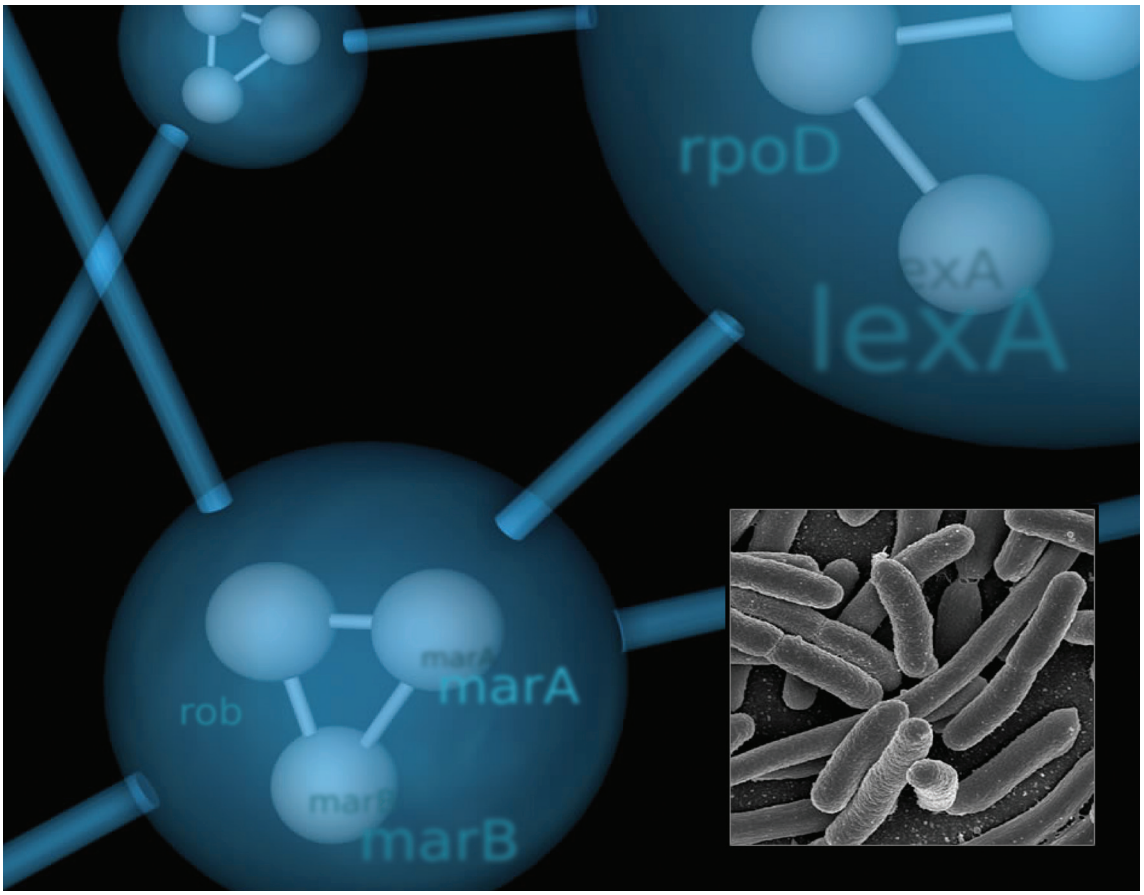
Molecular BioSystems

This article was published as part of the

Computational and Systems Biology themed issue

Please take a look at the full table of contents to access the other papers in this issue.



Supervised learning with decision tree-based methods in computational and systems biology†‡

Pierre Geurts,* Alexandre Irrthum and Louis Wehenkel

Received 21st April 2009, Accepted 8th September 2009
First published as an Advance Article on the web 5th October 2009
DOI: 10.1039/b907946g

At the intersection between artificial intelligence and statistics, supervised learning allows algorithms to automatically build predictive models from just observations of a system. During the last twenty years, supervised learning has been a tool of choice to analyze the always increasing and complexifying data generated in the context of molecular biology, with successful applications in genome annotation, function prediction, or biomarker discovery. Among supervised learning methods, decision tree-based methods stand out as non parametric methods that have the unique feature of combining interpretability, efficiency, and, when used in ensembles of trees, excellent accuracy. The goal of this paper is to provide an accessible and comprehensive introduction to this class of methods. The first part of the review is devoted to an intuitive but complete description of decision tree-based methods and a discussion of their strengths and limitations with respect to other supervised learning methods. The second part of the review provides a survey of their applications in the context of computational and systems biology.

Introduction

Recent advances in experimental techniques have resulted in a dramatic increase in the amount and diversity of data being generated in the context of molecular biology. With tools such as microarrays, high-throughput sequencing, mass spectrometry, or advanced imaging techniques, measurements are possible at various scales and for more and more components of biological systems. The integration of all these data with available knowledge to better understand biological systems is at the heart of systems biology. The increase of complexity and dimensionality requires advanced computational approaches to store, organize, and analyze the data before they could be turned into novel biological knowledge.

Among computational biology tools, machine learning naturally appears as one of the key components. In particular, supervised learning provides techniques to learn predictive models only from observations of a system and is thus particularly well suited to deal with the highly experimental nature of biological knowledge. Supervised learning has indeed been applied to a wide spectrum of problems in computational biology, ranging from genome annotations and structure prediction to the inference of networks of interactions between various kinds of biological entities, the prediction of protein functions, and the discovery of biomarkers and drugs.

Department of EE and CS & GIGA-Research, University of Liège, Belgium. E-mail: p.geurts@ulg.ac.be

† This article is part of a *Molecular BioSystems* themed issue on Computational and Systems Biology

‡ Electronic supplementary information (ESI) available: Information about various non-standard extensions of the decision tree-based approach to modeling, some practical guidelines for the choice of parameters and algorithm variants depending on the practical objectives of their application, pointers to freely accessible software packages, and a brief primer going through the different manipulations needed to use the tree-induction packages available in the **R** statistical tool; a readable version of Fig. 3 (top) is included. See DOI: 10.1039/b907946g

Pierre Geurts

Pierre Geurts graduated as an Electrical Engineer (in computer science) in 1998 and received the PhD degree in applied sciences in 2002, both from the University of Liège, Belgium. He is currently a research associate at the Belgian Fund for Scientific Research (F.N.R.S.). His research interests include machine learning and its applications in various fields, such as bioinformatics, computer vision, and computer networks.

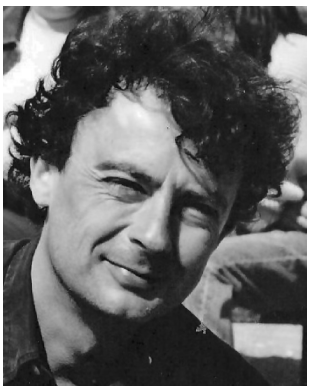
Alexandre Irrthum

Alexandre Irrthum graduated as a Bioengineer in 1996, got a Master in Biochemistry and Molecular and Cell Biology in 1997 and received the PhD degree in Biomedical Sciences in 2003, all from the Université catholique de Louvain, Belgium. His research interests include bioinformatics, systems biology and genetics, and particularly the application of machine learning to the analysis of biological networks.

Among all supervised learning methods, this paper is devoted to a particular family of methods, called decision tree-based methods. Decision trees are among the most popular learning algorithms and they have been applied extensively in computational biology. The key ingredients to the success of these methods are their interpretability, which makes their model transparent and understandable to human experts, their flexibility, which makes them applicable to a wide range of problems, and their ease of use, which makes them accessible even to non-specialists. Combined with ensemble methods, they often provide state-of-the-art results in terms of predictive accuracy. Importantly in the context of high-throughput data sets, tree-based methods are also highly scalable from a computational point of view.

The goal of this review is to provide an accessible and comprehensive introduction to tree-based supervised learning methods and a survey of their applications in computational and systems biology. The paper is structured as follows. We begin with a short introduction to supervised learning, with a discussion of the bias/variance tradeoff and of cross-validation methods. We then proceed with a detailed description of tree-based methods, starting with the main steps of standard decision tree induction, followed by a discussion of ensemble methods and tree-based attribute importance measures. We conclude this section with a discussion of the main strengths and limitations of these methods with respect to other supervised learning algorithms. The remainder of the paper is devoted to a survey of the main classes of applications of decision tree methods in the context of computational and systems biology. Supplementary material discusses several extensions of tree-based methods, gives some pointers towards freely available software packages implementing these methods, and provides a few practical guidelines on how to best exploit these methods, which are illustrated with an application in the **R** statistical language.‡

The reading of this paper can be usefully complemented by a few other papers. An short intuitive review of decision trees can be found in Kingsford and Salzberg.[1] Two more general introductions to machine learning in biology are given by Larrañaga *et al*.[2] and by Tarca *et al*.[3] Two other recent papers are dedicated to two other families of supervised learning methods and their applications in computational biology: support vector machines[4] and artificial neural networks.[5]



Louis Wehenkel

Louis Wehenkel graduated as an Electrical Engineer (electronics) in 1986 and received the PhD degree in 1990, both at the University of Liège, where he is full Professor of Electrical Engineering and Computer Science. His research interests lie in the fields of stochastic methods for systems and modeling, machine learning and data mining, with applications in electric power systems planning, operation and control, image analysis and bioinformatics.

Supervised learning

In this section, we briefly introduce supervised learning. For more details, we refer the reader to general textbooks about supervised learning (and machine learning in general).[6–8]

Problem statement

Machine learning denotes a broad class of computational methods which aim at extracting a model of a system from the sole observation (or the simulation) of this system in some situations. By the term *model*, we mean a set of exact or approximate relationships between the observed variables of the system. The goal of such models may be to predict the behavior of this system in some yet unobserved situations or to help understanding its already observed behavior.

Supervised learning refers to the subset of machine learning methods which derive models in the form of input–output relationships. More precisely, the goal of supervised learning is to identify a mapping from some *input* variables to some *output* variables on the sole basis of a given sample of joint observations of the values of these variables. The (input or output) variables are often called *attributes* or *features*, while joint observations of their values are called *objects* and the given sample of objects used to infer a model is generally called the *learning sample*.

As an illustration, consider the bi-dimensional classification problem of Fig. 1 and assume that the goal of learning is to find a function which discriminates at best between red and green points. For example, each point of this space might represent a patient whose state is described by two input variables that are the result of two medical tests. Patients marked in red are suffering from a particular disease while patients marked in green are not. The goal could be to help a physician to predict the status of a new patient (the output) for which the test results are known (the inputs). In the real world, of course, each point of the learning sample is generally described by (many) more than one or two input attributes and these attributes may have many other kinds of values than real numbers (discrete values, time series, images, sequences, for example). Also the output variables may represent more complex information than our binary decision variable.

The above problem is the kind of problem targeted by supervised learning algorithms. Loosely speaking, a supervised learning algorithm receives a learning sample (like the one represented in Fig. 1(a)) and returns an input-to-output function h (a hypothesis) which is chosen from a set of candidate models (its hypothesis space). Learning algorithms differ by the specific form of the hypotheses they consider (determined either by a parametric class of mathematical expressions or by a class of algorithmic procedures computing the outputs from the inputs), by their definition of the quality of hypotheses, and by the optimization strategy they use to find a high quality hypothesis in their hypothesis space.

For example, a potential candidate function returned by a learning algorithm is the one given in Fig. 1(b) in blue, represented by the decision boundary that it defines in the

$X1$	$X2$	Y
0.19	0.35	sick
0.44	0.94	healthy
0.63	0.08	sick
...		
0.20	0.63	healthy

(a)

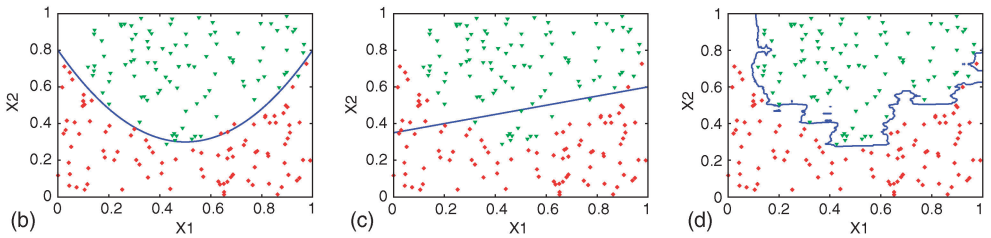


Fig. 1 An illustrative two-dimensional supervised learning problem. From left to right: (a) tabular learning sample, (b) scatter-plot of the learning sample together with the optimal classification boundary for this problem, (c) the classification boundary for a too simple model and (d) for a too complex one.

input space. Even though this function makes some mistakes on the learning sample, it seems that these mistakes are merely due to noise in the data (which may be due either to errors in measurements or to effects for which no input variables have been observed) and the function is thus a plausible explanation of the learning sample. As a matter of fact, for our toy problem, this particular input–output function turns out to be optimal in the sense that its predictions minimize the probability of mis-classification.

An important criterion to assess learning algorithms, especially in the context of computational and systems biology, is the interpretability of the models that it infers from the data, *i.e.* the information that it may bring to a human expert about the studied input–output relationship. For example, in our medical application, it would be interesting to know if one of the two tests is not useful or not necessary to predict the disease status. Computational efficiency and scalability are also of great concern, especially for very large datasets, be it in the number of objects or the number of attributes. However, the main criterion used to assess learning algorithms is their prediction accuracy, *i.e.* the way the model they produce generalizes to unseen data (*i.e.* data that were not observed in the learning sample).

Bias/variance tradeoff

To explain why supervised learning of models with good predictive accuracy is not trivial, it is necessary to understand the bias/variance tradeoff in this context.

To give some intuition about this tradeoff, let us first assume that we want to separate the two classes in Fig. 1 with a linear decision boundary. Fig. 1(c) shows one such model illustrating the fact that with linear models it is impossible to discriminate well enough the two classes of our toy problem. The family of linear models is not well suited to solve our task because the best linear model is too far from an optimal decision boundary like the one represented in Fig. 1(b), whatever the amount of data used to infer it. This source of error is called *bias*.

Let us assume now that we use a much larger family of hypotheses among which there exist one or several that realize a perfect separation between classes in our learning sample. Then, by using this hypothesis space, the learning algorithm might produce a decision boundary like the one shown in Fig. 1(d). This time, the classification boundary perfectly separates all learning objects of different classes. Like the linear model however, this model is still not appropriate. This time, the model is too complex and hence it learns "too much" information from our data. In this case, we say that the learning algorithm *overfits* the data. If we would use another learning sample for the same problem (*e.g.* by repeating the experiment on another group of patients), then it would be very likely that the model found by the same learning algorithm would be very different from the one of Fig. 1(d). In this case, we say that the learning algorithm suffers from *variance*, because the hypothesis that it returns for a given problem is very much dependent on the particular learning sample that is provided.

When it comes to apply the hypothesis inferred by supervised learning on new objects, both bias and variance are sources of error and hence they should ideally both be minimal. Moreover, as they react in opposite directions to variations of the complexity of the hypothesis space, there is a tradeoff between their effects which must be taken into account.

Because of its central role in supervised learning, this tradeoff has received a lot of attention in the design of supervised learning algorithms. The main approach to handle it is to adapt the complexity of the hypothesis space to the problem under consideration (during learning), *e.g.* by using cross-validation to select meta-parameters controlling the complexity of the hypothesis space.

Assessment by cross-validation

The evaluation of the predictive accuracy of a model is crucial in supervised learning, first in order to predict its future performance on new data, but also in order to provide the learning algorithm with a criterion to choose a model in its hypothesis space. Evaluation requires the choice of an estimation procedure and of a performance measure.

Usually the performance of the model as estimated on the learning sample, called the *resubstitution estimate*, is a very optimistic estimate of the true quality of the model. A more reliable estimate would be obtained from a sample independent of the data sample used during the learning stage. For example, a *hold-out estimate* is obtained by partitioning the learning sample ls into two disjoint subsets, the validation sample, vs, and the learning sample without the objects in vs, denoted $ls \backslash vs$. A model is learned from $ls \backslash vs$ and its performance is assessed on vs. A common practice is to save one third of the data for the validation. However, when the learning sample is very small, subdividing further the data will yield an unreliable estimate. In these cases, *cross-validation* provides a way to make a better use of the available sample. In k-fold cross-validation, we divide the learning sample into k disjoint subsets of the same size, $ls = ls_1 \cup ls_2 \cup \ldots \cup ls_k$. A model is then inferred by the learning algorithm from each sample

$ls \backslash ls_i$, $i = 1,\ldots,k$, and its performance is determined on the held out sample ls_i. The final performance is computed as the average performance over all these models. Notice that when k is equal to the number of objects in the learning sample, this method is called *leave-one-out*. Typically, smaller values of k (say, between 10 and 20) are however preferred for computational reasons.

Notice that when cross-validation is used to make a selection between several models (*e.g.* corresponding to different values of the meta-parameters of a learning algorithm or produced by different learning algorithms), then an additional external run of cross-validation is necessary to provide a reliable estimate of the finally chosen model.[9]

Evaluation also raises the question of the choice of a performance measure. This choice usually depends on the application. In the context of classification, *i.e.* when the output variable is discrete, the most common measure is the error rate of the model (*i.e.* the probability of making a wrong prediction as estimated for example by the proportion of wrong predictions on the validation sample). The main drawback of this measure is that it is sensitive to the proportion of objects of different classes in the validation sample and that it does not convey information about the kind of errors. In the context of binary classification, it is therefore usually complemented by other measures such as sensitivity (the proportion of objects in the class of interest that are well classified) or specificity (the proportion of objects in the other class that are well classified). Moreover, since most classifiers inferred by supervised learning actually produce a confidence score (*e.g.*, in the form of a class probability estimate), in the case of binary classification problems, one may use different thresholds on this score to achieve different compromises between sensitivity and specificity. To evaluate a model independently of the chosen threshold, one can use a *ROC curve*,[10] that plots true positive rate (sensitivity) *versus* false positive rate (1 − specificity) for different values of this threshold. The quality of a ROC curve is evaluated by the area under this curve (*AUC*), as we will illustrate below.

Decision tree induction

Decision tree induction[11,12] is currently one of the most interesting supervised learning algorithms. The paternity of tree-based learning methods is often attributed to Hunt with his Concept Learning System.[13] Later on, the method was formalized, extended, and popularized in the field of statistics by the famous book of Breiman, Friedman, Olsen, and Stone on Classification And Regression Trees (CART[11]). During the same period, Quinlan also contributed to the dissemination of this approach in the field of artificial intelligence with his ID3[14] and C4.5[12] algorithms.

Let us notice that decision tree induction, originally designed to solve classification problems, have been extended to handle more complex output spaces (*e.g.* single- or multi-dimensional regression). For the sake of simplicity, we will focus on the case of classification trees and describe in this context the main ideas behind this methodology. We will then discuss its main strengths and weaknesses with respect to other supervised learning methods. For advanced readers, the supplementary material elaborates on the extensions of the basic decision tree framework, in particular to regression.

Tree structured classifiers

In its simplest form, a decision tree combines several binary tests in a tree structure. For example, Fig. 2 plots a decision tree and the resulting decision boundary for the bi-dimensional example problem of Fig. 1. Each interior node of this tree is labeled with a test, which compares the value of an input attribute to a threshold, and each terminal node is labeled with a class. To produce a classification for a new object whose attribute values are known, we simply propagate it into the tree from the top node according to the test answers. When a terminal node is reached, its corresponding class label is attributed to the object. By using such tests, the tree progressively partitions the input space into hyper-rectangular regions where the output is constant.

Learning algorithm

There exist many variants of decision tree learning methods. Our goal in this section is therefore to give the general idea of these algorithms, rather than to make an exhaustive list of all their variants. We refer the interested reader to the references mentioned throughout this section for more details about specific methods.

Loosely speaking, the general idea behind tree induction algorithms is to find a simple tree that has good predictive performance on the learning sample. Since the enumeration of all possible trees is essentially intractable, most tree induction algorithms are based on heuristics. The most common heuristic is a greedy top-down recursive partitioning approach. This algorithm starts with a single node tree corresponding to the complete learning sample and finds a way to split this node by selecting a test among a set of candidate tests. The algorithm then proceeds recursively to split the successors of this node. The whole process results in a partition of the learning sample into smaller and smaller subsets. The development of a branch is stopped when some stop-splitting criterion applies. Eventually, each terminal node of the tree is labeled with a prediction (class name or vector of class probabilities) which is computed on the basis of the subset of objects which reach this node. This tree growing step is then usually followed by a pruning stage which aims at removing unessential parts of the tree, so as to avoid overfitting. The different ingredients of this algorithm are further explained below.

Score measure. To split a node, a score measure is defined and the test among the set of candidate tests which realizes the best score is selected. The score measure is designed to favor tests which discriminate between objects of different output classes. To this end, the score usually evaluates the ability of the test to reduce the impurity of the class within the local learning (sub)sample. More precisely, most score measures are of the following form:

$$\text{Score}(S, T) = I(S) - \sum_{i=1}^{p} \frac{N_i}{N} I(S_i), \qquad (1)$$

where T is the candidate test, S is the local learning sample of size N which is split by T into p subsets S_i of size N_i

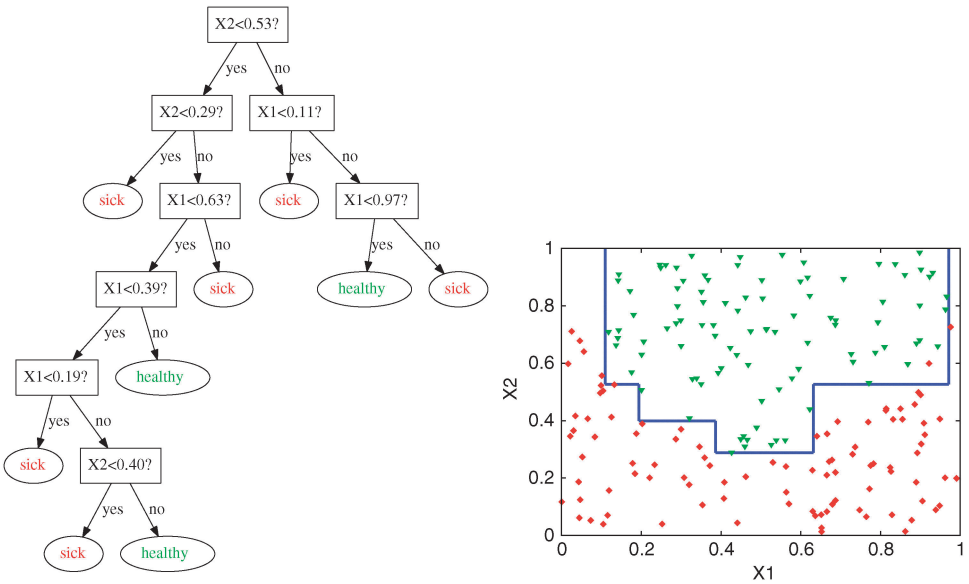


Fig. 2 Left, a decision tree, right, the corresponding decision boundary.

($i = 1,\ldots,p$), and $I(S)$ measures the impurity of the output in the sample S.

In the context of a classification problem with m classes, two popular measures of impurity are Shannon's (or logarithmic) entropy:

$$I(S) = -\sum_{c=1}^{m} \frac{N_c}{N} \log \frac{N_c}{N},$$

and Gini's (or quadratic) entropy:

$$I(S) = \sum_{c=1}^{m} \frac{N_c}{N} \left(1 - \frac{N_c}{N}\right),$$

where N is the size of the local learning sample S and N_c is the number of objects of output class c in S. These score measures give very similar results in practice and have been widely used in the context of decision tree induction, either as is[11,14] or normalized in different ways.[12,15]

Candidate tests. The set of candidate tests used in the score maximization procedure defines the actual hypothesis space and the representation power of decision trees of given complexity. In standard decision tree induction methods, candidate tests involve only one attribute at a time. For numerical attributes, standard tests are of the form $[X < x_{\text{th}}?]$, where x_{th} is a discretization threshold for the attribute X. The optimal thresholds are usually determined automatically by the induction algorithm, but a pre-discretization of numerical attributes could also be provided by the user. When the attribute takes its values in a finite set, typical tests consist in partitioning the set of all possible values of the attribute into a number of subsets and creating a successor node for each one of these subsets. In the literature, partitions into singletons (one successor for each value of the attribute) or into two subsets are generally considered. In the second case, tests are of the form $[X \in V?]$, where V is a subset of all possible values of the attribute X. Like discretization thresholds, subsets V are automatically determined by the induction algorithm so as to maximise the score.

Node labeling. In classification, the simplest labeling of decision tree leaves with a class prediction is obtained by computing the majority class among the objects present in the leaf. A more detailed information can be stored in the leaves in the form of an estimation of the conditional class probabilities. These probabilities can be estimated by class frequency counts but more sophisticated estimation techniques have also been proposed.[16]

Stop-splitting and pruning. During induction, the decision tree can usually be extended as far as needed to perfectly separate objects from the learning sample. However, from the discussion about the bias–variance tradeoff, such fully grown trees are often likely to overfit and suffer from a high variance. There exist essentially two ways to prevent a tree from overfitting: using a stop-splitting criterion to interrupt the development of a branch before the tree starts to overfit, or using a post-pruning algorithm simplifying an overly grown tree so as to maximise its performance, *e.g.* as estimated by cross-validation.

There are several possible stop-splitting criteria. A possibility is to compute the impurity of the output variable in the subset and stop splitting if this value is lower than a given threshold. It is also possible to stop induction when the number of instances in the current subset is too small. More elaborate stop-splitting criteria have also been designed based for example on statistical hypothesis testing.[12,15]

One disadvantage of stop-splitting is that it requires the user to fix in advance the value of one or several meta-parameters that determine the final complexity and it may thus miss the complexity that optimizes the bias–variance tradeoff for a given problem. Pruning on the other hand consists of simplifying the tree *a posteriori* by closing some of its test nodes so as to find the best compromise between bias and variance for the problem at hand. Post-pruning generates from a fully grown

tree a sequence of (nested) trees of decreasing complexity, and selects among these the sub-tree which minimizes the error rate estimated by holdout or by cross-validation. Different pruning methods essentially differ in the way they generate the sequence of nested sub-trees. One example of such algorithm is the *cost-complexity pruning* method proposed in CART.[11]

Ensembles of decision trees

Although pruning may reduce substantially the complexity (and hence improve the interpretability) of the tree by removing useless parts of it, its effect on accuracy is often limited.[17] A more efficient way to improve the accuracy of decision trees is to aggregate the predictions given by several of them for the same problem. Various techniques have been proposed to generate different decision trees from the same learning sample that often give very impressive improvements of accuracy when they are aggregated. Below, we discuss the two main families of such methods, namely randomization methods and boosting methods.

Randomization methods. The general idea of these methods is to introduce some random perturbation in the process that generates a tree from a learning sample in order to get an ensemble of different trees. To make a prediction with an ensemble, the test object is propagated in all trees and the final prediction is obtained by aggregating the predictions of the individual trees for this object. In classification, an aggregated prediction may be obtained simply by computing the majority class among all individual predictions. One can also average the class probability estimates produced by the individual trees of the ensemble to produce an aggregated score for each class and then select the class corresponding to the largest score.

The most prominent method in this family is *Bagging* (for *b*ootstrap *agg*regat*ing*[18]) method. Bagging is a generic method to randomize any learning algorithm. The idea of Bagging is to learn each model of the ensemble from a bootstrap sample§ drawn from the original learning sample using the unmodified learning algorithm. While Bagging very much improves standard single trees in terms of accuracy, several other methods have been proposed in the literature to generate randomized ensembles of decision trees and these methods are often more accurate than bagging. For example, Breiman's Random Forests algorithm[19] builds each tree from a bootstrap sample like Bagging but modifies the node splitting procedure as follows: at each test node, K attributes are selected at random among all input attributes, an optimal candidate test is found for each of these attributes, and the best test among them is eventually selected to split the node. The Extremely Randomized Trees algorithm,[20] on the other hand, grows each tree of the ensemble from the complete original learning sample and generates its splits by selecting at each test node the best split among K random ones. Compared to Random Forests, this algorithm typically grows trees that are more strongly randomized for a given value of K. When $K = 1$, it even builds totally randomized trees, whose tests do not depend anymore on the learning sample output values. Both methods usually generate more accurate models than Bagging and they are much more efficient from the computational point of view.

Intuitively, these randomization methods act as variance reducers while they mostly do not affect the bias of the original tree induction method. Indeed, even with randomization, the trees can still be extended to almost perfectly predict the learning sample and thus have a low bias. However, because of the randomization, these trees are less dependent on the learning sample and thus their aggregation has less variance than the original (optimized on the learning sample) trees. From a geometrical point of view, the resulting ensembles of trees are also able to model much smoother classification boundaries than single trees (see Fig. 3).

Boosting methods. Although the main idea of boosting is also to combine the predictions of several models, its philosophy is very different from that of randomization methods. While randomization methods improve a low-bias and high-variance method, the goal of boosting is to "boost" the performance of a *weak learning* method *i.e.* a method that outputs very simple and highly biased models. Unlike randomization methods, boosting builds the models sequentially; each model in the sequence being built to address the deficiencies of the previous models in the sequence. Usually, boosted predictions are obtained by a weighted average of the predictions of the models in the resulting ensemble, where the weights of the models are chosen according to theoretical arguments.

One of the first and most famous boosting algorithms for classification is Freund and Schapire's AdaBoost method (for "adaptive boosting"[21]). The main idea of AdaBoost is to build each model in the sequence in such a way that it focuses on correcting the errors made by the previous models in the sequence. One way to achieve this is to replicate objects in the learning sample that are misclassified by the previous models. AdaBoost however exploits the fact that most learning algorithms can handle weighted objects. Starting with uniform "object" weights, this algorithm sequentially increases the weights of the objects misclassified by the previous models in the sequence, thereby forcing the learning algorithm to focus on the most "difficult" instances. Eventually, predictions are carried out by a weighted majority vote among the models according to another set of "model" weights which depend on the accuracy of the corresponding model on the learning sample.[21]

The efficiency of AdaBoost at improving classification learning algorithms has been highlighted in many empirical studies, mainly in combination with decision trees (*e.g.* ref. 22,23). Several variants of this original algorithm have been proposed, for classification problems[24] but also for regression problems.[7,25] For the boosting algorithm to work, each model must be learned with a weak learning algorithm. In the context of decision trees, this means that their complexity has to be reduced so that they do not overfit the learning sample. This is usually achieved by adjusting the stop-splitting criterion. One popular variant of decision trees in the context of boosting uses decision trees with only one test node, called "decision stumps".

§ A bootstrap sample is a sample of the same size as the original sample where each object is drawn with replacement from the original sample.

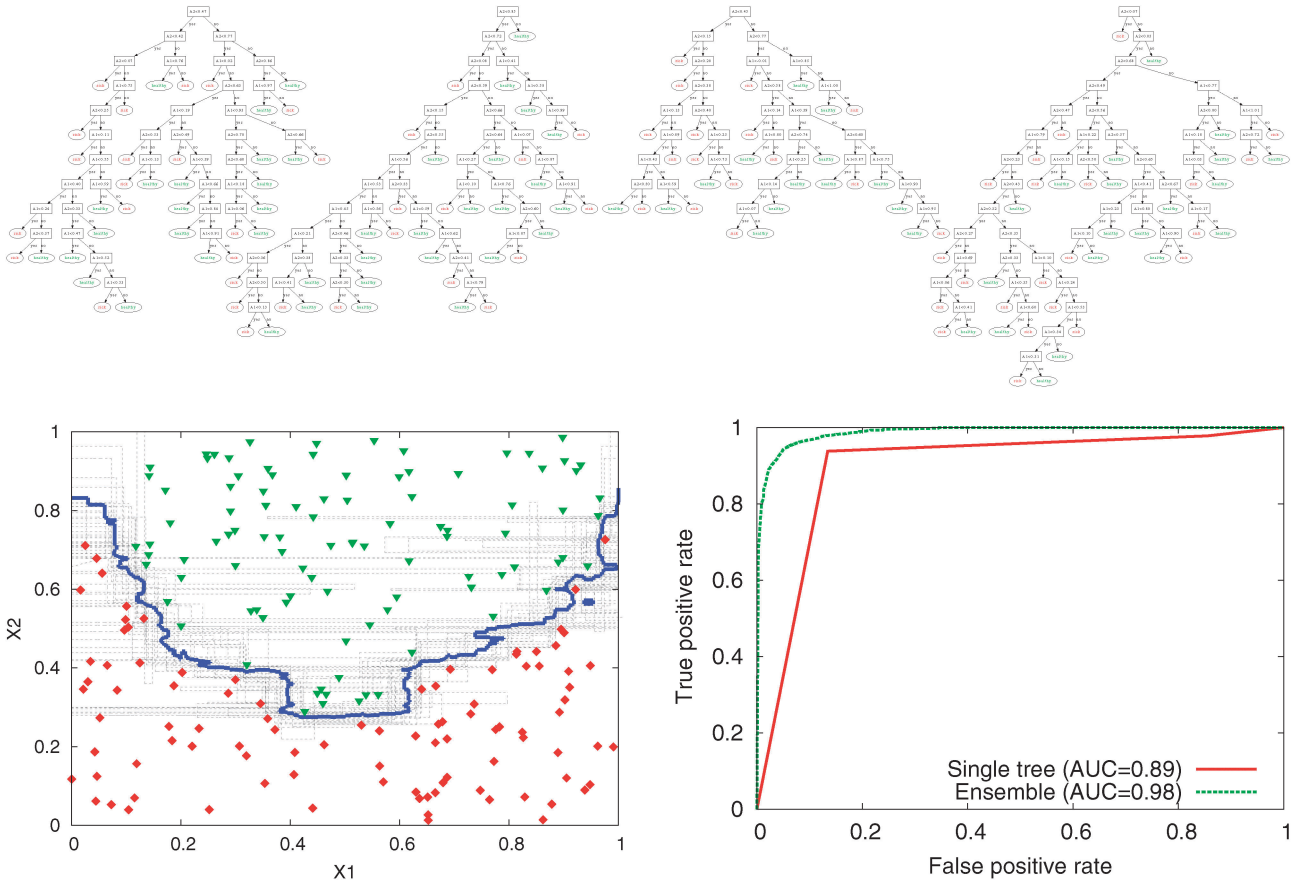


Fig. 3 Ensemble methods with trees: top, a subset of 4 trees built with the Extra-Trees randomization method[20] on the toy problem of Fig. 1 (a readable version is available in the ESI‡); bottom left, in dotted lines, the classification boundaries of 50 randomized trees and in plain blue line, the classification boundary obtained by a majority vote over these 50 trees; bottom right, the ROC curves for the single tree of Fig. 2 (red, broken solid line) and for the randomized ensemble (green, smooth dashed line) as estimated on a test sample of 4000 points. The classification boundary obtained with the ensemble is much smoother than each individual classification boundary. On this problem, the error rate (as estimated on a test sample of 4000 points) decreased from 10.2% with a single pruned tree to 5.4% with the ensemble. Also the AUC of ensembles of 50 trees is significantly better compared to the AUC of a single tree (0.98 as compared to 0.89).

Discussion. Although they often improve accuracy very significantly, the main drawback of ensemble methods is that they may jeopardize the interpretability and the efficiency with respect to the standard "single" decision tree method. In terms of interpretability, it is nevertheless still possible to rank the attributes using ensemble methods and, as a matter of fact, the resulting ranking is typically much more stable with an ensemble (see the next section). The negative effect on efficiency can also be mitigated. Indeed, randomization methods can be easily parallelized and also the introduction of randomization actually makes the growing of one random tree much faster than the growing of one standard tree. So, all in all, the computational costs of these methods is actually very similar to that of standard single trees. Boosting methods can not be so easily parallelized, but since they exploit trees of very low complexity their computational overhead remains nevertheless often acceptable.

Feature selection and ranking

One of the most interesting characteristics of decision trees, especially in the context of computational and systems biology,[26] is that they allow to carry out directly some feature selection, or in other words to identify among a large set of input attributes those that are really useful for solving the problem at hand. Indeed, because of the greedy nature of the tree induction algorithm, useless attributes will normally not be selected during the tree induction and, conversely, the attributes selected by the algorithm usually give a good indication of the most useful input variables for the problem at hand.

In addition to feature selection, it is possible to compute from a tree an attribute importance measure that allows to further rank the selected attributes according to their relevance for predicting the output. Several such importance measures have been proposed in the literature.[7,11,15,19,27] The simplest measure is derived from the impurity score in eqn (1). The overall importance of an attribute is defined as the sum of the total reduction of entropy $N_i\text{Score}(S_i,T_i)$ (where N_i is the size of S_i) over all nodes $i(a)$ in the tree where the attribute a is used to split. The attributes that are never selected obtain a null importance score, while those that are selected close to the root node typically obtain high scores.

One drawback of this measure is that it only takes into account the split that is effectively selected at each node. If two

variables have very similar scores at one node, one of them may be masked by the other and will thus receive a zero score although it could well be as important as the other one. In order to circumvent this effect, more sophisticated measures have been proposed that take into account the alternative unselected splits, called surrogate splits, at each node.[11]

Typically, ensembles of trees select a much larger proportion of the attributes and thus cannot be readily used for attribute selection. However, attribute importance measures can be easily extended to ensembles, simply by averaging importance scores over all trees in the ensemble. The resulting importance measure is even more reliable because of the variance reduction effect resulting from the averaging. It is also less prone to the masking effect of correlated attributes. In the context of Random Forests, Breiman[19] proposed another measure based on random permutations. In this method, the importance of an attribute is defined for each tree as the variation of its error rate when its values are randomly permuted in the validation set. The resulting importance is normalized to give a z-score from which a p-value is derived.

All these measures of importance provide interesting alternatives to attribute ranking based on the (adjusted) p-values obtained from classical statistical tests. The main advantage of these measures with respect to these statistical tests is that they do not make any assumption about the problem (such as gaussianity, linearity, or independence) and they are potentially able to detect multivariate effects, *i.e.* attributes that are only relevant through interaction with others. However, the tree-based importance measures are not yet as well principled as statistical tests, because their limitations and biases are not yet fully characterized, although research in this area is ongoing.[27–30]

Strengths and limitations

From the point of view of their statistical properties, tree-based methods (like other supervised learning methods) are non-parametric universal approximators, meaning that, with sufficient complexity, a tree can represent any continuous function with an arbitrary high precision. Under some very mild conditions, single tree induction algorithms are also consistent,[11] *i.e.* they yield models that converge to the best possible classifier for the problem at hand as the size of the learning sample tends to infinity (notice that this is not true for all methods based on ensembles of trees[31]). When used with numerical attributes, they are invariant with respect to monotone transformations of the input attributes.

From a more practical point of view, the main strength of tree-based methods is their interpretability. Single trees are readily understandable and interpretability is also gained through their ability to select or rank the attributes according to their relevance for predicting the output. They share this feature with almost no other non-parametric methods. With respect to other methods such as support vector machines (SVMs) or artificial neural networks, tree-based methods are also very easy to use. They only require to set the value of a very limited number of parameters and their accuracy is not overly sensitive to the setting of these parameters. The accuracy of ensemble methods such as Random Forests or boosting has been proved very competitive with other supervised learning methods. Their computational complexity at the learning stage is typically linear in the number of features and log linear in the number of objects, which is close to the lowest possible complexity one could expect for a learning algorithm. These methods are thus able to deal with very large datasets (*e.g.* of millions of objects and thousands of variables). Because of the greedy nature of the induction algorithm, their computing times are furthermore very predictable. These methods are also flexible: they can handle both continuous, discrete and even more complex structured input and output spaces. They are also easy to extend through the use of different sets of '*ad hoc*' candidate tests or node labeling schemes, which makes them useful in a wide range of applications.

These methods have nevertheless several limitations. The main drawback of single trees is their high variance: decision trees obtained from only slightly perturbed learning samples can be very different. This high variance makes the accuracy of single trees not competitive with other learning algorithms (*e.g.* neural networks). In addition, this instability reduces also their interpretability: one can not fully trust the choices of the tests if they are changing from one learning sample to another. In terms of accuracy, while ensemble methods with trees are competitive on many problems, fine-tuned support vector machines or artificial neural networks can sometimes reach higher levels of accuracy in some application domains.

One other possible drawback of ensemble methods is their high demand in terms of memory requirement for storing the model: the number of nodes of unpruned trees is typically of the same order as the number of objects in the datasets and thus the storage of an ensemble of trees might require more space than the original dataset, especially when there are many objects and few attributes. Even though decision tree tests can be easily extended, tree-based methods are still mostly restricted to the standard attribute-value representation, which makes them less suitable than, *e.g.*, kernel methods to handle very structured data types such as graphs or sequences.

To sum up, tree-based methods (especially in their ensemble flavor) are certainly methods of choice for many applications where simplicity, interpretability, and ease of use is required. Although their accuracy is often very good, in general slightly better results can nevertheless be obtained by fine-tuning other methods such as SVMs or artificial neural networks for instance, but this is then at the price of a lot of human intervention, a higher computational complexity, and a loss of interpretability and genericity.

Comparison with other supervised learning algorithms

In this section we compare a short list of some other popular supervised learning algorithms to decision tree-based methods. We refer the interested reader to the literature for further information about these other supervised learning methods.[6–8]

***k*-Nearest neighbors.** Probably the simplest and most intuitive learning algorithm, it assigns to an object an output value derived from those of its k nearest neighbors

(determined according to a distance measure computed from the inputs) in the learning sample. Although very simple, this method is sometimes very accurate. However, it does not provide explicitly a model and it is also very sensitive to the presence of irrelevant attributes (*i.e.* attributes which are not related to the output), contrary to tree-based methods. For large sample sizes it may also lead to prohibitive computing times at the moment of finding the nearest neighbors.

Naive Bayes. A classification method that estimates conditional class probabilities by applying Bayes' theorem under the (naive) assumption that the attribute values are independent given the class. Despite this strong assumption, Naive Bayes classifiers often work well on practical problems. They are highly scalable and rather interpretable. Their main limitation with respect to tree-based methods is their strong independence assumption and their inability to handle automatically numerical attributes without further parametric assumptions.

Linear models. The term linear model encompasses any method that uses as an hypothesis space the set of linear functions of the inputs (or, for classification, functions that correspond to a linear decision boundary in the input space). There exist many alternatives to learn linear models, that correspond to different statistical models or techniques to deal with the bias–variable tradeoff. Some examples are ridge regression, LASSO, or linear discriminant analysis.[7] These methods are especially well suited for high-dimensional problems. Like decision trees, they can also be used for attribute selection and ranking. Their main limitation with respect to tree-based methods is their inability to handle interactions of variables in an automatic way. Also, algorithms for inferring linear models are typically less scalable to very high dimensional input spaces.

Artificial neural networks. They are a generalization of linear models inspired by analogies with the brain. A neural network combines several simple linear units organized in layers through non-linear functions in order to improve expressiveness. These methods may be very accurate but they do not scale well with the number of inputs and are difficult to use for large scale practical problems. See ref. 5 for a review of these methods and their applications in bioinformatics.

Support vector machines. Loosely speaking, the support vector machine (SVM) method is a particular way to learn a linear classification model: the idea is to choose the hyperplane which is the furthest away from objects of each class, maximizing the so called "margin". One of the nice features of this algorithm is that its solution may be written only in terms of a similarity measure between the objects called a kernel (like the k-NN method). Using appropriate kernels, these methods can thus actually learn non-linear models and also handle naturally the complex data types that are often met in the context of computational biology. Compared with other methods, SVMs usually provide state-of-the-art results but like neural networks, they are essentially black-box models and not easy to use for non specialists. See ref. 4 for a review of these methods and their applications in computational biology.

Applications in computational and systems biology

Owing to their versatility, decision trees and, to a lesser extent, regression trees, have been applied to a wide variety of bioinformatic problems. Of particular interest for bioinformatics is the interpretability of decision trees, because the output of the algorithm must often be interpreted by biologists and validated in the laboratory. An additional advantage of decision trees in the context of bioinformatics is their ability to deal with heterogeneous data with relatively little preprocessing.

Following the evolution of machine learning methods, some bioinformatic problems have been re-analyzed using successive "generations" of decision tree methods. One conspicuous trend is the replacement of single tree methods by ensemble methods such as Random Forests. Another important trend is the increased emphasis on rigorous evaluation of prediction accuracy, using methods such as cross-validation.

In this section we provide a survey of the diversity of problems to which decisions tree-based methods have already been applied in the context of computational and systems biology. As dozens of articles have already been published in this context, we will not attempt to be comprehensive, but will rather focus on a few representative examples.

Sequence annotation

An early application of decision trees in bioinformatics was the identification of coding regions in eukaryotic genomic sequences using oblique decision trees and attributes related to the composition of the DNA sequences.[32]

The precise localization of the transcriptional start site is more challenging to predict, but good performance has been achieved by combining quadratic discriminant functions with decision trees.[33]

Decision trees have also been applied to the prediction of non-protein coding genes, for example Random Forests for the discovery of microRNA genes in *Drosophila* genomes.[34]

A second class of annotation problems is the inference of functions of unannotated genes and proteins. Kretschmann *et al.* used the C4.5 algorithm to infer protein functions in the SWISS-PROT database based on sequence patterns.[35] In a more recent application, a multi-label version of the C4.5 algorithm was used to assign functions to unannotated yeast genes, based on various data including sequence, expression patterns, knockout mutant experiments, PSI-BLAST homology searches and predicted secondary structure.[36]

Other applications of decision trees in protein sequence annotation include the prediction of membrane proteins,[37,38] of subcellular localization,[39] of ubiquitylation sites,[40] and of the specificity of the cleavage sites of viral proteases.[41]

Biomarker discovery

Biomarkers are biomolecules whose expression patterns are indicative of a biological condition of interest, such as a stage of differentiation or a disease. Examples of classes of biomarkers are mRNAs, proteins and peptides, metabolites, and microRNAs.

Decision trees are particularly suitable for biomarker discovery and tissue classification because they provide simple measures of attribute importance that can be used to rank biomarkers.

An important problem in genomics is the discrimination of cancer/non-cancer samples and the discrimination of sub-types of cancers based on gene expression patterns measured with microarrays. Ben-Dor *et al.* compared several methods for cancer classification, including nearest neighbor, decision stumps with boosting, and support vector machines (SVMs), and obtained comparably good accuracy with the SVMs and the boosted decision stumps.[42] Boulesteix *et al.* used a CART-based approach on expression microarray data to discover models of interactions between pairs of genes that are significantly correlated with cancer class.[43]

The goal is often to obtain a set of discriminating biomarkers that is as small as possible while remaining accurate, because this allows the development of cheap and convenient diagnostic tests. In this respect, decision trees are also advantageous because they often lead to compact decision models compared to other machine learning methods.[44] A striking example is the study by Chen *et al.* in which a decision tree model with five genes accurately predicted the survival of patients with non-small-cell lung cancer.[45]

The identification of biomarkers from biological fluids is a promising area of research, because it could detect medical conditions with minimally invasive procedures and without knowledge of the localization of the anomaly in the body. Mass spectrometry (MS) is the technique of choice for the characterization of proteins and peptides in these complex biological samples. Decision tree-based methods have been widely used in this domain.[46–54] For example, Qu *et al.* exploited boosted decision stumps to discriminate prostate cancer patients from noncancer patiens.[46] Geurts *et al.* compared several ensemble methods based on decision trees to the identification of serum biomarkers for rheumatoid arthritis and inflammatory bowel diseases in MS spectra.[50] Another study used decision trees to detect pancreatic cancer from serum biomarkers.[51]

Molecular interactions

A rich field of application of decision and regression trees in bioinformatics is the prediction of interactions between different types of biomolecules, such as protein–protein interactions (PPIs) and DNA–protein interactions.

Prediction of PPIs requires adjustment of the tree learning algorithms, because each attribute is measured twice, once for each protein of a pair. A number of papers have proposed to classify protein pairs with single decision trees[55] or Random Forests.[56,57] For example, in Chen and Liu,[57] a random forest classifier was trained in which protein pairs were represented by a vector of 4293 attributes corresponding to protein domains. These attributes have discrete values 0, 1 or 2, corresponding respectively to the absence of the domain in the pair, its presence in one of the proteins, or its presence in both.

Another possible approach for interaction prediction is to use attributes defined on single biomolecules to infer their positions in the network, and to deduce interactions from the network. Geurts *et al.* used an ensemble of regression trees and a kernelization of the output space to predict protein-protein interaction and metabolic networks.[58]

Supervised learning can also be used in the context of PPIs to clean and aggregate noisy experimental data. In order to identify new interactions in the yeast *Saccharomyces cerevisiae*, Krogan *et al.* performed a genome-wide purification of protein complexes, followed by protein identification by two complementary mass spectrometry procedures. To differentiate true from spurious PPIs in the noisy data, a classifier was built with a hand-curated database of PPIs as training set. This classifier combined the outputs of a Bayesian network, C4.5 decision trees and boosted stump decision trees to achieve best performance.[59]

Decision trees can also be used to infer PPIs among members of a particular class of proteins. In one study, correlated pairs of protein sequence motifs were extracted from known interacting transcription factor (TF) pairs. These motifs were then used to build a random forest predictor of TF interactions.[60]

Besides prediction of interacting biomolecules, decision trees have been used to predict the precise regions involved in the interaction in known interacting biomolecules. Random Forests have been used to predict residues involved in PPIs, based on combinations of sequence, physicochemical properties of the amino acids, evolutionary conservation and 3D structure.[61,62] Similar studies investigated prediction of amino acids binding DNA,[63] metal ions or small ligands.[64,65]

Genetics

Systematic knockout studies in model organisms such as the yeast *Saccharomyces cerevisiae* or the worm *Caenorhabditis elegans* have revealed that only a relatively small proportion of genes are essential for viability. However, the simultaneous mutation of two "non-essential" genes often confer a more strongly deleterious or a lethal phenotype, a type of genetic interaction known as sick and synthetic lethal (SSL) that may shed light on the basis of multigenic genetic traits and diseases. However, systematic generation of all double mutants in a model organism is extremely labor-intensive and decision tree methods have been applied to their computational identification, based on attributes defined on pairs of proteins.[66] More recently, Chipman and Singh combined random walks in biological networks with decision trees to identify gene pairs most likely to result in SSL interaction.[67]

Genome-wide association studies for complex diseases produce thousands of single nucleotide polymorphism (SNP) genotypes. A first step of the analysis is often to apply univariate statistical test of association to the individual SNPs. These test will fail to reveal SNP variants that have a small marginal effect on disease susceptibility, but a large effect when combined. Decision trees can take into account interactions among variables, and have been proposed as a method to identify interacting polymorphisms.[68–70]

The identification of genes likely to be associated with genetic diseases is also an application, related to sequence annotation, of

decision trees in molecular and statistical genetics. López-Bigas and Ouzounis have used decision trees to predict these genes based on protein sequence length and various measures of evolutionary conservation of the protein.[71]

Single nucleotide polymorphism (SNP) variants that are close together on a chromosome tend to be highly correlated because they are inherited together in blocks (haplotypes). It is thus advantageous to select a smaller set of SNPs (tagging SNPs) that convey the same information. The problem of finding the minimum set of tagging SNPs is NP-complete, but a fast algorithm based on decision trees has been developed that gives satisfactory results.[72]

More recently, Schlecht *et al.* used a hybrid machine learning approach including decision trees to infer patterns of SNP variants on sampled human Y chromosomes (haplogroups) from the patterns of another type of genetic markers, microsatellites.[73]

Discriminating between polymorphisms that have no functional effect (neutral polymorphisms), and functional ones is of crucial importance. Single nucleotide polymorphisms resulting in an amino acid substitution in the protein (non-synonymous SNPs) are of particular interest because they account for a large proportion of genetic diseases. Random Forests have been used to differentiate functional from neutral non-synonymous SNPs. Relevant input attributes include the physicochemical properties of the original and the substituted amino acids, their evolutionary conservation, and their structural environment in homologous protein structures.[74,75]

Another application of decision trees in molecular genetics is the prediction of antiretroviral drug resistance of HIV isolates from their sequence. In Beerenwinkel *et al.*, C4.5 decision trees were used to build interpretable models of drug resistance of the viral protease and reverse-transcriptase based on the protein sequence.[76]

Regulatory networks

The accurate modeling of regulatory networks is one of the grand goals of systems biology, and many modeling schemes have been proposed, from comparatively simple Boolean networks to complex models based on stochastic differential equations.[77]

Decision and regression trees provide suitable approaches to describe the structure of regulatory networks. For example, Soinov *et al.* used decision trees to predict the Boolean (on/off) expression levels of genes based the continuous expression values of other genes.[78] Phuong *et al.* used multiple output regression trees to predict gene expression levels from the presence of motifs in promoter sequences.[79] More sophisticated methods based on generalizations of decision trees[80,81] and of regression trees[82] have been proposed to model simultaneously the expression levels of target genes, of their regulators, and the presence of motifs in the promoters of the genes.

Segal *et al.* applied a probabilistic method to identify gene regulatory modules, *i.e.* sets of genes co-regulated in different conditions. In this approach, the logic of modules is modeled by regression trees, and the best modules and assignments of genes to modules are searched iteratively with the expectation maximization (EM) algorithm.[83]

Chen and Blanchette[84] modeled tissue-specific cis-regulatory modules using Bayesian networks. They addressed the problem of learning conditional probabilities of combinations of TFs by using regression trees as a compact representation of these probabilities.

Structural biology

Structural biology is another field of application of decision trees. The prediction of protein secondary structure has a long history in bioinformatics, and numerous algorithms have been proposed to solve this problem. For example, Selbig *et al.* used decision trees to combine the predictions of 10 existing methods, thereby outperforming all of these individual methods.[85]

Tree-based methods have also been used to predict specific types of structural elements in proteins. Siepen *et al.*[86] used a pruned C4.5 decision tree and a support vector machine to predict exposed edges of β-strands, a structural element that has been linked to protein aggregation and neurodegenerative disorders. The two methods achieved the same prediction accuracy, but they remarked that the interpretable rules generated by the decision tree were in close agreement with knowledge about protein structure. In another study, two tree-based algorithms were used for the prediction of the DNA-binding structural motif helix-turn-helix. A single decision tree model was used to get insight on the structural features associated with DNA-binding, while a more accurate but less interpretable model based on boosted (with AdaBoost) decision stumps was used to scan a whole database to find new proteins of this class.[87]

Image and video analysis

More and more, the acquisition of images or videos becomes a tool of choice for recording experiments in the context of systems biology. Indeed, in the recent literature, one can observe a significant trend of works exploiting various kinds of image acquisition and image analysis techniques in systems biology but also in medicine.[88–90]

One of the main problems in this context is to extract quantitative information from images and videos in a reproducible and automatic way, so as to allow statistical assessments of the significance of biological effects. For example, methods need to be designed for the automatic detection, classification, counting, quantification, and monitoring of various biological entities (molecules, cells, tissues) within such images. Since there is a large diversity of problems stemming from the diversity of biological objects and imaging modalities, machine learning approaches are a method of choice to design such systems in an effective and mostly automated way.

This field of applications is only at its very early stage of development but has a great future. To give some ideas about the possibilities, we refer the reader to some recent publications in this field which are based on the use of decision tree-based methods.[91–93]

Acknowledgements

The authors thank Raphaël Marée and Tom Druet for comments on the manuscript. PG is Research Associate of the F.R.S.-FNRS. This paper presents research results of the Belgian Network BIOMAGNET (Bioinformatics and Modeling: from Genomes to Networks), funded by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office. This work is supported by the European Network of Excellence PASCAL2. The scientific responsibility rests with the authors.

References

1 C. Kingsford and S. L. Salzberg, *Nat. Biotechnol.*, 2008, **26**, 1011–3.
2 P. Larrañaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armañanzas, G. Santafé, A. Pérez and V. Robles, *Briefings Bioinf.*, 2006, **7**, 86–112.
3 A. L. Tarca, V. J. Carey, X.-w. Chen, R. Romero and S. Draghici, *PLoS Comput. Biol.*, 2007, **3**, e116.
4 A. Ben-Hur, C. S. Ong, S. Sonnenburg, B. Schölkopf and G. Rätsch, *PLoS Comput. Biol.*, 2008, **4**, e1000173.
5 L. Lancashire, C. Lemetre and G. Ball, *Briefings Bioinf.*, 2009, **10**, 315–329.
6 R. Duda, P. Hart and D. Stork, *Pattern Classification*, John Wiley & Sons, New York, 2nd edn, 2000.
7 T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer, New York, 2001.
8 C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2006.
9 C. Ambroise and G. McLachlan, *Proc. Natl. Acad. Sci. U. S. A.*, 2002, **99**, 6562–6566.
10 T. Fawcett, *Pattern Recognit. Lett.*, 2006, **27**, 861–874.
11 L. Breiman, J. Friedman, R. Olsen and C. Stone, *Classification and Regression Trees*, Wadsworth International, California, 1984.
12 J. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, 1986.
13 E. Hunt, *Concept Learning*, Wiley, New York, 1962.
14 J. Quinlan, *Mach. Learn.*, 1986, **1**, 81–106.
15 L. Wehenkel, *Automatic Learning Techniques in Power Systems*, Kluwer Academic, Boston, 1998.
16 F. Provost and P. Domingos, *Mach. Learn.*, 2003, **52**, 199–215.
17 J. Mingers, *Mach. Learn.*, 1989, **4**, 227–243.
18 L. Breiman, *Mach. Learn.*, 1996, **24**, 123–140.
19 L. Breiman, *Mach. Learn.*, 2001, **45**, 5–32.
20 P. Geurts, D. Ernst and L. Wehenkel, *Mach. Learn.*, 2006, **63**, 3–42.
21 Y. Freund and R. E. Schapire, *Proceedings of the Second European Conference on Computational Learning Theory*, 1995, pp. 23–27.
22 H. Drucker and C. Cortes, *Advances in Neural Information Processing Systems*, 1996, pp. 479–485.
23 E. Bauer and R. Kohavi, *Mach. Learn.*, 1999, **36**, 105–139.
24 R. Meir and G. Rätsch, in *Advanced Lectures on Machine Learning*, Springer, 2003, ch. An introduction to Boosting and Leveraging.
25 J. Friedman, T. Hastie and R. Tibshirani, *Ann. Stat.*, 2000, **28**, 337–374.
26 Y. Saeys, I. Inza and P. Larranaga, *Bioinformatics*, 2006, **23**, 2507–2517.
27 C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin and A. Zeileis, *BMC Bioinformatics*, 2008, **9**, 307.
28 C. Strobl, A. Boulesteix, A. Zeileis and T. Hothorn, *BMC Bioinformatics*, 2007, **8**, 25.
29 K. Archer and R. Kimes, *Comput. Stat. Data Anal.*, 2008, **52**, 2249–2260.
30 V. Huynh-Thu, L. Wehenkel and P. Geurts, *JMLR Workshop and Conference proceedings*, 2008, **4**, 60–73.
31 G. Biau, L. Devroye and G. Lugosi, *J. Mach. Learn. Res.*, 2008, **9**, 2015–2033.
32 S. Salzberg, *J. Comput. Biol.*, 1995, **2**, 473–485.
33 R. V. Davuluri, I. Grosse and M. Q. Zhang, *Nat. Genet.*, 2001, **29**, 412–417.
34 A. Stark, P. Kheradpour, L. Parts, J. Brennecke, E. Hodges, G. J. Hannon and M. Kellis, *Genome Res.*, 2007, **17**, 1865–1879.
35 E. Kretschmann, W. Fleischmann and R. Apweiler, *Bioinformatics*, 2001, **17**, 920–926.
36 A. Clare and R. D. King, *Bioinformatics*, 2003, **19 Suppl 2**, ii42–ii49.
37 M. M. Gromiha and Y. Yabuki, *BMC Bioinformatics*, 2008, **9**, 135.
38 J. Y. Yang, M. Q. Yang, A. K. Dunker, Y. Deng and X. Huang, *BMC Genomics*, 2008, **9**(suppl 1), S7.
39 Y. Q. Shen and G. Burger, *BMC Bioinformatics*, 2007, **8**, 420.
40 C.-W. Tung and S.-Y. Ho, *BMC Bioinformatics*, 2008, **9**, 310.
41 Z. R. Yang, *Bioinformatics*, 2005, **21**, 2644–2650.
42 A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer and Z. Yakhini, *J. Comput. Biol.*, 2000, **7**, 559–583.
43 A.-L. Boulesteix, G. Tutz and K. Strimmer, *Bioinformatics*, 2003, **19**, 2465–2472.
44 R. Díaz-Uriarte and S. A. de Andrés, *BMC Bioinformatics*, 2006, **7**, 3.
45 H.-Y. Chen, S.-L. Yu, C.-H. Chen, G.-C. Chang, C.-Y. Chen, A. Yuan, C.-L. Cheng, C.-H. Wang, H.-J. Terng, S.-F. Kao, W.-K. Chan, H.-N. Li, C.-C. Liu, S. Singh, W. J. Chen, J. J. W. Chen and P.-C. Yang, *N. Engl. J. Med.*, 2007, **356**, 11–20.
46 Y. Qu, B.-L. Adam, Y. Yasui, M. D. Ward, L. H. Cazares, P. F. Schellhammer, Z. Feng, O. J. Semmes and G. L. Wright, *Clin. Chem.*, 2002, **48**, 1835–1843.
47 H. Liu, J. Li and L. Wong, *Genome Informatics*, 2002, **13**, 51–60.
48 B. Wu, T. Abbott, D. Fishman, W. McMurray, G. Mor, K. Stone, D. Ward, K. Williams and H. Zhao, *Bioinformatics*, 2003, **19**, 1636–1643.
49 G. Izmirlian, *Ann. N. Y. Acad. Sci.*, 2004, **1020**, 154–174.
50 P. Geurts, M. Fillet, D. de Seny, M.-A. Meuwis, M. Malaise, M.-P. Merville and L. Wehenkel, *Bioinformatics*, 2005, **21**, 3138–3145.
51 Y. Yu, S. Chen, L.-S. Wang, W.-L. Chen, W.-J. Guo, H. Yan, W.-H. Zhang, C.-H. Peng, S.-D. Zhang, H.-W. Li and G.-Q. Chen, *Oncology*, 2005, **68**, 79–86.
52 J. Cui, X. Kang, Z. Dai, C. Huang, H. Zhou, K. Guo, Y. Li, Y. Zhang, R. Sun, J. Chen, Y. Li, Z. Tang, T. Uemura and Y. Liu, *J. Cancer Res. Clin. Oncol.*, 2007, **133**, 825–834.
53 Y. Su, J. Shen, H. Qian, H. Ma, J. Ji, H. Ma, L. Ma, W. Zhang, L. Meng, Z. Li, J. Wu, G. Jin, J. Zhang and C. Shou, *Cancer Sci.*, 2007, **98**, 37–43.
54 Y.-S. Wei, Y.-H. Zheng, W.-B. Liang, J.-Z. Zhang, Z.-H. Yang, M.-L. Lv, J. Jia and L. Zhang, *Cancer*, 2008, **112**, 544–551.
55 L. Zhang, S. Wong, O. King and F. Roth, *BMC Bioinformatics*, 2004, **5**, 15.
56 Y. Qi, J. Klein-Seetharaman and Z. Bar-Joseph, *Pacific Symposium of Biocomputing*, 2005.
57 X.-W. Chen and M. Liu, *Bioinformatics*, 2005, **21**, 4394–4400.
58 P. Geurts, N. Touleimat, M. Dutreix and F. d'Alché Buc, *BMC Bioinformatics*, 2007, **8**(suppl 2), S4.
59 N. J. Krogan, G. Cagney, H. Yu, G. Zhong, X. Guo, A. Ignatchenko, J. Li, S. Pu, N. Datta, A. P. Tikuisis, T. Punna, J. M. Peregrín-Alvarez, M. Shales, X. Zhang, M. Davey, M. D. Robinson, A. Paccanaro, J. E. Bray, A. Sheung, B. Beattie, D. P. Richards, V. Canadien, A. Lalev, F. Mena, P. Wong, A. Starostine, M. M. Canete, J. Vlasblom, S. Wu, C. Orsi, S. R. Collins, S. Chandran, R. Haw, J. J. Rilstone, K. Gandi, N. J. Thompson, G. Musso, P. S. Onge, S. Ghanny, M. H. Y. Lam, G. Butland, A. M. Altaf-Ul, S. Kanaya, A. Shilatifard, E. O'Shea, J. S. Weissman, C. J. Ingles, T. R. Hughes, J. Parkinson, M. Gerstein, S. J. Wodak, A. Emili and J. F. Greenblatt, *Nature*, 2006, **440**, 637–643.
60 A. D. J. van Dijk, C. J. F. ter Braak, R. G. Immink, G. C. Angenent and R. C. H. J. van Ham, *Bioinformatics*, 2008, **24**, 26–33.
61 X. wen Chen and J. C. Jeong, *Bioinformatics*, 2009, **25**, 585–591.
62 M. Šikić, S. Tomić and K. Vlahoviček, *PLoS Comput. Biol.*, 2009, **5**, e1000278.
63 J. Wu, H. Liu, X. Duan, Y. Ding, H. Wu, Y. Bai and X. Sun, *Bioinformatics*, 2009, **25**, 30–35.
64 A. J. Bordner, *Bioinformatics*, 2008, **24**, 2865–2871.
65 M. Lippi, A. Passerini, M. Punta, B. Rost and P. Frasconi, *Bioinformatics*, 2008, **24**, 2094–2095.

66 S. L. Wong, L. V. Zhang, A. H. Y. Tong, Z. Li, D. S. Goldberg, O. D. King, G. Lesage, M. Vidal, B. Andrews, H. Bussey, C. Boone and F. P. Roth, *Proc. Natl. Acad. Sci. U. S. A.*, 2004, **101**, 15682–15687.
67 K. C. Chipman and A. K. Singh, *BMC Bioinformatics*, 2009, **10**, 17.
68 K. L. Lunetta, L. B. Hayward, J. Segal and P. V. Eerdewegh, *BMC Genet.*, 2004, **5**, 32.
69 R. Jiang, W. Tang, X. Wu and W. Fu, *BMC Bioinformatics*, 2009, **10**(suppl 1), S65.
70 S. S. F. Lee, L. Sun, R. Kustra and S. B. Bull, *Bioinformatics*, 2008, **24**, 1603–1610.
71 N. López-Bigas and C. A. Ouzounis, *Nucleic Acids Res.*, 2004, **32**, 3108–3114.
72 P. Zhang, H. Sheng and R. Uehara, *BMC Bioinformatics*, 2004, **5**, 89.
73 J. Schlecht, M. E. Kaplan, K. Barnard, T. Karafet, M. F. Hammer and N. C. Merchant, *PLoS Comput. Biol.*, 2008, **4**, e1000093.
74 L. Bao and Y. Cui, *Bioinformatics*, 2005, **21**, 2185–2190.
75 J. Hu and C. Yan, *BMC Bioinformatics*, 2008, **9**, 297.
76 N. Beerenwinkel, B. Schmidt, H. Walter, R. Kaiser, T. Lengauer, D. Hoffmann, K. Korn and J. Selbig, *Proc. Natl. Acad. Sci. U. S. A.*, 2002, **99**, 8271–8276.
77 T. Schlitt and A. Brazma, *BMC Bioinformatics*, 2007, **8**(suppl 6), S9.
78 L. A. Soinov, M. A. Krestyaninova and A. Brazma, *GenomeBiology*, 2003, **4**, R6.
79 T. M. Phuong, D. Lee and K. H. Lee, *Bioinformatics*, 2004, **20**, 750–757.
80 M. Middendorf, A. Kundaje, C. Wiggins, Y. Freund and C. Leslie, *Bioinformatics*, 2004, **20**(suppl_1), i232–i240.
81 A. Kundaje, M. Middendorf, M. Shah, C. H. Wiggins, Y. Freund and C. Leslie, *BMC Bioinformatics*, 2006, **7**(suppl 1), S5.
82 J. Ruan and W. Zhang, *Bioinformatics*, 2006, **22**, 332–340.
83 E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein, D. Koller and N. Friedman, *Nat. Genet.*, 2003, **34**, 166–176.
84 X. Chen and M. Blanchette, *BMC Bioinformatics*, 2007, **8**(suppl 10), S2.
85 J. Selbig, T. Mevissen and T. Lengauer, *Bioinformatics*, 1999, **15**, 1039–1046.
86 J. A. Siepen, S. E. Radford and D. R. Westhead, *Protein Sci.*, 2003, **12**, 2348–59.
87 W. A. McLaughlin and H. M. Berman, *J. Mol. Biol.*, 2003, **330**, 43–55.
88 S. G. Megason and S. E. Fraser, *Cell*, 2007, **130**, 784–95.
89 A. R. Kherlopian, T. Song, Q. Duan, M. A. Neimark, M. J. Po, J. K. Gohagan and A. F. Laine, *BMC Syst. Biol.*, 2008, **2**, 74.
90 H. Peng, *Bioinformatics*, 2008, **24**, 1827–1836.
91 G. Giannone, B. Dubin-Thaler, O. Rossier, Y. Cai, O. Chaga, G. Jiang, W. Beaver, H. Dobereiner, Y. Freund, G. Borisy and M. Sheetz, *Cell*, 2007, **128**, 561–575.
92 R. Marée, P. Geurts and L. Wehenkel, *BMC Cell Biol.*, 2007, **8**(suppl 1), S2.
93 R. Liu, Y. Freund and G. Spraggon, *Acta Crystallogr., Sect. D: Biol. Crystallogr.*, 2008, **64**, 1187–95.