publications. This provides a new source of valuable information, where *text mining* techniques are required for the knowledge extraction. Thus, text mining is becoming more and more interesting in computational biology, and it is being applied in functional annotation, cellular location prediction and protein interaction analysis [8]. A review of the application of text mining techniques in biology and biomedicine can be found in Ananiadou and McNaught [9].

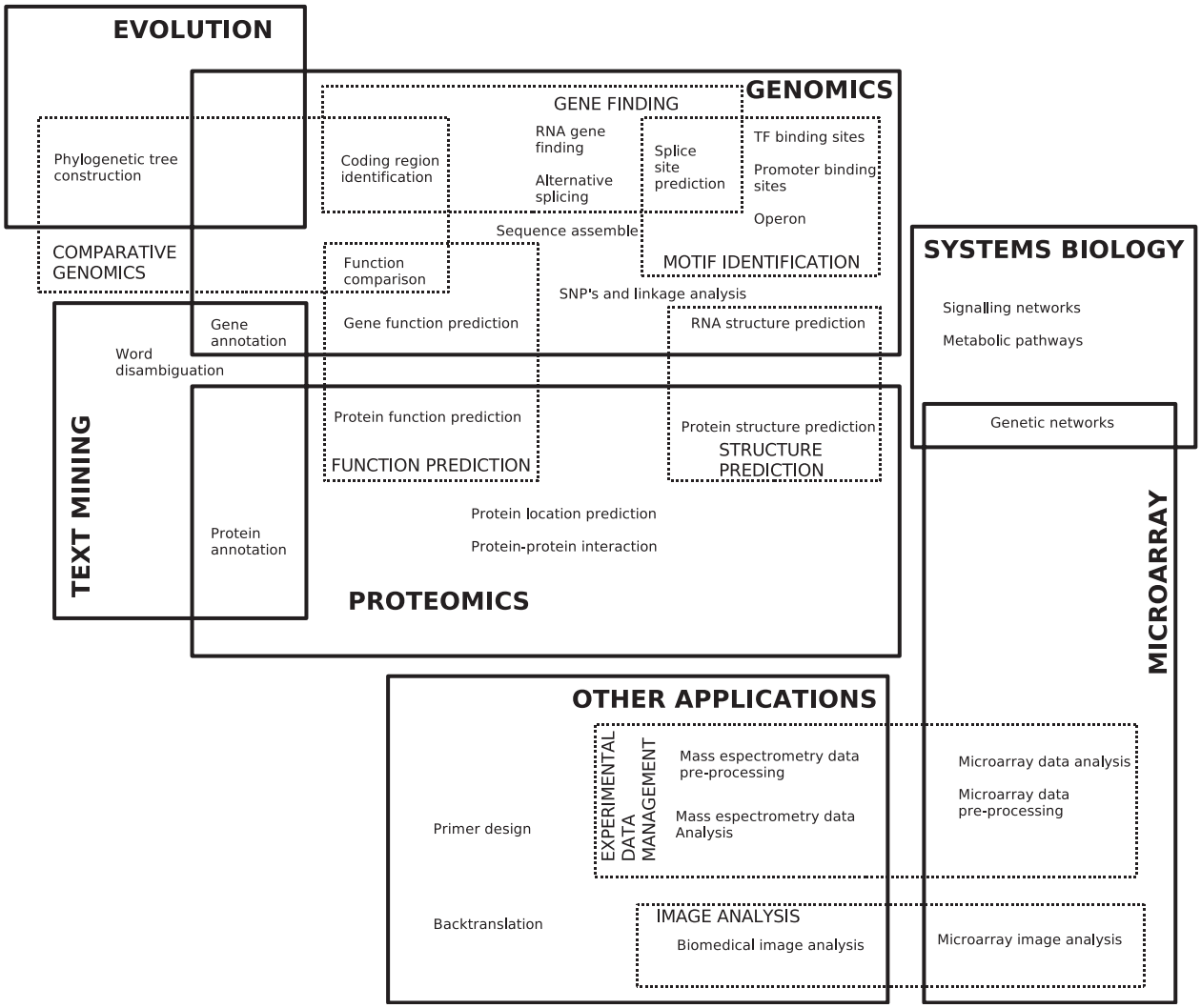

Figure I: Classification of the topics where machine learning methods are applied.

In addition to all these applications, computational techniques are used to solve other problems, such as efficient primer design for PCR, biological image analysis and backtranslation of proteins (which is, given the degeneration of the genetic code, a complex combinatorial problem).

Machine learning consists in programming computers to optimize a performance criterion by using example data or past experience. The optimized criterion can be the accuracy provided by a predictive model—in a modelling problem—, and the value of a fitness or evaluation function—in an optimization problem.

In a *modelling problem*, the 'learning' term refers to running a computer program to induce a model by using training data or past experience. Machine learning uses statistical theory when building computational models since the objective is to make inferences from a sample. The two main steps in this process are to induce the model by processing the huge amount of data and to represent the model and making inferences efficiently. It must be noticed that the efficiency of the learning and inference algorithms, as well as their space and time complexity and their transparency and interpretability, can be as important as their predictive accuracy. The process of transforming data into knowledge is both iterative and interactive. The iterative phase consists of several steps. In the first step, we need to integrate and merge the different sources of information into only one format. By using data warehouse techniques, the detection and resolution of outliers and inconsistencies are solved. In the second step, it is necessary to select, clean and transform the data. To carry out this step, we need to eliminate or correct the uncorrected data, as well as

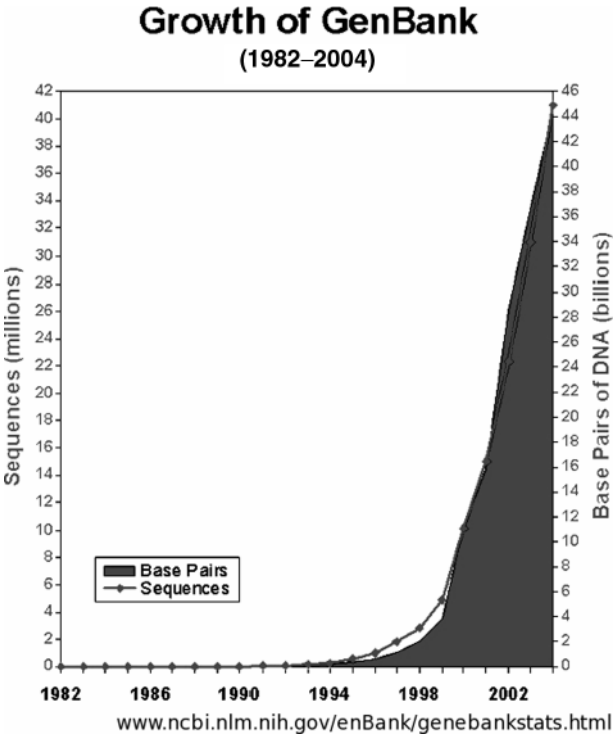

Figure 2: Evolution of the GenBank database size.

decide the strategy to impute missing data. This step also selects the relevant and non-redundant variables; this selection could also be done with respect to the instances. In the third step, called data mining, we take the objectives of the study into account in order to choose the most appropriate analysis for the data. In this step, the type of paradigm for supervised or unsupervised classification should be selected and the model will be induced from the data. Once the model is obtained, it should be evaluated and interpreted—both from statistical and biological points of view—and, if necessary, we should return to the previous steps for a new iteration. This includes the solution of conflicts with the current knowledge in the domain. The model satisfactorily checked—and the new knowledge discovered—are then used to solve the problem.

Optimization problems can be posed as the task of finding an optimal solution in a space of multiple (sometimes exponentially sized) possible solutions. The choice of the optimization method to be used is crucial for the problem solution. Optimization approaches to biological problems can be classified, according to the type of solutions found, into *exact* and *approximate* methods. Exact methods output the optimal solutions when convergence is achieved. However, they do not necessarily converge for every instance. Approximate algorithms always output a candidate solution, but it is not guaranteed to be the optimal one.

Optimization is also a fundamental task when modelling from data. In fact, the process of learning from data can be regarded as searching for the model that gives the data the best fitting. In this search, in the space of models any type of heuristic can be used. Therefore, optimization methods can also be seen as an ingredient at modelling.

There are several reference books on machine learning topics [10–15]. Recently, some interesting books intersecting machine learning and bioinformatics domains have been published [7, 16–27]. Special issues in journals [28–30] have also been published covering machine learning topics in bioinformatics.

The goal of this article is to serve as an insightful categorization and classification of the machine learning methods in bioinformatics including a listing of their applications and providing a context for readers new to the field. Due to space restrictions, this article must not be considered a detailed discussion of the different methods in modelling and optimization.

This article is organized as follows. 'Supervised classification' section presents the supervised classification problem, techniques for assessing and comparing classification algorithms, feature subset selection and several classification paradigms. 'Clustering' section shows different types of clustering—partition clustering, hierarchical clustering and clustering based on mixture models—as well as validation techniques. 'Probabilistic graphical models' section focuses on probabilistic graphical models, a paradigm able to produce supervised and unsupervised models, and to discover knowledge in molecular biology domains. 'Optimization' section shows heuristic optimization methods that have been proposed in bioinformatics to solve some hard computational problems. In all the previous sections, pointers to bioinformatics literature are provided. Final section explains the conclusions of this revision on machine learning methods in bioinformatics.

SUPERVISED CLASSIFICATION

Introduction

In a classification problem, we have a set of elements divided into classes. Given an element (or instance) of the set, a class is assigned according to some of the

element's features and a set of classification rules. In many real-life situations, this set of rules is not known, and the only information available is a set of labelled examples (i.e. a set of instances associated with a class). Supervised classification paradigms are algorithms that induce the classification rules from the data.

As an example, we will see a possible way to tackle splice site prediction as a supervised classification problem. The instances to be classified will be DNA sequences of a given size (for example, 22 base pairs, 10 upstream and 10 downstream the 2 bp splice site). The attributes of a given instance will be the nucleotide at each position in the sequence. In the example, we will assume that we are looking for donor sites, so the possible values for the class will be true donor site or false donor site. As we are approaching the problem as supervised classification, we need a set of labelled examples, i.e. a set of sequences of true and false donor sites along with their label. At this point, we can use this training set to build up a classifier. Once the classifier has been trained, we can use it to label new sequences, using the nucleotide present at each position as an input to the classifier and getting the assigned label (true or false donor site) as an output.

In two-group supervised classification, there is a *feature vector* $\mathrm{X} \in \Re^n$ whose components are called *predictor variables* and a *label* or *class* variable $\mathcal{C} \in \{0, 1\}$. Hence, the task is to induce classifiers from *training data*, which consists of a set of N independent observations $\mathcal{D}_N = \{(\mathbf{x}^{(1)}, c^{(1)}), \ldots, (\mathbf{x}^{(N)}, c^{(N)})\}$ drawn from the joint probability distribution $p(\mathbf{x}, c)$ as shown in Table 1. The classification model will be used to assign labels to new instances according to the value of its predictor variables.

Table 1: Raw data in a supervised classification problem

	X_1	...	X_n	C
$(\mathbf{x}^{(1)}; c^{(1)})$	$x_1^{(1)}$	...	$x_n^{(1)}$	$c^{(1)}$
$(\mathbf{x}^{(2)}; c^{(2)})$	$x_1^{(2)}$	...	$x_n^{(2)}$	$c^{(2)}$
$(\mathbf{x}^{(N)}; c^{(N)})$	$x_1^{(N)}$	...	$x_n^{(N)}$	$c^{(N)}$
$\mathbf{x}^{(N+1)}$	$x_1^{(N+1)}$	...	$x_n^{(N+1)}$	???

Table 2: Confusion matrix in a two classes problem

		Predicted class	
		Positive	**Negative**
True class	Positive	TP: True positive	FN: False negative
	Negative	FP: False positive	TN: True negative

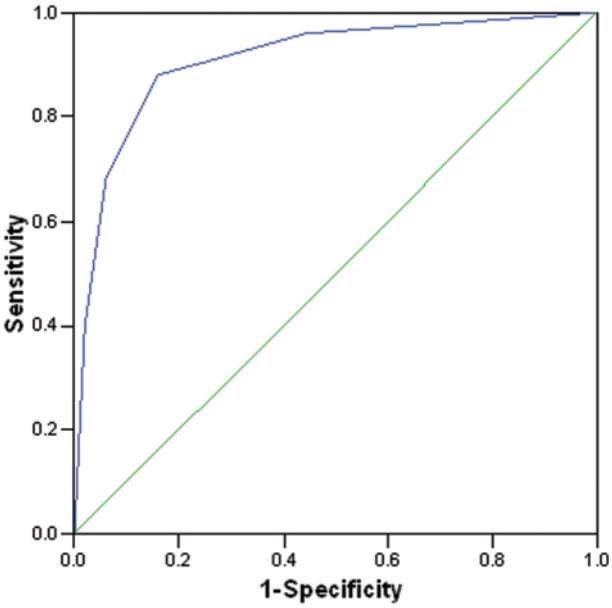

Figure 3: An example of ROC curve.

Assessing and comparing classification algorithms

Error rate and ROC curve

When a 0/1 loss is used, all errors are equally bad, and our error calculations are based on the *confusion matrix* (Table 2). In this case, we can define the *error rate* as $(|FN| + |FP|)/N$, where $N = |TP| + |FP| + |TN| + |FN|$ is the total number of instances in the validation set.

To fine-tune a classifier, another approach is to draw the *receiver operating characteristics* (ROCs) curve [31], which shows hit rate versus false alarm rate, namely, 1-specificity $= |FP|/(|FP| + |TN|)$ versus sensitivity $= |TP|/(|TP| + |FN|)$, and has a form similar to Figure 3. For each classification algorithm, there is a parameter, for example, a threshold of decision, which we can play with to change the number of true positives versus false positives. Increasing the number of true positives also increases the number of false alarms; decreasing the number of false alarms also decreases the number of hits. Depending on how good/costly these are for the particular application we have, we decide on a point on this curve. The area under the receiver operating

characteristic curve is used as a performance measure for machine learning algorithms [32].

Estimating the classification error

An important issue related to a designed classifier is how to estimate its (expected) error rate when using this model for classifying unseen (new) instances.

The simplest and fastest way to estimate the error of a designed classifier in the absence of test data is to compute its error on the sample data itself. This *resubstitution estimator* is very fast to compute and a usually optimistic (i.e. low-biased) estimator of the true error.

In *k-fold cross-validation* [33], $\mathcal{D}_N$ is partitioned into *k folds*. Each fold is left out of the design process and used as a testing set. The estimate of the error is the overall proportion of the errors committed on all folds. In *leave-one-out cross-validation*, a single observation is left out each time, which corresponds to N-fold cross-validation.

The *bootstrap* methodology is a general resampling strategy that can be applied to error estimation [34]. It is based on the notion of an 'empirical distribution', which puts mass $1/N$ on each of the N data points. A 'bootstrap sample' obtained from this 'empirical distribution' consists of N equally likely draws with replacement from the original data set $\mathcal{D}_N$. The *bootstrap zero estimator* and the 0.632 *bootstrap estimator* [35] are used.

In bioinformatics, Baldi *et al.* [36] provide an overview of different methods to assess the accuracy of prediction algorithms for classification. The application of the previous error estimation methods has mainly concentrated on the analysis of supervised classifiers designed for microarray data. In this sense, Michiels *et al.* [37] use multiple random sets for the prediction of cancer outcome with microarrays. Ambroise *et al.* [38] recommend, in a gene selection problem based on microarray gene-expression data, using 10-fold rather than leave-one-out cross-validation. Concerning the bootstrap, they suggest using the so-called 0.632 bootstrap error estimate designed to handle overfitted prediction rules. The superiority of the 0.632 bootstrap estimator in small-sample microarray classification has also been reported [39]. These same authors [40] have recently proposed a new method called *bolstered error estimation* which is superior to bootstrap in feature-ranking performance. A method combining bootstrap and cross-validation has also been proposed with very good results in Fu *et al.* [41].

Comparing classification algorithms

Given two learning algorithms and a training set, an interesting question is to know whether the differences in the estimation of the expected error rates provided by both algorithms are statistically significant. To answer this question, classic [42] and recently proposed tests [43–45] have been used.

Feature subset selection

One question that is common to all supervised classification paradigms is whether all the n descriptive features are useful when learning the classification rule. In trying to respond to this question, the so-called *feature subset selection* (FSS) problem appears, which can be reformulated as follows: given a set of candidate features, select the best subset under some learning algorithm.

This dimensionality reduction made by an FSS process can bring several advantages to a supervised classification system, such as a decrease in the cost of data acquisition, an improvement in the understanding of the final classification model, a faster induction of the final classification model and an increase in the classifier accuracy.

FSS can be viewed as a search problem, with each state in the search space specifying a subset of the possible features of the task. An exhaustive evaluation of possible feature subsets is usually unfeasible in practice because of the large amount of computational effort required. Four basic issues determine the nature of the search process: a search space starting point, a search organization, a feature subset evaluation function and a search-halting criterion.

The *search space starting point* determines the direction of the search. One might start with no features and successively add them, or one might start with all the features and successively remove them. One might also select an initial state somewhere in the middle of the search space.

The *search organization* determines the strategy of the search in a space of size 2^n, where n is the number of features in the problem. The search strategies can be optimal or heuristic. Two classic *optimal search algorithms* which exhaustively evaluate all possible subsets are depth-first and breadth-first [46]. Otherwise, branch and bound search [47] guarantees the detection of the optimal subset for monotonic evaluation functions without the systematic examination of all subsets. When monotonicity cannot be satisfied, depending on the number of features and the evaluation function used, an exhaustive search

can be impractical. In this situation, heuristic search is interesting because it can find near optimal solutions, if not optimal. Among heuristic methods, there are deterministic and stochastic algorithms. On one hand, classic *deterministic heuristic* FSS algorithms are sequential forward and backward selection [48], floating selection methods [49] or best-first search [50]. They are deterministic in the sense that all runs always obtain the same solution and, due to their hill-climbing nature, they tend to get trapped on local peaks caused by interdependencies among features. On the other hand, *stochastic heuristic* FSS algorithms use randomness to escape from local maxima, which implies that one should not expect the same solution from different runs. Genetic algorithms [51] and estimation of distribution algorithms [52] have been applied to the FSS problem.

The *evaluation function* measures the effectiveness of a particular subset of features after the search algorithm has chosen it for examination. Each subset of features suggested by the search algorithm is evaluated by means of a criterion (accuracy, area under the ROC curve, mutual information with respect to the class variable, etc.) that should be optimized during the search. In the so-called *wrapper approach* to the FSS problem, the algorithm conducts a search for a good subset of features using the error reported by a classifier as the feature subset evaluation criterion. However, if the learning algorithm is not used in the evaluation function, the goodness of a feature subset can be assessed by only regarding the intrinsic properties of the data. The learning algorithm only appears in the final part of the FSS process to construct the final classifier using the set of selected features. The statistics literature proposes many measures to assess the goodness of a candidate feature subset [53]. This approach to the FSS is called *filter* in the machine learning field.

Regarding the *search-halting criterion*, an intuitive approach is the non-improvement of the evaluation function value of alternative subsets. Another classic criterion is to fix a number of possible solutions to be visited along the search.

The applications of FSS methodology to microarray data try to obtain robust identification of differentially expressed genes. The most usual approach to FSS in this domain is the filter approach, because of the huge number of features from which we obtain information [54–56]. Wrapper approaches have been proposed in Inza *et al.* [57]—sequential wrapper—and in [58–60]—genetic algorithms [61]. Hybrid combinations of filter and wrapper approaches have also been proposed [62].

Classification paradigms

In this section, we introduce the main characteristics of some of the most representative classification paradigms. It should be noticed that, in a domain such as bioinformatics, where the discovery of new knowledge is of great importance, the transparency and interpretability of the paradigm into consideration should also be considered.

Each supervised classification paradigm has an associated decision surface that determines the type of problems the classifier is able to solve. In this sense, a version of the non free-lunch theorem [63] introduced in optimization is also valid for classification—there is no best classifier for all possible training sets.

Bayesian classifiers

Bayesian classifiers [64] minimize the total misclassification cost using the following assignment: $\gamma(\boldsymbol{x}) = arg\ \min_k \sum_{c=1}^{r0} cost\ (k, c)p(c|x_1, x_2, \ldots, x_n)$, where cost$(k, c)$ denotes the cost for a bad classification. In the case of a 0/1 loss function, the Bayesian classifier assigns the *most probable a posteriori class* to a given instance, that is: $\gamma(x) = arg\ \max_c p(c\,|\,x_1, x_2, \ldots, x_n) = arg\ \max_c\ p(c)p(x_1, x_2, \ldots, x_n\,|\,c)$. Depending on the way $p(x_1, x_2, \ldots, x_n\,|\,c)$ is approximated, different Bayesian classifiers of different complexity are obtained.

Naive Bayes [65] is the simplest Bayesian classifier. It is built upon the assumption of conditional independence of the predictive variables given the class (Figure 4). Although this assumption is violated in numerous occasions in real domains, the paradigm still performs well in many situations. The most probable *a posteriori* assignment of the class variable is calculated as

$$c^* = arg\ \max_c p(c|x_1, \ldots, x_n) = arg\ \max_c p(c) \prod_{i=1}^{n} p(x_i|c).$$

The *seminaive Bayes* classifier [66] tries to avoid the assumptions of conditional independence of the predictive variables, given the class variable, by taking into account new variables. These new variables consist of the values of the Cartesian product of domain variables which overcome a condition. The condition is related to the

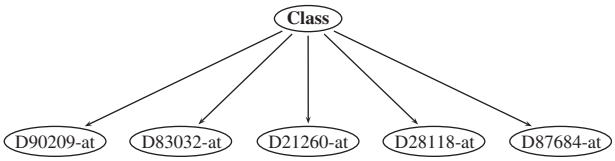

Figure 4: Structure of a naive Bayes model.

independence concept and the reliability on the conditional probability estimations.

The *tree augmented naive Bayes* [67] classifier also takes into account relationships between the predictive variables by extending a naive Bayes structure with a tree structure among the predictive variables. This tree structure is obtained adapting the algorithm proposed by Chow and Liu [68] and calculating the conditional mutual information for each pair of predictive variables, given the class. The tree augmented naive Bayes classification model is limited by the number of parents of the predictive variables. In it, a predictive variable can have a maximum of two parents: the class and another predictive variable. The *k dependence Bayesian* (*k*DB) *classifier* [69] avoids this restriction by allowing a predictive variable to have up to k parents aside from the class.

Logistic regression

The logistic regression paradigm [70] is defined as $p(C=1|\mathbf{x}) = 1/[1+e^{-(\beta_0+\sum_{i=1}^{n}\beta_i x_i)}]$, where $\boldsymbol{x}$ represents an instance to be classified, and $\beta_0, \beta_1, \ldots, \beta_n$ are the parameters of the model. These parameters should be estimated from the data in order to obtain a concrete model. The parameter estimation is performed by means of the maximum likelihood estimation method. The system of $n+1$ equations and $n+1$ parameters to be solved does not have an analytic solution. Thus, the maximum likelihood estimations are obtained in an iterative manner. The Newton–Raphson procedure is a standard in this case.

The modelling process is based on the Wald test and on the likelihood ratio test. The search in the space of models is usually done with forward, backward or stepwise approaches.

Discriminant analysis

Fisher linear discriminant analysis [71] in based on finding linear combinations, $\mathbf{xw}$, of n-dimensional predictor variable values $\mathbf{x}=(x_1,\ldots,x_n)$, with large ratios of between-group to within-group sums of squares. For an $N\times(n+1)$ learning set data matrix, the ratio of between-group to within-group sums of squares is given by $\mathbf{w}'B\mathbf{w}/\mathbf{w}'W\mathbf{w}$, where B and W denote the $n\times n$ matrices of between-group and within-group sums of squares and crossproducts. The extreme values of $\mathbf{w}'B\mathbf{w}/\mathbf{w}'W\mathbf{w}$ are obtained from the eigenvalues and eigenvectors of $W^{-1}B$. Denoting by r_0 the number of values of the class variable C, the matrix $W^{-1}B$ has at most $s=\min(r_0-1, n)$ non-zero eigenvalues, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_s$, with corresponding linear independent eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_s$. The discriminant variables are defined as $u_l=\mathbf{x}\mathbf{v}_l$, $l=1,\ldots,s$ and, in particular, $\mathbf{w}=\mathbf{v}_1$ maximizes $\mathbf{w}'B\mathbf{w}/\mathbf{w}'W\mathbf{w}$.

Linear discriminant analysis constructs, for a two-classes problem, a separating hyperplane between the two datasets. The hyperplane is described by a linear discriminant function $v_1x_1+v_2x_2+\cdots+v_nx_n+c$ which is equal to zero at the hyperplane if two pre-conditions are fulfilled: (i) multivariate normal distribution in both datasets and (ii) homogeneity of both covariance matrices. For discriminant analysis, the hyperplane is defined by the geometric means between the centroids (i.e. the centres of gravity) of the two datasets. To take different variances and covariances in the datasets into account, the variables are usually transformed first into standard means ($\mu=0$) and variances ($\sigma^2=1$) and the Mahalanobis distance (an ellipsoid distance determined from the covariance matrix of the dataset) is more preferable than the Euclidean distance [72].

Classification trees

It is natural and intuitive to classify a pattern through a sequence of questions in which the next question asked depends on the answer to the current question. It is also usual to display the sequence of questions in a directed *classification tree* [73]—also called *classification tree* [74]—where the root node is located at the top, connected by successive and directional links or branches to other nodes. These are similarly connected until we reach terminal or leaf nodes, which have no further links. The classification of a particular pattern begins at the root node, which asks for the value of a particular property of the pattern. The different links from the root node correspond to the different possible values. Based on the answer, we follow the appropriate link to a subsequent or descendant node. In classification trees, the links must be mutually distinct and exhaustive, i.e. one and only one link will be followed. The next step is

to make the decision at the appropriate subsequent node, which can be considered the root of a subtree.We continue this way until we reach a leaf node, which has no further questions. Each leaf node bears a category label, and the test pattern is assigned to the category of the leaf node reached.

The problem of inducing a classification tree model from a set of labelled data can be seen as a problem of organizing the predictor variables into a tree. Any classification tree will progressively split the set of training labelled data into smaller and smaller subsets. In an ideal situation, all samples in each subset would have the same category label. In that situation, we would say that each subset was pure and could terminate that portion of the tree (Figure 5). Usually, however, there is a mixture of labels in each subset. Thus, for each branch, we will have to decide to either stop splitting and accept an imperfect decision, or select another variable and grow the tree further. This suggests an obvious *recursive tree-growing process*: given the data included at a node, either declare that node a leaf (and state which category to assign to it) or find another variable to split the data into subsets.

In order to select the appropriate variable at each level of the tree different, impurity measures—Gini index, gain ratio—have been used. Other questions, such as when a node should be declared a leaf or how a tree that becomes 'too large' can be made smaller and simpler are also noteworthy.

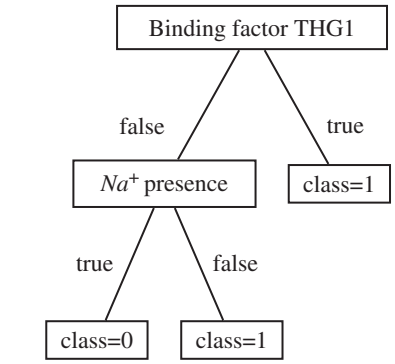

Figure 5: A simple classification tree.

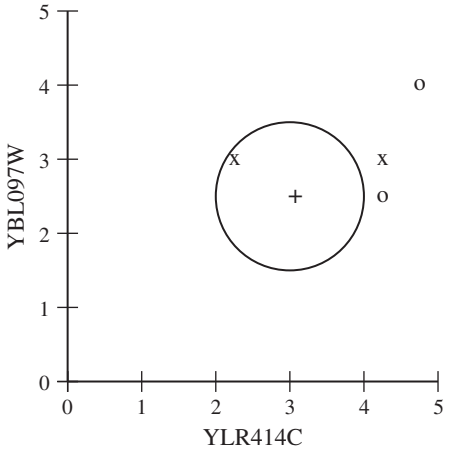

Figure 6: A nearest-neighbour classifier.

Nearest neighbour

The *nearest-neighbour rule* [75] to classify **x** is to asign it to the label associated with the prototype nearest to the test point (Figure 6). An obvious extension of the nearest-neighbour rule is the *k-nearest-neighbour rule*. This rule classifies **x** by assigning it to the label most frequently represented among the k nearest samples. In other words, a decision is made by examining the labels on the k-nearest-neighbours and voting.

A practical problem with this simple method is that it tends to be slow for large training sets because the entire set must be searched for each test instance. A strategy to avoid the computational complexity of the nearest neighbour algorithm is to classify each example with respect to the examples already seen and to save only those that are misclassified. This strategy is known as condensing.

It should be noted that this paradigm does not provide an explicit model of the data. Hence it is said that instead of an induction process, the nearest-neighbour paradigm is based on a *transduction* process that avoids the specification of the model.

Neural networks

Artificial *neural networks* [76] originated from the idea of mathematically modelling human intellectual abilities by means of biologically plausible engineering designs. In an artificial neural network, the elementary processing units (also called 'nodes' or 'neurons') are organised in layers, in such a way that usually only units belonging to two consecutive layers are connected. In a *feedforward neural network* structure, a *unit* will receive information of several units belonging to the previous layer. The most simple neural network, called perceptron [77], is a one-neutron classifier that, using a threshold activation function, separates two classes by a linear discrimination function.

By connecting perceptrons we can design a neural network structure called *multilayer perceptron*. This is a

feedforward structure because the output of the input layer and all intermediate layers are submitted only to the higher layer. The feature vector $\mathbf{x}$ is submitted to an input layer, and at the output layer there are c discriminant functions $g_1(\mathbf{x}), \ldots, g_c(\mathbf{x})$. The number of hidden layers and the number of perceptrons at each hidden layer are not limited. It can be shown that a multilayer perceptron with two hidden layers of threshold nodes can approximate any classification region with a specified precision. Assuming that the structure of the multilayer perceptron is already chosen and fixed, the problem of determining the values of the parameters (weights) for all nodes is solved by the *backpropagation algorithm* [78].

Support vector machines

Support vector machines [79] rely on pre-processing the data to represent patterns in a high dimension—typically much higher than the original feature space. With an appropriate non-linear mapping to a sufficiently high dimension, data from two categories can always be separated by a hyperplane. This choice will often be informed by the designer's knowledge of the problem domain. In absence of such information, one might choose to use polynomials, Gaussians, or other basic functions. The dimensionality of the mapped space can be arbitrarily high (though in practice it may be limited by computational resources).

Defining the margin as any positive distance from the decision hyperplane, the goal in training support vector machines is to find the separating hyperplane with the largest margin. We expect that the larger the margin, the better the generalization of the classifier.

The *support vectors* (Figure 7) are the (transformed) training patterns that are (equally) close to the hyperplane. The support vectors are the training samples that define the optimal separating hyperplane and are the most difficult patterns to classify. Informally speaking, they are the most informative patterns for the classification task.

The problem of minimizing the magnitude of the weight vector constrained by the separation can be reformulated into an unconstrained problem by the method of Lagrange undetermined multipliers. Using the so-called Kuhn–Tucker construction, this optimization can be rewritten as a maximizing problem that can be solved using quadratic programming [80].

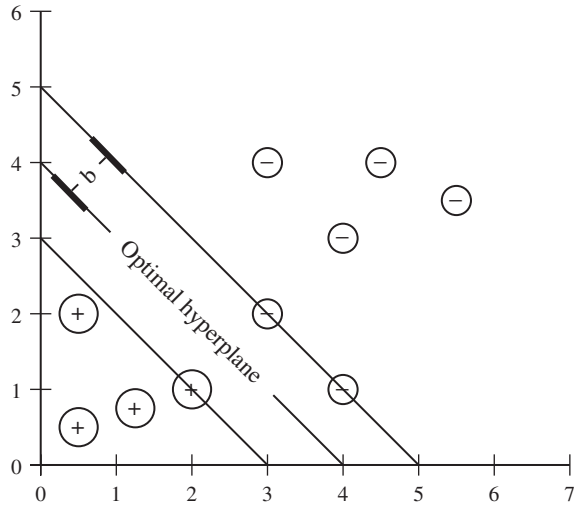

Figure 7: Three support vectors of a support vector machine classifier.

Combining classifires

Each classifier paradigm has an associated decision surface. With the combination of classifiers, our aim is to obtain more flexible decision surfaces and a more accurate decision at the expense of an increased complexity. The combination of classifiers is a field of pattern recognition that is rapidly growing and getting a lot of attention from the machine learning community [81]. An important aspect to be considered is the diversity of the different base classifiers to be combined.

The combination of classifiers can be done in different ways and at different levels. The simplest strategy is the *majority vote*. The unseen instance will be classified as the class that obtains more votes from the different base classifiers whose output labels are fused. Similar strategies used to combine the output labels of the different classifiers are *simple majority* (50% + 1) and *unanimity vote* (all agree).

Another classic way of combining different base classifiers is the so-called *stacked generalization* [82]. The idea is to induce a classifier from the database containing the classification output of each instance of the initial database. This way, the number of features that characterizes the instances coincides with the number of base classifiers. Breiman [83] introduces the concept of *bagging*, as an acronym of Bootstrap AGGregatING. The idea behind bagging is simple and appealing: the ensemble is made of classifiers built—using a unique base classifier—on bootstrap replicates of the training set. The classifier outputs are combined by the majority vote. To make

use of the variations in the training set, the base classifier should be unstable, that is, small changes in the training set should lead to large changes in the classifier output. One of the most unstable classifiers are classification trees. This explains the proposal of Breiman [84] called *random forests*. Random forests is a general class of ensemble building methods using a classification tree as the base classifier. Another traditional way to combine the same base classifier is the *AdaBoost* algorithm [85]. The term comes from ADAptive BOOSTing. The general idea is to develop the classifier team incrementally, adding one classifier at a time. The classifier that joins the ensemble at one step is trained in a dataset selectively sampled from the initial training data set. The sampling distribution begins uniformly, and progresses towards increasing the likelihood of 'difficult' data points. Thus the distribution is updated at each step, increasing the likelihood of the objects misclassified at the previous step.

From the field of statistics, there is one approach to modelling called the *Bayesian approach* which considers all possible structures and, for each structure, all possible values of the parameters. This is called the *full Bayesian approach* to modelling. It can be considered an extreme case of an ensemble of classifiers with only one base classifier.

Supervised classification in bioinformatics

Genomics

One of the most important applications of machine learning techniques can be found in the gene finding problem. Mathé *et al.* [1] is a good review of the methods of gene prediction. Salzberg [86] uses classification trees when searching for protein coding regions in human DNA. In Castelo and Guigó [87] a new Bayesian classifier is applied to the splice site prediction problem.

Feature subset selection has been used in the gene finding problem. For instance, in Saeys *et al.* [88] optimization procedures are applied to the FSS in the splice site prediction problem. Another example of FSS applied to the splice site prediction can be consulted in Degroeve *et al.* [89]. The idea of combining different sources of evidence in gene prediction can be found in Allen *et al.* [90] and in Pablovic *et al.* [91].

An example of the use of classification paradigms in the search for RNA genes can be seen in Carter *et al.* [5]. In this article, support vector machines and neural networks are used in the computational identification of functional RNA genes.

López-Bigas and Ouzounis [92] use classification trees in the genome-wide identification of genes likely to be involved in genetic diseases, taking different conservation scores and gene length as predicting variables.

Other applications of the classification paradigms can be found in Bao and Cui [93] where the authors compare support vector machines to random forests in the prediction of the phenotypic effects of non-synonymous single nucleotide polymorphisms using structural and evolutionary information. In Sebban *et al.* [94] the C4.5 algorithm is used to discover intelligible knowledge rules from data generated by means of a DNA analysis technique (genotyping) called spacer oligonucleotide typing.

The reconstruction of amino-acid sequences by means of spectral features has been addressed using dynamic programming [224]. Dynamic programming [225] is also the type of algorithms preferred for RNA secondary structure prediction. Evolutionary algorithms have been used to identify RNA structural elements [226]. RNA tertiary structure determination has been approached with tabu search [227].

Proteomics

Several applications of nearest neighbour have been done in the prediction of the secondary structure of proteins [95–97]. In Selbig *et al.* [98], a consensus method based on a classification tree for the prediction of the protein secondary structure is presented.

Yang *et al.* [99] develop a two-stage method consisting of a support vector machine and a Bayesian classifier to predict the surface residues of a protein that participate in protein–protein interactions.

The problem of predicting the protein subcellular location automatically from its sequence has been treated with a fuzzy *k*-nearest neighbour algorithm [100].

Microarray

A survey about pattern recognition in microarray data can be found in Valafar [101].

In Krishnapuram *et al.* [102], a Bayesian generalization of the support vector machine is used to simultaneously select the optimal classifier and the optimal subset of genes for cancer diagnosis based on expression data. Nearest neighbour has been used in

Olshen and Jain [103] and in Li *et al.* [59]. In the second article *k*-nearest neighbour is used in conjunction with a genetic algorithm in a wrapper approach for gene selection.

An ensemble approach can be consulted in Tan and Gilbert [104]. This article shows that ensemble learning (bagged and boosted decision trees) performs better than single classification trees in the classification of cancerous gene expression profiles.

Comparisons between different classification paradigms can be found in several works. In Dudoit *et al.* [105], nearest-neighbour classifiers, linear discriminant analysis, classification trees, bagging and boosting are empirically compared in three cancer gene expression studies. A comparison between three binary classifiers (*k*-nearest neighbours, weighted voting and support vector machines) in a classification problem with 14 tumour classes can be found in Ramaswamy *et al.* [106]. Statnikov *et al.* [107] show a very extensive empirical comparison between several major classification algorithms (support vector machines, *k*-nearest neighbour, neural networks, and different ensemble classifiers) in 11 datasets for cancer diagnosis, and Lee *et al.* [108] evaluates the performance of 21 classification methods in 7 datasets in microarray datasets.

Other applications of microarray data can be found in Ben-Dor *et al.* [109], Brown *et al.* [110] and Kim and Cho [111].

Systems biology

Although probabilistic graphical models are the most used approach in systems biology, some works tackle the problem from a supervised point of view. For instance, Hautaniemi *et al.* [112] use classification trees when modelling signal–response cascades, and the methodology is applied to the prediction of cell migration speed using phosphorylation levels of signalling proteins. In Middendorf *et al.* [113], the authors use boosting with classification trees as the base classifier for the prediction of a gene regulatory response, which is considered a binary variable (up- or down-regulated).

Text mining

A review of the text-mining applications in bioinformatics can be found in Krallinger *et al.* [8]. In the BMC Bioinformatics journal, the first special issue of 2005 also concentrates on text-mining. As an example of application, Zhou *et al.* [114] present a new method for protein/gene identification in text, based on an ensemble of a support vector machine and two hidden Markov models ('Probabilistic Graphical Models' section). In Stapley *et al.* [115], support vector machines are used in the prediction of the sub-cellular location of proteins.

Other applications

Two examples of the use of mass spectrometry data can be found in Wu *et al.* [116] and Baumartner *et al.* [117]. Wu *et al.* [116] apply linear discriminant analysis, quadratic discriminant analysis, *k*-nearest-neighbour classifier, bagging and boosting classification trees, support vector machines and random forests in the detection of earlystage ovarian cancer patients using mass spectrometry data as biomarkers. In Baumgartner *et al.* [117], classification algorithms (discriminant analysis, logistic regression, classification trees, *k*-nearest-neighbour classifiers, neural networks, and support vector machines) are empirically compared in the classification of two metabolic disorders in newborns, using data obtained from mass spectrometry technology. Other examples of the use of mass spectrometry data can be found in Li *et al.* [118] and Satten *et al.* [119]. Other problems where computational methods are used can be found in Jung and Cho [120] and Perner *et al.* [121].

CLUSTERING

Introduction

Clustering consists in partitioning a set of elements into subsets according to the differences between them. In other words, it is the process of grouping similar elements together. The main difference from the supervised classification is that, in clustering, we have no information about how many classes there are.

The most typical example of clustering in bioinformatics is the clustering of genes in expression data. In microarray essays, we obtain the expression value for thousands of genes in a few samples. An interesting information we can extract from these data is which genes are coexpressed in the different samples. This is a clustering problem where genes with similar expression level in all samples are grouped into a cluster.

Cluster analysis, also called data segmentation, has a variety of goals. All relate to grouping or

segmenting a collection of objects into subsets or 'clusters', such that those within each cluster are more closely related to one another than objects assigned to different clusters. Sometimes the goal is to arrange the clusters into a natural hierarchy. This involves successively grouping the clusters themselves so that, at each level of the hierarchy, clusters within the same group are more similar to each other than those in different groups.

Central to all of the goals of cluster analysis is the notion of the degree of similarity (or dissimilarity) between the individual objects being clustered. A clustering method attempts to group the objects based on the definition of similarity supplied to it. This can only come from subject matter considerations.

Clustering approaches

Partition clustering

Partition clustering (Figure 8) aims at obtaining a partition of the data. Each point belongs to a unique cluster. It is a common strategy to fix the number of clusters, although some algorithms can search for the most appropriate number of clusters while allocating the objects in the different clusters.

The *K-means algorithm* is one of the most popular iterative descent clustering methods [122]. The aim of the *K*-means algorithm is to partition the data into *K* clusters so that the within-group sum of squares is minimized. The simplest form of the *K*-means algorithm is based on alternating two procedures. The first one is the assignment of objects to groups. An object is usually assigned to the group whose mean is the closest in the Euclidean sense. The second procedure is the calculation of new group means based on the assignments. The process terminates when no movement of an object to another group will reduce the within-group sum of squares.

There are many variants of the *K*-means algorithm that improve its efficiency in terms of reducing the computing time and achieving a smaller error. Some algorithms allow new clusters to be created and existing ones to be deleted during the iterations. Others may move an object to another cluster on the basis of the best improvement in the objective function. Alternatively, the first encountered improvement while passing by the dataset could be used.

A method related to the *K*-means algorithm is vector quantization [123]. The main purpose of vector quantization is to compress data. A vector quantizer consists of two components: an encoder and a decoder. An algorithm known as the *generalized Lloyd algorithm* [124] in the vector quantization literature is clearly a variant of the *K*-means algorithm. Moreover, self-organizing feature maps are a special kind of vector quantization in which there is an ordering or topology imposed on the code vectors. The aim of self-organization is to represent high-dimensional data as a low-dimensional array of numbers (usually in 1D or 2D array) that captures the structure in the original data.

Hierarchical clustering

Hierarchical clustering procedures [125] are the most commonly used methods to summarize data structures in bioinformatics. A hierarchical tree (Figure 9) is a nested set of partitions represented by a tree diagram or *dendrogram*. Sectioning a tree at a

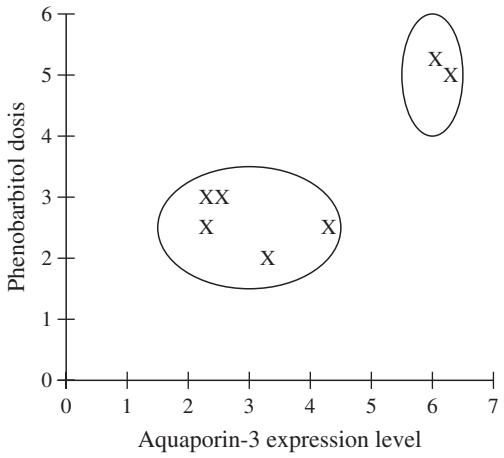

Figure 8: Partition clustering.

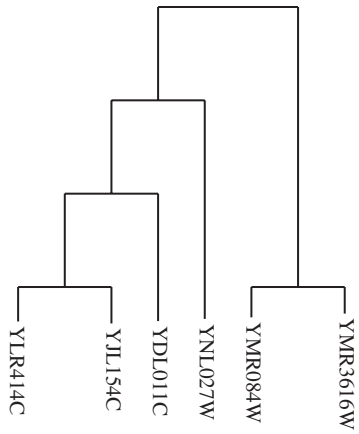

Figure 9: An example of dendrogram produced by hierarchical clustering.

particular level produces a partition into K disjoint groups. If two groups are chosen from different partitions (the result of partitioning at different levels), then either the groups are disjoint or one group wholly contains the other. In hierarchical clustering, there is a measure of the distance or dissimilarity between two merged clusters. The matrix containing the dissimilarity between pair of clusters is called dissimilarity matrix. Examples of dissimilarity measures for the case of continuous variables are Minkowski, Mahalabobis, Lance–Willians and Jeffreys–Matusita. The hierarchical structure is constructed merging the closest two groups.

There are several different algorithms to find a hierarchical tree. An *agglomerative algorithm* begins with N subclusters, each containing a single point, and, at each stage, it merges the two most similar groups to form a new cluster, thus reducing the number of clusters by one. The algorithm proceeds until all the data fall within a single cluster. A *divisive algorithm* operates by successively splitting groups, beginning with a single group and continuing until there are N groups, each of a single individual. Generally, divisive algorithms are computationally inefficient.

The most common measures of distances between clusters are *single-linkage* (the distance between two groups is the distance between their closest members), *complete-linkage* (defined as the distance between the two farthest points), *Ward's hierarchical clustering method* (at each stage of the algorithm, the two groups that produce the smallest increase in the total within-group sum of squares are amalgamated), *centroid distance* (defined as the distance between the cluster means or centroids), *median distance* (distance between the medians of the clusters) and *group average linkage* (average of the dissimilarities between all pairs of individuals, one from each group).

Mixture models

In the mixture method of clustering [126], each different group in the population is assumed to be described by a different probability distribution. The population is described by a *finite mixture distribution* of the form $p(\mathrm{x}) = \sum_{i=1}^{K} \pi_i p(\boldsymbol{x}; \theta_i)$, where π_i are the mixing proportions ($\sum_{i=1}^{K} \pi_i = 1$) and $p(\mathbf{x}; \theta_i)$ is an n-dimensional probability function depending, in each mixture, on a parameter vector θ_i. There are three sets of parameters to estimate: the values of π_i, the components of the vectors θ_i and the value of K, the number of groups in the population.

The usual approach to clustering using finite mixture distributions is, first of all, to specify the form of the component distributions, $p(\mathbf{x}; \theta_i)$. For continuous variables, a usual election is the mixture of normal distributions (each component follows a multivariate normal distribution), while, for mixture of binary variables, the Bernouilli distribution is often chosen. After specifying the form of the component distributions, the number of clusters, K, is prescribed. The parameters of the model are now estimated (this task may be achieved by using the EM algorithm [127]) and the objects are gouped on the basis of their estimated posterior probabilities of group membership. In other words, the object $\mathbf{x}$ is assigned to group i if $\pi_i p(\mathbf{x}; \theta_i) \geq \pi_j p(\mathbf{x}; \theta_j)$ for all $j \neq i$; $j = 1, \ldots, K$.

The main difficulty about the method of mixtures concerns the number of components, K, which in almost all of the approaches should be specified before the remaining parameters can be estimated. Another problem with the mixture model approach is that there are many local minima of the likelihood function and several initial configurations may have to be tried before a satisfactory clustering is produced [128].

Validation

Depending on the specific choice of the preprocessing method, the distance measure, the cluster algorithm and other parameters, different runs of clustering will produce different results. Therefore, it is very important to validate the relevance of the cluster. Validation can be either statistical or biological. Statistical cluster validation can be done by assessing cluster coherence, by examining the predictive power of the clusters or by testing the robustness of a cluster result against the addition of noise. From a biological point of view, it is very hard to choose the best cluster solution if the biological system has not been characterized completely. Sheng *et al.* [129] reviews some of the recent methodologies described in the literature to validate clustering results in bioinformatics.

Clustering in bioinformatics

The main application domain of clustering methods is related to the analysis of microarray data. Based on the assumption that expressional similarity (i.e. co-expression) implies some kind of regulatory or functional similarity of the genes (and vice versa),

the challenge of finding genes that might be involved in the same biological process is thus transformed into the problem of clustering genes into groups based on their similarity in expression profiles.

Following Sheng *et al*. [129], the first generation of clustering algorithms applied to gene expression profiles (*K*-means, hierarchical clustering and self-organizing maps) were mostly developed outside biological research. Although encouraging results have been produced [130, 131], some of the characterstics (such as the determination of the number of clusters, clustering of outliers and computational complexity) often complicate their use for clustering expression data [132].

For this reason, a second generation of clustering algorithms has started to tackle some of the limitations of the earlier methods. These algorithms include, among others, model-based algorithms [133, 134], the self-organizing tree algorithm [135], quality-based algorithms [136]—which produce clusters with a quality guarantee that ensures that all members of a cluster are co-expressed—and biclustering algorithms [137]—they cluster both the genes and the experiments at the same time.

PROBABILISTIC GRAPHICAL MODELS

Probabilistic graphical models represent multivariate joint probability densities via a product of terms, each of which involves only a few variables. The structure of the product is represented by a graph that relates variables that appear in a common term. This graph specifies the product form of the distribution and also provides tools for reasoning about the properties entailed by the product. Although probabilistic graphical models that use undirected graphs—Markov networks [138] and region-based approximations [139, 140]—have been also applied in bioinformatics, in this section we restrict ourselves to the probabilistic graphical models where the corresponding graph is a directed acyclic graph. We consider two types of probabilistic graphical models depending on the status of the random variables. If all the variables are discrete, we name the model a Bayesian network, while in the case of continuous variables—following Gaussian distributions—we will present the so-called Gaussian networks.

More formally, let $\boldsymbol{X} = (X_1, \ldots, X_n)$ be a vector of random variables. A *probabilistic graphical model* for $\boldsymbol{X}$ is a graphical factorization of the joint generalized probability distribution, $\rho(\boldsymbol{X} = \boldsymbol{x})$ (or simply $\rho(\boldsymbol{x})$). The representation consists of two components: a structure and a set of local generalized probability distributions. The structure S for $\boldsymbol{X}$ is a directed acyclic graph (DAG) that represents a set of conditional (in)dependence [141] assertion on the variables in $\boldsymbol{X}$.

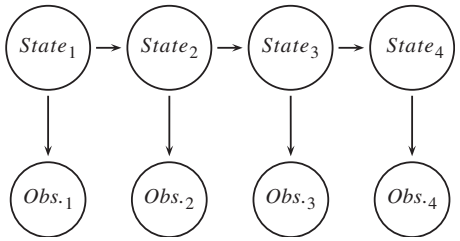

Figure 10: Structure of a hidden Markov model.

The structure S for $\boldsymbol{X}$ represents the assertions that, for all $i = 1, \ldots, n$, $\boldsymbol{X}_i$ and its non-descendants are independent given $\boldsymbol{pa}_i^S$, the node parents of $\boldsymbol{X}_i$ in S. Thus, the factorization is as follows: $\rho(\boldsymbol{x}) = \prod_{i=1}^{n} \rho(x_i | \boldsymbol{pa}_i^S)$. The local generalized probability distributions depend on a finite set of parameters $\theta_S \in \Theta_S$. Thus, we rewrite the previous equation as follows: $\rho(\boldsymbol{x} | \boldsymbol{\theta}_S) = \prod_{i=1}^{n} \rho(x_i | \boldsymbol{pa}_i^S, \boldsymbol{\theta}_i)$ where $\theta_S = (\theta_1, \ldots, \theta_n)$. Taking both components of the probabilistic graphical model into account, the model will be represented by $M = (S, \boldsymbol{\theta}_S)$.

Probabilistic graphical models can be used for supervised classification, for clustering and for representing the relationships between different variables of the domain. In the case of clustering, the variable denoting the group is considered hidden. *Hidden Markov models* [142], a very popular paradigm in bioinformatics, can be seen as an instantiation of probabilistic graphical models (Figure 10). In order to represent molecular networks by using probabilistic graphical models, we associate each molecular entity with a random variable. The values of this random variable are determined by the possible levels of the molecular entity. These types of stochastic models have proved to be very adequate to represent, for instance, the regulation between genes.

Bayesian networks

Bayesian networks have been surrounded by a growing interest in recent years, as shown by the large number of dedicated books and the wide range of theoretical and practical publications in this field.

Textbooks include the classic Pearl [143]. Lauritzen [144] provides a mathematical analysis of graphical models and, more recently, Cowell *et al*. [145], Jensen [146] and Neapolitan [147] are excellent compilations of material covering recent advances in the field.

The Bayesian network paradigm is mainly used to reason in domains with an intrinsic uncertainty. Bayesian networks are used to model relationships between variables. There are situations where the value of some of the variables of the system are known (this is called evidence) and we can be interested in knowing how this evidence affects the probability distribution of the rest of the variables of the system. This type of reasoning is done by means of the propagation of the evidence through the Bayesian network, and this can be proved [148] to be an NP-hard risk in the general case of multiply connected Bayesian networks.

Once the Bayesian network is built, it constitutes an efficient device to perform probabilistic inference. Nevertheless, the problem of building such a network remains. The structure and conditional probabilities necessary to characterize the Bayesian network can be provided either externally by experts—time consuming and subject to mistakes—or by automatic learning from a database of cases. On the other hand, the learning task can be separated into two subtasks: *structure learning*, that is, to identify the topology of the Bayesian network, and *parametric learning*, the numerical parameters (conditional probabilities) for a given network topology.

There are two main ways [149] to learn Bayesian networks from data. One of them is by detecting conditional (in)dependencies of triplets of variables using hypothesis testing. The other is the so-called score + search method, explained subsequently.

Every algorithm that tries to recover the structure of a Bayesian network by *detecting (in)dependencies* has some conditional (in)dependence relations between some subset of variables of the model as input, and a directed acyclic graph that represents a large percentage (and even all of them if possible) of these relations as output. Once the structure has been learnt, the conditional probability distributions required to completely specify the model are estimated from the database—using some of the different approaches to parameter learning—or are given by an expert.

Although the approach to model elicitation based on detecting conditional (in)dependencies is quite appealing, due to its closeness to the semantics of Bayesian networks, a large percentage of structure learning algorithms developed belongs to the category of *score + search methods*. To use this learning approach, we need to define a metric that measures the goodness of every candidate Bayesian network with respect to a datafile of cases. In addition, we also need a procedure to move intelligently through the space of possible networks. In most of the score + search approaches, the search is performed in the space of directed acyclic graphs that represent feasible Bayesian network structures. Other possibilities include searching in the space of equivalence classes of Bayesian networks [150] or in the space of orderings of the variables [151]. The problem of finding the best network according to some criterion from the set of all networks in which each node has no more than K parents ($K > 1$) is NP-hard [152]. This result gives a good opportunity to use different heuristic search algorithms. These heuristic search methods can be more efficient when the model selection criterion is separable, that is, when the model selection criterion can be written as a product (or a sum) of variable-specific criteria. Among all heuristic search strategies used to find good models in the space of Bayesian network structures, we have different alternatives: greedy search, simulated annealing, tabu search, genetic algorithms, evolutionary programming, estimation of distribution algorithm, etc. Scoring metrics that have been used in the learning of Bayesian networks from data are penalized maximum likelihood, Bayesian scores (like marginal likelihood) and scores based on information theory.

Gaussian networks

Another particular case of probabilistic graphical models is when each univariate variable X_i is continuous and each local density function is the linear-regression model $f(x_i|\boldsymbol{pa}_i^S, \theta_i) \equiv \mathcal{N}(x_i; m_i + \sum_{x_j \in \boldsymbol{pa}_i} b_{ji}(x_j - m_j), v_i)$ where $\mathcal{N}(x; \mu, \sigma^2)$ is a univariate normal distribution with mean μ and variance σ^2. A probabilistic graphical model constructed with these local density functions is called a Gaussian network [153].

The main difficulty when working with multivariate normal distributions is to assure that the assessed covariance matrix is positive-definite. However, with the Gaussian network representation it is not necessary to be aware of this constraint. Therefore, Gaussian networks are more suitable

for model elicitation and understanding than the standard representation of multivariate normal distributions.

As in the case of Bayesian networks, there are different approaches to induce Gaussian networks from data. The most usual ones are based on edge exclusion tests [154], penalized maximum liklihood metric and Bayesian scores [155].

Probabilistic graphical models in bioinformatics

Genomics

The main application of probabilistic graphical models in genomics is the modelling of DNA sequences. In Meyer and Durbin [156], hidden Markov models are used in the gene finding process and, in Cawley and Pachter [157] in the alternative splicing detection. In Won *et al*. [4], genetic algorithms are used in the training of hidden Markov models to identify promoter and coding regions. Bayesian networks are used in splice site prediction in [158]. Gene modelling is not the only application of probabilistic graphical models. For instance, in Greenspan and Geiger [159], Bayesian networks are used when modelling haplotype blocks and, later on, these models are used in linkage disequilibrium mapping. Bockhorst *et al*. [3] show an example of the application of Bayesian networks in operon prediction.

Proteomics

Bayesian networks have been used for the prediction of protein contact maps [160] and for the protein fold recognition and superfamily classification problem [161].

Microarray

An example of the application of Bayesian networks to expression pattern recognition in microarray data can be found in Friedman *et al*. [162].

Systems biology

One of the most important applications of the probabilistic graphical models is the inference of genetic networks [163].

Some *advantages* of using this paradigm to model genetic networks are as follows. They are based on probability theory, a scientific discipline with sound mathematical development. Probability theory could be used as a framework to deal with the uncertainty and noise underlying biological domains. The graphical component of these models—the structure—allows the representation of the interrelations between the genes—variables—in an interpretable way. The conditional independence between triplets of variables gives a clear semantic. The quantitative part of the models—the conditional probabilities—allows the strength of the interdependencies between the variables to be established. Inference algorithms—–exact and approximate—developed in these models enable different types of reasoning inside the model. Already there are algorithms that search for probabilistic graphical models from observational data based on well-understood principles at statistics. These algorithms make it possible to include hidden variables which are not observable in reality. It is also achievable to combine multiple local models into a joint global model. The declarative nature of the probabilistic graphical models is an advantage to the modelling process by taking additional aspects into account, such as the existence of some edges in the model based on previous knowledge. The models are biologically interpretable and can be rigorously scored against observational data.

However, not all the characteristics of probabilistic graphical models are appropriate for this task. A *disadvantage* is that very few work has been done in the development of learning algorithms able to represent causality between variables [164]. The description of casual connections among gene expression rates is a matter of special importance to obtain biological insight about the underlying mechanisms in the cell. Furthermore, the features of the analysed databases with very few cases, in the order of dozens, and a very large number of variables, in the order of thousands, make it necessary to adapt the learning algorithms developed. This way, learning algorithms that are able to carry-out the modelling of subnetworks and, at the same time, provide robustness in the graphical structure obtained should be of interest [165]. Finally, the inclusion of hidden variables—where and how many—is a difficult problem when learning probabilistic graphical models from data.

Static and dynamic probabilistic graphical models have been suggested in the literature to reconstruct gene expression networks from microarray data.

An introduction to the problem can be found in Husmeier [166]. There are several works that use *static Bayesian networks* to model genetic networks [162, 165–176], In Tamada et al. [177] DNA sequence information is mixed with microarray

data in the Bayesian network in order to obtain a more accurate estimation of the network when the number of microarray data is limited. In Nariai *et al.* [178] genetic networks estimated from expression data are refined using protein–protein interactions. Imoto *et al.* [179] propose a new method to measure the reliability of inferred genetic networks based on bootstrap. In De Hoon *et al.* [180], sequence information is combined with expression data to improve gene regulation prediction. Husmeier [181] tests the viability of the Bayesian network paradigm for gene network modelling. *Static Gaussian networks* have also been proposed to infer genetic regulatory networks [138, 182–184].

Dynamic Bayesian networks are able to show how genes regulate each other across time in the complex workings of regulatory pathways. The analysis of time–series data potentially allows us to determine regulatory pathways across time, rather than merely associating genes that are regulated together. Different works have considered the use of dynamic Bayesian networks to infer regulatory pathways [185–190].

In Steffen *et al.* [191], clustering methods applied to microarray data and protein–protein interaction data are combined in the construction of a signal transduction network.

OPTIMIZATION

Many problems in bioinformatics can be posed as the task of finding an optimal solution in a space of multiple (sometimes exponentially sized) possible solutions. The choice of the optimization method to be used is crucial for the problem solution. In this section, we describe a number of optimization algorithms developed by the machine learning community, and review their application to problems of bioinformatics.

In our analysis, we will not consider a number of classic optimization and heuristic methods that, although widely employed for the solution of biological problems, are not relevant from the machine learning point of view. These methods include hill climbing, greedy heuristics, dynamic and integer programming and branch and bound methods. However, in the section which reviews optimization applications to bioinformatics, we include, as a way to illustrate different alternatives to the problems treated, references to the use of these classic optimization methods.

Optimization approaches to bioinformatics problems can be classified, according to the type of solutions found, into *exact* and *approximate* methods. Exact methods output the exact solutions when convergence is achieved. However, they do not necessarily converge at every instance. Approximate algorithms always output a candidate solution, but not necessarily the optimal one.

Exact optimization methods

Common exact optimization approaches include exhaustive search methods. However, these algorithms are feasible only for small search domains and are not relevant to our review. Some methods are able to use knowledge about the problem to reduce the search space. This can be done by enforcing some constraints which the optimal solution has to fulfill [192].

Approximate optimization methods

Approximate algorithms can be further classified into *deterministic* and *stochastic* according to the way solutions are found. Given a set of input parameters, a deterministic method will converge to the same solution. Stochastic methods use a random component that may cause them to obtain different solutions when running with the same input parameters.

Stochastic algorithms can be divided into local and population-based search methods. *Local search algorithms* visit one point of the search space at each iteration. *Population-based* search methods use a set or population of points instead of a single point. Examples of local search methods are Monte Carlo-based search, simulated annealing and tabu search.

When used in the optimization framework, the *Monte Carlo algorithm* [193] associates a probability distribution with each point of the search space based on the objective function. Markov chain Monte Carlo produces a Markov chain of conformations which, for a sufficiently large number of iterations, approximates the canonical distribution. The configurations obtained by the method are samples from the search space and can be combined with energy minimization to find the optimal solution.

Simulated annealing [194] is inspired by the annealing process that arises in physics. It uses transition probabilities based on a Boltzmann distribution and a non-increasing function, called

the cooling schedule, to tune the search for the optimal solutions. *Tabu search* [195] allows local search heuristic algorithms to escape from local minima where the algorithm cannot find any further solution in the neighbourhood that improves the objective function value. The overall approach is to avoid entering cycles by forbidding or penalising the moves that the algorithm takes in the next iteration to points in the solution space previously visited.

Evolutionary algorithms are among the best-known population-based search methods. They start from a random population of points and iterate until some pre-defined stopping criterion is satisfied. At every iteration, usually called generation, a subset of points is selected. By applying some variation operators to the selected set, a new population is created. An example of evolutionary algorithms are *genetic algorithms* (GAs) [196]. The distinguishing feature of GAs is the application of the recombination and mutation operators.

Another evolutionary algorithm used for the solution of bioinformatic problems is *genetic programming* [197], employed in order to evolve a program code able to solve a given problem. Another class of population-based search methods comprises those algorithms that use probabilistic modelling of the solutions instead of genetic operators. *Estimation of distribution algorithms* (EDAs) [198] are evolutionary algorithms that construct an explicit probability model of a set of selected solutions. This model can capture, by means of probabilistic dependencies, relevant interactions among the variables of the problem. The model can be conveniently used to generate new promising solutions.

Optimization in bioinformatics

Genomics

Several optimization algorithms have been proposed to solve the multiple sequence alignment problem [199] (Figure 11). These include tabu search [200], Monte Carlo optimizaton [201], methods based on genetic algorithms [202], relaxation methods [203], simulated annealing [204], iterative algorithms [205] and parallel simulated annealing [206].

The prediction of promoters from DNA sequences has been achieved using GAs together with neural networks [207]. A fuzzy guided GA [208] has been applied to recover the operon structure of the prokaryotic genome. Evolved neural networks have also shown good results for the task of discriminating functional elements

```
TGGAGACCAC CGTGAACGCC CATCA − − − GG TCT T GCCCAA
TGGAGACCAC CGTGAACGCC CATCA − − A AG TCT   GCCCAA
TGGAGACCAC CGTGAACGCC GCCCA TCT AT TCT T GCCCAA
TGGAGACCAC CGTGAACGCC CACCA − − −  AT TCT T GCCCAA
TGGAGACCAC CGTGAACGCC CATCA − − − CG TCC T GCCCAA
```

Figure 11: Multiple DNA sequence alignment.

associated with coding nucleotides from non-coding sequences of DNA [209]. Optimization of neural network architecture using genetic programming has improved the detection and modelling of gene–gene interactions in studies of human diseases [210]. Moreover, estimation of distribution algorithms have been applied to splice site prediction [88, 211] and gene selection [212].

DNA sequencing has been approached using tabu search [213], GAs [214] and greedy algorithms [215]. Tabu search has been also recently employed to determine sequences of aminoacids in long polypeptides [216] and to extract motifs from DNA sequences [217].

The physical mapping of chromosomes has been treated with branch and bound optimization methods [218], Monte Carlo algorithms [219], greedy techniques [220] and parallel GAs [221]. The identification of a consensus sequence on DNA sequences has been approached using linear programming techniques [222] and simulated annealing [223]. Haplotype block partitioning and tag SNP selection have been treated using dynamic programming algorithms.

The reconstruction of amino acid sequences using only spectral features has been solved using dynamic programming [224]. Dynamic programming [225] is also the choice preferred for RNA secondary structure prediction which can, in general, be handled with polynomial algorithms. Evolutionary algorithms have also been used to discover RNA structural elements [226]. RNA tertiary structure determination has been approached with tabu search [227].

Proteomics

Several optimization approaches have been used for protein folding in simplified models. These include: tabu search [228, 229], Monte Carlo methods [230–232], GAs [233–235] and EDAs [236].

Protein side-chain prediction, an important problem for homology-based protein structure prediction and protein design, has been approached using dead-end elimination algorithms [192, 237], GAs [238–240] and other population-based search

methods [241]. Simulated annealing [242], optimization methods based on inference from graphical models [243] and the self-consistent mean field approach [244] have also been employed to solve this problem.

Simulated annealing has been used in the modelling of loops in protein structures [245] and genetic programming has been employed for contact map prediction in proteins [246].

Systems biology

There are several applications of genetic programming to the inference of gene networks [247–249] and metabolic pathways from observed data [250]. The identification of transcription factor binding sites has been treated using Markov chain optimization [251].

GAs have been applied to model genetic networks [252], select regulatory structures [253] and estimate the parameter of bioprocesses [254]. Inference of genetic networks has been achieved using other evolutionary algorithms [255, 256].

Microarray

Simulated annealing has been recently applied [257] to the design of dual channel microarray studies. It has also been employed to align experimental transcription profiles to a set of reference experiments [258], biclustering of expression data [259] and in the analysis of temporal gene expression profiles [260].

Evolutionary algorithms have been employed to cluster microarray data [261]. GAs have also been applied in the normalization of gene expression data [262], a necessary step before quantizing gene expression data into the binary domain. The multiclass prediction of gene expression data has been accomplished using GAs [60]. *k*-nearest neighbour genetic hybrid methods [59] have been applied to gene expression data analysis.

Evolution and other applications

Inference of the best phylogenetic tree that fits the data has been approached using different optimization methods. Exhaustive searchers have been used when the dimension of the search space is small. Branch and bound and other heuristic techniques have been applied in other cases [263].

Greedy algorithms [264, 265], hill climbing methods [266] and simulated annealing [267] have been used given their simple and fast implementations. Haplotype reconstruction has been approached using both exact and approximate methods [268]. Small size problems have been solved using branch and bound techniques, while more complex instances have been solved using GAs [268].

Genetic algorithms have been used in the optimization of linkage disequilibrium studies, minimizing the genotyping burden [269], the back-transition of a protein sequence into a nucleid acid sequence [270] and in primer design [271]. Evolutionary algorithms have been also employed to improve the fractal visualization of sequence data [272].

CONCLUSIONS

Nowadays, one of the most challenging problems in computational biology is to transform the huge volume of data, provided by newly developed technologies, into knowledge. Machine learning has become an important tool to carry out this transformation.

This article introduces some of the most useful techniques for modelling—Bayesian classifiers, logistic regression, discriminant analysis, classification trees, nearest neighbour, neural networks, support vector machines, ensembles of classifiers, partitional clustering, hierarchical clustering, mixture models, hidden Markov models, Bayesian networks and Gaussian networks—and optimization—Monte Carlo algorithms, simulated annealing, tabu search, GAs, genetic programming and estimation of distribution algorithms—giving some pointers to the most relevant applications of the former techniques in bioinformatics. The article can serve as a gateway to some of the most representative works in the field and as an insightful categorization and classification of the machine learning methods in bioinformatics.

Key Points

- Supervised classification, clustering and probabilistic graphical models for bioinformatics are reviewed.
- A review of deterministic and stochastic heuristics for optimization in the same domain is presented.
- Applications in genomics, proteomics, systems biology, evolution and text mining are also shown.

Acknowledgements

The authors are grateful to the anonymous reviewers for their comments, which have helped us to greatly improve this article. This work was partly supported by the University of the Basque Country, the Basque Government and the Ministry of Education and Science under grants 9/UPV 00140.226-15334/2003, SAIOTEK S-PE04UN25, ETORTEK-GEN MODIS, ETORTEK-BIOLAN and TIN2005-03824.

References

1. Mathé C, Sagot M.-F, Schlex T, *et al*. Current methods of gene prediction, their strengths and weaknesses. *Nucleic Acids Research* 2002;**30**(19):4103–17.
2. Stein Aerts, Peter Van Loo, Yves Moreau, *et al*. A genetic algorithm for the detection of new cis-regulatory modules in sets of coregulated genes. *Bioinformatics* 2004;**20**(12): 1974–76.
3. Bockhorst J, Craven M, Page D, *et al*. A Bayesian network approach to operon prediction. *Bioinformatics* 2003;**19**(10): 1227–35.
4. Won K.-J, Prügel-Bennet A, Krogh A. Training HMM structure with genetic algorithm for biological sequence analysis. *Bioinformatics* 2004;**20**(18):3613–19.
5. Carter RJ, Dubchak I, Holbrook SR. A computational approach to identify genes for functional RNAs in genomic sequence. *Nucleic Acids Research* 2001;**29**(19): 3928–38.
6. Bower JM, Bolouri H (eds). *Computational Modeling of Genetic and Biochemical Networks*. MIT Press, March 2004.
7. Baldi P, Brunak S. *Bioinformatics. The Machine Learning Approach*. MIT Press, 2001.
8. Krallinger M, Erhardt RA, Valencia A. Text-mining approaches in molecular biology and biomedicine. *Drug Discovery Today* 2005;**10**(6):439–45.
9. Ananiadou S, McNaught J (eds). *Text Mining for Biology and Biomedicine*. Artech House Publishers, January 2006.
10. Devroye L, Györfi L, Lugosi G. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.
11. Duda R, Hart P, Stork DG. *Pattern Classification*. Wiley, 2001.
12. Fukunaga K. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.
13. Hastie T, Tibshirani R, Friedman J. *The Elements of Statistical Learning*. Springer-Verlag, 2001.
14. Mitchell TM. *Machine Learning*. McGraw-Hill, 1997.
15. Webb A. *Statistical Pattern Recognition*. Wiley, 2002.
16. Durbin R, Eddy SR, Krogh A, *et al*. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.
17. Gary B. Fogel, David W. Corne. *Evolutionary Computation in Bioinformatics*. Morgan Kaufmann, 2002.
18. Frasconi P, Shamir R (eds). *Artificial Intelligence and Heuristic Methods in Bioinformatics*, Volume 183, NATO Science Series: Computer and Systems Sciences Edited. NATO, 2003.
19. Higgins D, Taylor W (eds). *Bioinformatics. Sequence, Structure, and Databanks*. Oxford University Press, 2000.
20. Husmeier D, Dybowski R, Roberts S (eds). *Probabilistic Modeling in Bioinformatics and Medical Informatics*. Springer Verlag, 2005.
21. Jagota A. *Data Analysis and Classification for Bioinformatics*. Bioinformatics by the Bay Press, 2000.
22. Jiang T, Xu X, Zhang MQ (eds). *Current Topics in Computational Molecular Biology*. The MIT Press, 2002.
23. Pevzner PA. *Computational Molecular Biology. An Algorithmic Approach*. MIT Press, 2000.
24. Schölkopf B, Tsuda K, Vert J.-P (eds). *Kernel Methods in Computational Biology,* . The MIT Press, 2004.
25. Seiffert U, Jain LC, Schweizer P (eds). *Bioinformatics Using Computational Intelligence Paradigms*. Springer Verlag, 2005.
26. Wang JTL, Zaki MJ, Toivonen HTT, *et al*. (eds). *Data Mining in Bioinformatics*. Springer-Verlag, 2004.
27. Wu CH, McLarty JW. *Neural Networks and Genome Identification*. Elsevier, 2000.
28. Larrañaga P, Menasalvas E, Peña JM, *et al*. Special issue in data mining in genomics and proteomics. *Artificial Intelligence in Medicine* 2003;**31**:III–IV.
29. Li J, Wong L, Yang Q. Special issue on data mining for bioinformatics. *IEEE Intelligent Systems* 2005;**20**(6).
30. Ling CX, Noble WS, Yang Q. Special issue: Machine learning for bioinformatics-part 1. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2005; **2**(2):81–2.
31. Green DM, Swets JA. *Signal Detection Theory and Psychophysics*. Wiley, 1974.
32. Bradley AP. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition* 1997;**30**(7):1145–59.
33. Stone M. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society Series B* 1974;**36**:111–47.
34. Efron B. Bootstrap methods: another look at the jacknife. *Annals of Statistics* 1979;**7**:1–26.
35. Efron B. Estimating the error rate of a prediction rule: Improvement on cross-validation. *J Am Statistical Association* 1983;**78**:316–31.
36. Baldi P, Brunak S, Chauvin Y, *et al*. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics* 2000;**16**:412–24.
37. Michiels S, Koscielny S, Hill C. Prediction of cancer outcome with microarrays: A multiple random validation strategy. *Lancet* 2005;**365**:488–92.
38. Ambroise C, MacLachlan GJ. Selection bias in gene extraction on the basis of microarray gene-expression data. *PNAS* 2002;**99**(10):6562–6.
39. Braga-Neto UM, Dougherty ER. Is cross-validation valid for small-sample microarray classification?. *Bioinformatics* 2004;**20**(3):374–80.
40. Sima C, Braga-Neto U, Dougherty ER. Superior featureset ranking for small samples using bolstered error classification. *Bioinformatics* 2005;**21**(7):1046–54.
41. Fu WJ, Carroll RJ, Wang S. Estimating misclassification error with small samples via bootstrap cross-validation. *Bioinformatics* 2005;**21**:1979–1986.
42. McNemar Q. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* 1947;**12**:153–7.
43. Alpaydin E. Combining 5×2 cv F-test for comparing supervised classification learning algorithms. *Neural Computation* 1999;**11**:1885–92.
44. Dietterich TG. Approximate statistical tests for comparing supervised classification algorithms. *Neural Computation* 1998;**10**:1895–1923.
45. Nadeau C, Bengio Y. Inference for the generalization error. *Machine Learning* 2003;**52**(3):239–81.
46. Liu H, Motoda H. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic, 1998.

47. Narendra P, Fukunaga K. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computation, C* 1977;**26**(9):917–22.
48. Kittler J. Feature set search algorithms. *Pattern Recognition and Signal Processing*. Sijthoff and Noordhoff, 1978: pp. 41–60.
49. Pudil P, Novovicova J, Kittler J. Floating search methods in feature selection. *Pattern Recognition Letters* 1994;**15**(1): 1119–25.
50. Kohavi R, John G. Wrappers for feature subset selection. *Artificial Intelligence* 1997;**97**(1–2):273–324.
51. Kuncheva L. Genetic algorithms for feature selection for parallel classifiers. *Information Processing Letters* 1993;**46**: 163–8.
52. Inza I, Larrañaga P, Etxeberria R, *et al*. Feature subset selection by Bayesian network-based optimization. *Artificial Intelligence* 2000;**123**:157–84.
53. Ben-Bassat M. Pattern recognition and reduction of dimensionality. *Handbook of Statistics–II*. North-Holland, 1982: pp. 773–91.
54. Pan W. A comparative review of statistical methods for discovering differentially expressed genes in replicated microarray experiments. *Bioinformatics* 2002;**18**(4):546–54.
55. Troyanskata OG, Garber ME, Brown PO, *et al*. Nonparametric methods for identifying differentially expressed genes in microarray data. *Bioinformatics* 2002; **18**(11):1454–61.
56. Wang Y, Tetko IV, Hall MA, *et al*. Gene selection from microarray data for cancer classification–a machine learning approach. *Computational Biology and Chemistry* 2004;**29**: 37–46.
57. Inza I, Sierra B, Blanco R, *et al*. Gene selection by sequential search wrapper approaches in microarray cancer class prediction. *J Intelligent and Fuzzy Systems* 2002;**12**(1):25–34.
58. Jarvis RM, Goodacre R. Genetic algorithm optimization for preprocessing and variable selection of spectroscopic data. *Bioinformatics* 2005;**21**(7):860–68.
59. Li L, Weinberg CR, Darden TA, *et al*. Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method. *Bioinformatics* 2001;**17**(12):1131–42.
60. Ooi CH, Tan P. Genetic algorithms applied to multi–class prediction for the analysis of gene expression data. *Bioinformatics* 2003;**19**(1):37–44.
61. Inza I, Larrañaga P, Blanco R, *et al*. Filter versus wrapper gene selection approaches in DNA microarray domains. *Artificial Intelligence in Medicine* 2004;**31**(2):91–103.
62. Xing EP, Jordan MI, Karp RM. Feature selection for highdimensional genomic microarray data. In: *Proceedings of the Eighteenth International Conference in Machine Learning*. ICML, 2001: pp. 601–8.
63. Wolpert DH, Macready WG. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1997;**1**(1):67–82.
64. Duda RO, Hart P. *Pattern Classification and Scene Analysis*. Jon Wiley and Sons, 1973.
65. Minsky M. Steps toward artificial intelligence. *Transactions on Institute of Radio Engineers* 1961;**49**:8–30.
66. Pazzani MJ. Searching for dependencies in Bayesian classifiers. In: Fisher D, Lenz H (eds). *Artificial Intelligence and Statistics IV, Lecture Notes in Statistics*. Springer-Verlag, 1997.
67. Friedman N, Geiger D, Goldszmidt M. Bayesian network classifiers. *Machine Learning* 1997;**29**(2):131–64.
68. Chow C, Liu C. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory* 1968;**14**:462–7.
69. Sahami M. Learning limited dependence Bayesian classifiers. In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining* 1996: pp. 335–8.
70. Kleinbaum DG, Kupper LL, Chambless LE. Logistic regression analysis of epidemiologic data: theory and practice. *Communications on Statistics* 1982;**11**(5):485–547.
71. Fisher RA. The use of multiple measurements in taxonomic problems. *Annals of Eugenics* 1936;**7**:179–88.
72. McLachlan GJ. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, 1992.
73. Breiman L, Friedman JH, Olshen RA, *et al*. *Classification and Regression Trees*. Chapman and Hall, 1993.
74. Quinlan R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
75. Fix E, Hodges JL. Discriminatory analysis: nonparametric discrimination: consistency properties. *USAF School of Aviation Medicine* 1951;**4**:261–79.
76. McCulloch WS, Pitts W. A logical calculus of ideas imminet in nervous activity. *Bulletin of Mathematical Biophysics* 1943;**5**:115–33.
77. Rosenblatt F. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, 1962.
78. Rumelhart DE, Hinton GE, Williams RJ. Learning internal representations by backpropagation errors. *Nature* 1986; **323**(99):533–6.
79. Vapnik V. *The Nature of Statistical Learning*. Springer-Verlag, 1995.
80. Schölkopf B, Burges CJC, Smola AJ (eds). *Advances in Kernel Methods: Support Vector Learning*. MIT Press, 1999.
81. Kuncheva LI. Combining pattern classifiers. *Methods and Algorithms*. John Wiley and Sons, 2004.
82. Wolpert DH. Stacked generalization. *Neural Netorks* 1992; **5**(2):241–60.
83. Breiman L. Bagging predictors. *Machine Learning* 1996; **26**(2):123–40.
84. Breiman L. Random forests. *Machine Learning* 2001;**45**: 5–32.
85. Freund Y, Schapire R. A decision-theoretic generalization of on-line learning and an application to boosting. *J Comp and System Sciences* 1997;**55**(1):119–39.
86. Salzberg S. Localing protein coding regions in human DNA using a decision tree algorithm. *J Comput Biol* 1995;**2**: 473–85.
87. Castelo R, Guigó R. Splice site identification by *idl*BNs. *Bioinformatics* 2004;**20**(Suppl. 1):i69–76.
88. Yvan Saeys, Sven Degroeve, Dirk Aeyels, *et al*. Feature selection for splice site prediction: a new method using EDA-based feature ranking. *BMC Bioinformatics* 2004;**5**:64.
89. Degroeve S, De Baets B, Van de Peer Y, *et al*. Feature subset selection for splice site prediction. *Bioinformatics* 2002; **18**(Suppl. 2):S75–83.
90. Allen JE, Pertea M, Salzberg SL. Computational gene prediction using multiple source of evidence. *Genome Research* 2004;**14**:142–8.

91. Pavlovic V, Garg A, Kasif S. A Bayesian framework for combining gene predictions. *Bioinformatics* 2002;**18**(1):19–27.
92. López-Bigas N, Ouzounis CA. Genome-wide identification of genes likely to be involved in human genetic disease. *Nucleic Acids Research* 2004;**32**(10):3108–14.
93. Bao L, Cui Y. Prediction of the phenotypic effects of nonsynonymous single nucleotide polymorphisms using structural and evolutionary information. *Bioinformatics* 2005;**21**(5):2185–90.
94. Sebban M, Mokrousov I, Rastogi N, *et al*. A data–mining approach to spacer oligonucleotide typing of mycobacterium tuberculosis. *Bioinformatics* 2002;**18**(2):235–43.
95. Kim S. Protein beta-turn prediction using nearest-neighbor method. *Bioinformatics* 2004;**20**(1):40–4.
96. Salamov AA, Solovyev VV. Prediction of protein secondary structure by combining nearest-neighbor algorithms and multiple sequence alignments. *Journal of Molecular Biology* 1995;**247**:11–15.
97. Yi T.-M, Lander ES. Protein secondary structure prediction using nearest-neighbor methods. *J Mol Biol* 1993;**232**:1117–29.
98. Selbig J, Mevissen T, Lengauer T. Decision tree-based formation of consensus protein secondary structure prediction. *Bioinformatics* 1999;**15**(12):1039–46.
99. Yang C, Dobbs D, Honavar V. A two-stage classifier for identification of protein-protein interface residues. *Bioinformatics* 2004;**20**:i371–8.
100. Huang Y, Li Y. Prediction of protein subcellular locations using fuzzy *k*-NN mathos. *Bioinformatics* 2004; **20**(1):21–8.
101. Valafar F. Pattern recognition techniques in microarray data analysis: a survey. *Annals of the New York Academy of Sciences* 2002;**980**:41–64.
102. Krishnapuram B, Carin L, Hartemink AJ. Joint classifier and feature optimization for comprehensive cancer diagnosis using gene expression data. *J Comput Biol* 2004; **11**(2–3):227–42.
103. Olshen AB, Jain AN. Deriving quantitative conclusions from microarray data. *Bionformatics* 2002;**18**(7):961–70.
104. Tan AC, Gilbert D. Ensemble machine learning on gene expression data for cancer classification. *Applied Bioinformatics* 2002;**2**(3)S:75–83.
105. Dudoit S, Fridlyand J, Speed P. Comparison of discrimination methods for classification of tumors using gene expression data. *J Am Statistical Association* 2002;**97**:77–87.
106. Ramaswamy S, Yeang CH, Tamayo P, *et al*. Molecular classification of multiple tumor types. *Bioinformatics* 2001;**1**: S316–S322.
107. Statnikov A, Aliferis CF, Tsamardinos I, *et al*. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics* 2005;**21**(5):631–43.
108. Lee JW, Lee Bok J, Park M, *et al*. An extensive comparison of recent classification tools applied to microarray data. *Computational Statistics and Data Analysis* 2005;**48**:869–85.
109. Ben-Dor A, Bruhn L, Friedman N, *et al*. Tissue classification with gene expression profiles. *Journal of Computational Biology* 2000;**7**(3–4):559–84.
110. Brown MPS, Grundy WN, Lin D, *et al*. Knowledge-based analysis of microarray gene expression data by using support vector machines. *J Comput Biol* 2004;**11**(2–3):227–42.
111. Kim K.-J, Cho S.-B. Prediction of colon cancer using an evolutionary neural network. *Neurocomputing* 2004;**61**: 361–79.
112. Hautaniemi S, Kharait S, Iwabu A, *et al*. Modeling of signal-response cascades using decision tree analysis. *Bioinformatics* 2005;**21**:2027–2035.
113. Middendorf M, Kundaje A, Wiggins C, *et al*. Predicting genetic regulatory response using classification. *Bioinformatics* 2004;**20**:i232–40.
114. Zhou GD, Shen D, Zhang J, *et al*. Recognition of protein/gene names from text using an ensemble of classifiers. *BMC Bioinformatics* 2005;**6**(Suppl. 1):S7.
115. Stapley BJ, Kelley LA, Sternberg MJ. Predicting the subcellular location of proteins from text using support vector machines. In: *Proceedings of the 7th Pacific Symposium on Biocomputing* 2002: pp. 374–85.
116. Wu B, Abbott T, Fishman D, *et al*. Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data. *Bioinformatics* 2003;**19**(13):1636–43.
117. Baumgartner C, Böhm C, Baumgartner D, *et al*. Supervised machine learning techniques for the classification of metabolic disorders in newborns. *Bioinformatics* 2004;**20**(17):2985–96.
118. Li L, Umbach DM, Terry P, *et al*. Application of the GA/KNN methodh to SELDI proteomics data. *Bioinformatics* 2004;**20**(10):1638–40.
119. Satten GA, Datta S, Moura H, *et al*. Standardization and denoising algorithms for mass spectra to classify whole-organism bacterial specimens. *Bioinformatics* 2004;**20**(17): 3128–36.
120. Jung H.-Y, Cho H.-G. An automatic block and spot indexing with k-nearest neighbors graph for microarray image analysis. *Bioinformatics* 2002;**18**(Suppl. 2): S141–51.
121. Perner P, Perner H, Müller B. Mining knowledge for HEp-2 cell image classification. *Artificial Intelligence in Medicine* 2002;**26**:161–73.
122. Forgy E. Cluster analysis for multivariate data: efficiency vs. interpretability of classifications (abstract). *Biometrics* 1965;**21**:768–9.
123. Gersho A, Gray RM. *Vector Quantization and Signal Compression*. Kluwer Academic, 1992.
124. Linde Y, Buzo A, Gray RM. An algorithm for vector quantizer design. *IEEE Transactions on Communications* 1980; **28**(1):84–95.
125. Jardine N, Sibson R. *Mathematical Taxonomy*. Wiley, 1971.
126. McLachlan GJ, Basford K. *Mixture Models: Inference and Application to Clustering*. Dekker, 1988.
127. Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the EM algorithm. *J Royal Statistical Society Series B* 1977;**39**:1–38.
128. Böhning D, Seidel W. Recent developments in mixture models. *Computational Statistics and Data Analysis* 2003;**41**: 349–57.
129. Sheng Q, Moreau Y, De Smet F, *et al*. Advances in cluster analysis of microarray data. *Data Analysis and Visualization in Genomics and Proteomics*. John Wiley and Sons, 2005: pp. 153–73.
130. Spellman PT, Sherlock G, Zhang MQ, *et al*. Comprehensive identification of cell cycleregulated genes of the yeast *saccharomyces cerevisiase* by microarray hybridization. *Molecular Biology Cell* 1998;**9**:3271–97.

131. Tamayo P, Slonim D, Mesirov J, *et al*. Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. In: *Proceedings of the National Academic of Sciences USA* 1999;**96**:2907–12.
132. Sherlock G. Analysis of large-scale gene expression data. *Briefings in Bioinformatics* 2001;**2**(4):350–62.
133. McLachlan GJ, Bean RW, Peel D. A mixture model-based approach to the clustering of microarray data: from expression to regulation. *Proceedings of the IEEE* 2002; **90**(11):1722–43.
134. Yeung K, Haynor D, Ruzzo W. Validating clustering for gene expression data. *Bioinformatics* 2001;**17**(4):309–18.
135. Herrero J, Valencia A, Dopazo J. A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics* 2001;**17**(2):126–36.
136. De Smet F, Mathys J, Marchal K, *et al*. Adaptive quality-based clustering of gene expression profiles. *Bioinformatics* 2002;**20**(5):660–7.
137. Sheng Q, Moreau Y, De Moor B. Biclustering microarray data by Gibbs sampling. *Bioinformatics* 2003;**19**(Suppl. 2): II196–205.
138. Schäfer J, Strimmer K. An empirical Bayes approach to inferring large-scale gene association networks. *Bioinformatics* 2005;**21**(6):754–64.
139. Jojic V, Jojic N, Meek C, *et al*. Efficient approximations for learning phylogenetic HMM models from data. *Bioinformatics* 2004;**20**:161–8.
140. Leone M, Pagnani A. Predicting protein functions with message passing algorithms. *Bioinformatics* 2005;**21**:239–47.
141. Dawid AP. Conditional independence in statistical theory. *Journal of the Royal Statistics Society, Series B* 1979;**41**:1–31.
142. Krogh A, Brown M, Mianan IS, *et al*. Hidden Markov models in computational biology: applications to protein modelling. *J Mol Biol* 1994;**235**:1501–31.
143. Pearl J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, 1988.
144. Lauritzen SL. *Graphical Models*. Oxford University Press, 1996.
145. Cowell RG, Dawid AP, Lauritzen SL, *et al*. *Probabilistic Networks and Expert Systems*. New York: Springer-Verlag, 1999.
146. Jensen FV. *Bayesian Networks and Decision Graphs*. New York: Springer, 2001.
147. Neapolitan E. *Learning Bayesian Networks*. Upper Saddle River, NJ: Prentice Hall, 2003.
148. Cooper GF. The computational complexity of probabilistic inference using belief networks. *Artificial Intelligence* 1990;**42**:393–405.
149. Heckerman D. A Tutorial on Learning with Bayesian Networks. Technical report, Microsoft Advanced Technology Division, Microsoft Corporation, Seattle, Washington, 1995.
150. Chickering M. Learning equivalence classes of Bayesian networks structures. In: *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*. Portland: Morgan Kaufmann, 1996: pp. 150–7.
151. Larrañaga P, Kuijpers CMH, Murga RH, *et al*. Searching for the best ordering in the structure learning of Bayesian networks. *IEEE Transactions on Systems, Man and Cybernetics* 1996;**41**(4):487–93.
152. Chickering DM, Geiger D, Heckerman D. Learning Bayesian Networks is NP–hard. Technical report, Microsoft Research, Redmond, WA 1994.
153. Shachter R, Kenley C. Gaussian influence diagrams. *Management Science* 1989;**35**:527–50.
154. Smith PWF, Whittaker J. Edge exclusion tests for graphical Gaussian models. *Learning in Graphical Models*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 1998: pp. 555–74.
155. Geiger D, Heckerman D. Learning Gaussian Networks. Technical report, Microsoft Advanced Technology Division, Microsoft Corporation, Seattle, Washington 1994.
156. Meyer IM, Durbin R. Gene structure conservation aids similarity based gene prediction. *Nucleic Acids Research* 2004;**32**(2):776–83.
157. Cawley SL, Pachter L. HMM sampling and applications to gene finding and alternative splicing. *Bioinformatics* 2003; **19**(Suppl. 2):ii36–41.
158. Cai D, Delcher A, Kao B, *et al*. Modeling splice sites with Bayes networks. *Bioinformatics* 2000;**16**(2):152–8.
159. Greenspan G, Geiger D. High density linkage disequilibrium mapping using models of haplotype block variation. *Bioinformatics* 2004;**20**(Suppl. 1):i137–44.
160. Pollastri G, Baldi P. Prediction of contact maps by GIOHMMs and recurrent neural networks using lateral propagation from all four cardinal corners. *Bioinformatics* 2002;**18**(Suppl. 1):S62–70.
161. Raval A, Ghahramani Z, Wild DL. A Bayesian network model for protein fold and remote homologue recognition. *Bioinformatics* 2002;**18**(6):788–801.
162. Friedman N, Linial M, Nachman I, *et al*. Using Bayesian networks to analyze expression data. *J Comput Biol* 2000; **7**(3–4):601–20.
163. Larrañaga P, Inza I, Flores JL. A guide to the literature on inferring genetic networks by probabilistic graphical models. *Data Analysis and Visualization in Genomics and Proteomics*. John Wiley and Sons, Ltd., 2005: pp. 215–38.
164. Pearl J. *Causality. Models, Reasoning, and Inference,* . Cambridge University Press, 2000.
165. Pe'er D, Regev A, Elidan G, *et al*. Inferring subnetworks from perturbed expression profiles. *Bioinformatics* 2001;**17**: 215–24.
166. Husmeier D. Reverse engineering of genetic networks with Bayesian networks. *Biochemical Society Transactions* 2003;**31**(6):1516–18.
167. Rangeland C, Angus J, Ghahramani Z. *et al*. *Modelling Genetic Regulatory Networks using Gene Expression Profiling and Statespace Models*. Springer-Verlag, 2005: pp. 269–93.
168. Chang J.-H, Hwang K.-B, Zhang B.-T. Analysis of gene expression profiles and drug activity patterns by clustering and Bayesian network learning. *Methods of Microarray Data Analyis II*. Kluwer Academic Publishers, 2002: pp. 169–184.
169. Hartemink AJ, Gifford DK, Jaakkola TS, *et al*. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. *Pacific Symposium on Biocomputation 6*, 2001: pp. 422–33.
170. Hwang K.-B, Cho D.-Y, Park S.-W, *et al*. Applying machine learning techniques to analysis of gene expression data: cancer diagnosis. *Methods of Microarray Data Analysis*. Kluwer Academic Publishers, 2001: pp. 167–82.

171. Lee PH, Lee D. Modularized learning of genetic interaction networks from biological annotations and mRNA expression data. *Bioinformatics* 2005;**21**(11): 2739–47.

172. Markowetz F, Spang R. Reconstructing gene regulation networks from passive observations and active interventions. In: *Proceedings of the European Conference on Computational Biology*, 2003.

173. Pasanen T, Toivanen T, Tolvanen M, *et al*. DNA Microarray. *Data Analysis*, CSC–Scientific Computing Ltd., 2003.

174. Peña JM, Björkegren J, Tegnér J. Growing Bayesian network models of gene networks from seed genes. *Bioinformatics* 2005;**21**(Supp. 2):ii224–9.

175. Segal E, Taskar B, Gasch A, *et al*. Rich probabilistic models for gene expression. *Bioinformatics* 2001;**17**(1):243–52.

176. Spirtes P, Glymour C, Scheines R, *et al*. Constructing Bayesian networks models of gene expression networks from microarray data. In: *Proceedings of the Atlantic Symposium on Computational Biology*, 2000.

177. Tamada Y, Kim SY, Bannai H, *et al*. Estimating gene networks from gene expression data by combining Bayesian network model with promotor element detection. *Bioinformatics* 2003;**19**(Suppl. 2):ii227–36.

178. Nariai N, Kim S, Imoto S, *et al*. Using protein-protein interactions for refining gene networks estimated from microarray data by Bayesian networks. In: *Proceedings of the 9th Pacific Symposium on Biocomputing*, 2004: pp. 336–47.

179. Imoto S, Kim SY, Shimodaira H, *et al*. Bootstrap analysis of gene networks based on Bayesian networks and nonparametric regression. *Genome Informatics* 2002;**13**:369–70.

180. De Hoon MJL, Makita Y, Imoto S, *et al*. Predicting gene regulation by sigma factors in bacillus subtilis from genome–wide data. *Bionformatics* 2004;**20**:i101–8.

181. Husmeier D. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics* 2003;**19**(17): 2271–82.

182. Friedman N. Inferring cellular networks using probabilistic graphical models. *Science* 2004;**303**:799–805.

183. Imoto S, Higuchi T, Goto T, *et al*. Using Bayesian networks for estimating gene networks from microarrays and biological knowledge. In: *Proceedings of the European Conference on Computational Biology*, 2003.

184. Wu X, Ye Y, Subramanian KR. Interactive analysis of gene interactions using graphical Gaussian model. In: *BIOKDD03: 3rd ACM SIGKDD Workshop on Data Mining in Bioinformatics* 2003: pp. 63–69.

185. Husmeier D. *Inferring Genetic Regulatory Networks from Microarray Experiments with Bayesian Networks*. Springer-Verlag, 2005: pp. 239–67.

186. Murphy K, Mian S. Modelling Gene Expression Data using Dynamic Bayesian Networks. Technical report, Department of Computer Science. University of California at Berkeley, 1999.

187. Nachman I, Regev A, Friedman N. Inferring quantitative models of regulatory networks from expression data. *Bioinformatics* 2004;**20**(Suppl. 1):i248–56.

188. Ong IM, Glasner JD, Page D. Modelling regulatory pathways in e. coli from time series expression profiles. *Bioinformatics* 2002;**18**(Suppl. 1):S241–8.

189. Ong IM, Page D. Inferring Regulatory Pathways in e.coli using Dynamic Bayesian Networks. Technical Report 1426, Computer Sciences. University of Wisconsin–Madison, 2001.

190. Sugimoto N, Iba H. Inference of gene regulatory networks by means of dynamic differential Bayesian networks and nonparametric regression. *Genome Informatics* 2004;**15**(2):121–30.

191. Steffen M, Petti A, Aach J, *et al*. Automated modelling of signal transduction networks. *BMC Bioinformatics* 2002;**3**:34.

192. Looger LL, Hellinga HW. Generalized dead-end elimination algorithms make large-scale protein side-chain structure prediction tractable: Implications for protein design and structural genomics. *J Mol Biol* 2001;**307**(1):429–45.

193. Metropolis N, Rosenbluth AW, Teller AH, *et al*. Equations of state calculations by fast computing machines. *J Chem Phys* 1953;**21**:1087–91.

194. Kirkpatrick S, Gelatt CD, Jr, Vecchi MP. Optimization by simulated annealing. *Science* 1983;**220**:671–80.

195. Glover F. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research* 1986;**5**:533–49.

196. Goldberg D. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

197. Koza JR. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: The MIT Press, 1992.

198. Larrañaga P, Lozano JA (eds). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Boston/Dordrecht/London: Kluwer Academic Publishers, 2002.

199. Wei Shi, Wanlei Zhou, Yi-Ping Phoebe Chen. Biological sequence assembly and alignment. In: Yi-Ping Chen (ed). *Bioinformatics Technology*. Springer-Verlag, 2005: pp. 244–61.

200. Tariq Riaz, Yi Wang, Kuo-Bin Li. Multiple sequence alignment using tabu search. In: *Proceedings of the Second Conference on Asia-Pacific Bioinformatics*. Australian Computer Society, Inc., 2004: pp. 223–32.

201. Neuwald AF, Liu JS. Gapped alignment of protein sequence motifs through Monte Carlo optimization of a hidden Markov model. *BMC Bioinformatics* 2004;**5**:157–73.

202. Hung Dinh Nguyen, Ikuo Yoshihara, Kunihito Yamamori, *et al*. Aligning multiple protein sequences by parallel hybrid genetic algorithm. *Genome Informatics* 2002; **13**:123–32.

203. Thomas D. Schneider, David N. Mastronarde. Fast multiple alignment of ungapped DNA sequences using information theory and a relaxation method. *Discrete Applied Mathematics* 1996;**71**:259–68.

204. Kim J, Cole JR, Pramanik S. Alignment of possible secondary structures in multiple RNA sequences using simulated annealing. *Computer applications in the Biosciences* 1996;**12**(8):259–67.

205. Hirosawa M, Totoki Y, Hoshida M, *et al*. Comprehensive study on iterative algorithms of multiple sequence alignment. *Computer Applications in the Biosciences* 1995;**11**(1):13–18.

206. Ishikawa M, Toya T, Hoshida M, *et al*. Multiple sequence alignment by parallel simulated annealing. *Computer Applications in the Biosciences* 1993;**9**(3):267–73.

207. Knudsen S. Promoter 2.0: for the recognition of Pol II promoter sequences. *Bioinformatics* 1999;**15**(5): 356–61.

208. Jacob E, Sasikumar R, Nair KNR. A fuzzy guided genetic algorithm for operon prediction. *Bioinformatics* 2005;**21**(8): 1403–7.

209. Gary B. Fogel, Kumar Chellapilla, David B. Fogel. Identification of coding regions in DNA sequences using evolved neural networks. In: Gary B. Fogel, David W. Corne (eds). *Evolutionary Computation in Bioinformatics,* . Morgan Kaufmann, 2002: pp. 195–218.

210. Marylyn D. Ritchie, Bill C. White, Joel S. Parker, *et al*. Optimization of neural network architecture using genetic programming improves detection and modeling of gene-gene interactions in studies of human diseases. *BMC Bioinformatics* 2003;**4**(28):7.

211. Yvan Saeys, Sven Degroeve, Dirk Aeyels, *et al*. Fast feature selection using a simple estimation of distribution algorithm: A case study on splice site prediction. *Bioinformatics* 2003;**19**(2):ii179–88.

212. Blanco R, Larrañaga P, Inza I, *et al*. Selection of highly accurate genes for cancer classification by estimation of distribution algorithms. In: *Proceedings of the Workshop 'Bayesian Models in Medicine' held within AIME, 2001* 2001: pp. 29–34.

213. Blazewicz J, Formanowicz P, Kasprzak M, *et al*. Tabu search algorithm for DNA sequencing by hybridization with isothermic libraries. *Computational Biology and Chemistry* 2004;**28**(1):11–19.

214. Takaho A. Endo. Probabilistic nucleotide assembling method for sequencing by hybridization. *Bioinformatics* 2004;**20**(14):2181–8.

215. Allon G. Percus, David C. Torney. Greedy algorithms for optimized DNA sequencing. In: *SODA'99: Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Philadelphia, PA, USA, Society for Industrial and Applied Mathematics, 1999: pp. 955–6.

216. Blazewicz J, Borowski M, Formanowicz P, *et al*. Tabu Search Method for Determining Sequences of Amino Acids in Long Polypeptides. Volume 3449 of Lecture Notes in Computer Science. Springer Verlag, 2005: pp. 22–32.

217. Matsuura T, Ikeguchi T. Tabu search for extracting motifs from DNA sequences. In: *Proceedings of the 6th Metaheuristics International Conference* 2005. To appear.

218. Christof T, Junger M, Kececioglu J, *et al*. A branch-and-cut approach to physical mapping of chromosomes by unique end-probes. *J Comput Biol* 1997;**4**(4):433–47.

219. Bhandarkar SM, Huang J, Arnold J. Parallel Monte Carlo methods for physical mapping of chromosomes. In: *Proceedings of the IEEE Computer Society Bioinformatics Conference*. IEEE press, 2002: pp. 64–75.

220. Brown DG, Vision TJ, Tanksley SD. Selecting mapping: a discrete optimization approach to select a population subset for use in a high-density genetic mapping project. *Genetics* 2000;**155**:407–20.

221. Jinling Huang, Suchendra M. Bhandarkar. A comparison of physical mapping algorithms based on the maximum likelihood model. *Bioinformatics* 2003;**19**(7):1303–10.

222. Han-Lin Li, Chang-Jui Fu. A linear programming approach for identifying a consensus sequence on DNA sequences. *Bioinformatics* 2005;**21**(9):1838–45.

223. Jonathan M. Keith, Peter Adams, Darryn Bryant, *et al*. A simulated annealing algorithm for finding consensus sequences. *Bioinformatics* 2001;**18**(10):1494–9.

224. Chen T, Kao MY, Tepel M, *et al*. A dynamic programming approach to de novo peptide sequencing via tandem mass spectrometry. *J Comput Biol* 2001;**8**(3):325–37.

225. Michael Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research* 2003; **31**(13):3406–15.

226. Gary B. Fogel, William Porto V, Dana G. Weekes, *et al*. Discovery of RNA structural elements using evolutionary computation. *Nucleid Acid Research* 2002; **30**(23):5310–17.

227. Blazewicz J, Lukasiak P, Milostan M. RNA tertiary structure determination: NOE pathways construction by tabu search. *Bioinformatics* 2005;**21**(10):2356–61.

228. Blazewicz J, Lukasiak P, Milostan M. Application of tabu search strategy for finding low energy structure of protein. *Artificial Intelligence in Medicine* 2005;**35**:135–45.

229. Neal Lesh, Michael Mitzenmacher, Sue Whitesides. A complete and effective move set for simplified protein folding. In: *Proceedings of the Seventh Annual International Conference on Research in Computational Molecular Biology* 2003: pp. 188–95.

230. Hsiao-Ping Hsu, Vishal Mehra, Peter Grassberger. Structure optimization in an off-lattice protein model. *Physical Review E* 2003;**68**(2):4.

231. Hsiao-Ping Hsu, Vishal Mehra, Walter Nadler, *et al*. Growth algorithms for lattice heteropolymers at low temperatures. *J Chemical Physics* 2003;**118**(1):444–51.

232. Liang S, Wong WH. Evolutionary Monte Carlo for protein folding simulation. *Journal of Chemical Physics* 2001; **115**:3374–80.

233. Natalio Krasnogor, Blackburne BP, Edmund K. Burke, *et al*. Algorithms for protein structure prediction. In: Merelo JJ, Adamidis P, Beyer HG, Fernandez-Villacañas JL, Schwefel HP, (eds). *Parallel Problem Solving from Nature - PPSN VII,* Volume 2439 of Lecture Notes in Computer Science. Granada, Spain Springer Verlag, 2002: pp. 769–78.

234. Gary B. Lamont, Lawrence D. Merkle. Toward effective polypeptide structure prediction with parallel fast messy genetic algorithms. In: Gary B. Fogel, David W. Corne, (eds). *Evolutionary Computation in Bioinformatics*. Morgan Kaufmann, 2002: pp. 137–62.

235. Smith J. The co-evolution of memetic algorithms for protein structure prediction. In: William WH, Krasnogor N, Smith JE (eds). *Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing*. Springer, 2004: pp. 105–28.

236. Roberto Santana, Larrañaga P, Lozano JA. Protein folding in 2-dimensional lattices with estimation of distribution algorithms. *Proceedings of the First International Symposium on Biological and Medical Data Analysis*, Volume 3337 of Lecture Notes in Computer Science. Barcelona, Spain: Springer Verlag, 2004: pp. 388–98.

237. De Maeyer M, Desmet J, Lasters I. The dead-end elimination theorem: mathematical aspects, implementation, optimizations, evaluation, and performance. *Methods in Molecular Biology* 2000;**143**:265–304.

238. Zhijie Liu, Weizhong Li, Shide Liang, *et al*. Beyond rotamer library: Genetic algorithm combined with disturbing mutation process for upbuilding protein side-chains. *Proteins: Structure, Function, and Genetics* 2003; **50**:49–62.

239. Tuffery P, Etchebest C, Hazout S, *et al*. A new approach to the rapid determination of protein side chain conformations. *J Biomolecular Structure Dynamics* 1991;**8**:1267–89.

240. Jinn-Moon Yang, Chi-Hung Tsai, Ming-Jing Hwang, *et al*. GEM: a Gaussian evolutionary method for predicting protein side-chain conformations. *Protein Science* 2002;**11**: 1897–907.

241. Glick M, Rayan A, Goldblum A. A stochastic algorithm for global optimization for best populations: a test case of side chains in proteins. *Proceedings of the National Academy of Sciences* 2002;**99**(2):703–8.

242. Lee C, Subbiah S. Prediction of protein side-chain conformation by packing optimization. *J Mol Biol* 1991; **217**:373–88.

243. Yanover C, Weiss Y. Approximate inference and protein-folding. In: Becker S, Thrun S, Obermayer K (eds). *Advances in Neural Information Processing Systems 15*. Cambridge, MA: MIT Press, 2003: pp. 1457–64.

244. Koehl P, Delarue M. Building protein lattice models using self consistent mean field theory. *J Chemical Physics* 1998; **108**:9540–49.

245. Fiser A, Do RK, Sali A. Modeling of loops in protein structures. *Protein Science* 2000;**9**:1753–73.

246. Robert M. MacCallum. Striped sheets and protein contact prediction. *Bioinformatics* 2004;**20**(8)i:224–31.

247. Shin Ando, Hitoshi Iba, Erina Sakamoto. Modeling genetic network by hybrid GP. In: David B. Fogel, Mohamed A. El-Sharkaw, iXin Yao, Garry Greenwood, Hitoshi Iba, Paul Marrow, Mark Shackleton, (eds). *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*. IEEE Press, 2002: pp. 291–96.

248. Shin Ando, Erina Sakamoto, Hitoshi Iba. Evolutionary modeling and inference of gene network. *Information Sciences* 2002;**145**(3–4):237–59.

249. Sakamoto E, Iba H. Inferring a system of differential equations for a gene regulatory network by using genetic programming. In: *Proceedings of Congress on Evolutionary Computation*. IEEE Press, 2001: pp. 720–26.

250. Koza JR, Mydlowec W, Lanza G, *et al*. Reverse engineering of metabolic pathways from observed data using genetic programming. In: *Proceedings of the Pacific Symposium on Biocomputing 6*. Hawaii: World Scientific Press, 2001: pp. 434–45.

251. Kyle Ellrott, Chuhu Yang, Frances M. Sladek, *et al*. Identifying transcription factor binding sites through Markov chain optimization. *Bioinformatics* 2002; **18**(90002):100–9.

252. Shinichi Kikuchi, Daisuke Tominaga, Masanori Arita, *et al*. Dynamic modeling of genetic networks using genetic algorithm and S-system. *Bioinformatics* 2003;**19**(3):643–50.

253. Gilman A, Ross J. Genetic-algorithm selection of a regulatory structure that directs flux in a simple metabolic model. *Biophysical Journal* 1995;**69**:1321–33.

254. Park LJ, Park CH, Park C, *et al*. Application of genetic algorithms to parameter estimation of bioprocesses. *Medical and Biological Engineering and Computing* 1997;**35**(1):47–9.

255. Shuhei Kimura, Kaori Ide, Aiko Kashihara, *et al*. Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm. *Bioinformatics* 2005;**21**(7):1154–63.

256. Noman N, Iba H. Inference of gene regulatory networks using S-system and differential evolution. In: *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*. ACM Press, 2005: pp. 439–46.

257. Ernst Wit, Agostino Nobile, Raya Khanin. Near-optimal designs for dual channel microarray studies. *J Royal Statistical Society Series C* 2005;**54**(5):817–30.

258. Jonathan D. Wren, Tinghua Yao, Marybeth Langer, *et al*. Simulated annealing of microarray data reduces noise and enables cross-experimental comparisons. *DNA and Cell Biology* 2004;**23**(10):695–700.

259. Kenneth Bryan, Padraig Cunningham, Nadia Bolshakova. Biclustering of expression data using simulated annealing. In: *18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)* 2005: pp. 383–8.

260. Alexander V. Lukashin, Rainer Fuchs. Analysis of temporal gene expression profiles: clustering by simulated annealing and determining the optimal number of clusters. *Bioinformatics* 2001;**17**(5):405–14.

261. Emanuel Falkenauer, Arnaud Marchand. Clustering microarray data with evolutionary algorithms. In: Gary B. Fogel, David W. Corne (eds). *Evolutionary Computation in Bioinformatics*. Morgan Kaufmann, 2002: pp. 219–30.

262. Ilya Shmulevich, Wei Zhang. Binary analysis and optimization based normalization of gene expression data. *Bioinformatics* 2002;**18**(4):555–65.

263. Gary B. Fogel. Evolutionary computation for the inference of natural evolutionary histories. *IEEE Connections* 2005; **3**(1):11–14.

264. Kumar S. A stepwise algorithm for finding minimum evolution trees. *Mol Biol Evol* 1996;**13**(4):584–93.

265. Ribeiro CC, Vianna DS. A GRASP/VND heuristic for the phylogeny problem using a new neighborhood structure. *International Transactions in Operational Research* 2005;**12**:325–38.

266. Guindon S, Gascuel O. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic Biology* 2003;**52**(5):696–704.

267. Barker D. LVB: parsimony and simulated annealing in the search for phylogenetic trees. *Bioinformatics* 2004;**20**(1): 274–5.

268. Rui-Sheng Wang, Ling-Yun Wu, Zhen-Ping Li, *et al*. Haplotype reconstruction from SNP fragments by minimum error correction. *Bioinformatics* 2005;**21**(5): 2456–62.

269. Jaime R. Robles, Edwin JCG. van den Oord. lga972: a cross-platform application for optimizing LD studies using a genetic algorithm. *Bioinformatics* 2004;**20**(17): 3244–5.

270. Moreira A. Genetic algorithms for the imitation of genomic styles in protein backtranslation. *Theoretical Computer Science* 2004;**322**:297–312.

271. Jain-Shing Wu, Chungnan Lee, Chien-Chang Wu, *et al*. Primer design using genetic algorithm. *Bioinformatics* 2004; **20**(11):1710–17.

272. Dan Ashlock, Jim Golden. Evolutionary computation and fractal visualization of sequence data. In: Gary B. Fogel, David W. Corne (eds). *Evolutionary Computation in Bioinformatics*. Morgan Kaufmann, 2002: pp. 231–53.