BRIEFINGS IN BIOINFORMATICS. VOL 14. NO 5. 633–647
Advance Access published on 4 April 2013
doi:10.1093/bib/bbt020

Pattern recognition in bioinformatics

Dick de Ridder, Jeroen de Ridder and Marcel J. T. Reinders

Submitted: 14th December 2012; Received (in revised form): 5th March 2013

Abstract

Pattern recognition is concerned with the development of systems that learn to solve a given problem using a set of example instances, each represented by a number of features. These problems include clustering, the grouping of similar instances; classification, the task of assigning a discrete label to a given instance; and dimensionality reduction, combining or selecting features to arrive at a more useful representation. The use of statistical pattern recognition algorithms in bioinformatics is pervasive. Classification and clustering are often applied to high-throughput measurement data arising from microarray, mass spectrometry and next-generation sequencing experiments for selecting markers, predicting phenotype and grouping objects or genes. Less explicitly, classification is at the core of a wide range of tools such as predictors of genes, protein function, functional or genetic interactions, etc., and used extensively in systems biology. A course on pattern recognition (or machine learning) should therefore be at the core of any bioinformatics education program. In this review, we discuss the main elements of a pattern recognition course, based on material developed for courses taught at the BSc, MSc and PhD levels to an audience of bioinformaticians, computer scientists and life scientists. We pay attention to common problems and pitfalls encountered in applications and in interpretation of the results obtained.

Keywords: *bioinformatics; pattern recognition; clustering; classification; dimensionality reduction*

INTRODUCTION

Over the past two decades, techniques used for a variety of molecular measurements have dramatically improved: cost has dropped, and throughput has increased by many orders of magnitude. Examples include the microarray for measuring transcripts or genotyping [1]; mass spectrometry for protein and metabolite levels [2]; and, most recently, next-generation sequencing for genomics and transcriptomics [3]. The resulting enormous increase in the volumes of data produced has likely been the most important factor underlying the rapid growth of bioinformatics over the past decade, as dedicated computational tools are essential to help handle, manipulate and make sense of it [4].

Where traditionally, measurements are used to test pre-defined (a priori) hypotheses, a more data-driven approach is increasingly taken in which high-throughput measurement data are mined to answer a posteriori formulated questions [5]. For example, which of the 50 000 probes on a microarray show differential expression between two conditions? Which genomic variation is predictive of a certain phenotype? Such questions often share the fact that a theoretical model of the system studied is lacking (or too complex to formulate), yet a large amount of noisy measurement data are available. The core challenge is therefore generalization, i.e. deriving approximate predictive 'black-box' models for a biological phenomenon [6].

This development has led to an increasing use of techniques from the computer science discipline of artificial intelligence [7]: data mining, concerned with the analysis of large unstructured datasets, with an emphasis on finding recurrent patterns [8]; machine learning, focusing on efficient algorithms

Corresponding author. Dick de Ridder, Delft Bioinformatics Lab, Department of Intelligent Systems, Faculty of Electrical Engineering, Mathematics & Computer Science, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands. Tel.: +31 15 2785114; Fax: +31 15 2787022; E-mail: d.deridder@tudelft.nl

Dick de Ridder is an associate professor in the Delft Bioinformatics Lab, working on learning-based algorithms and models for microbial systems biology and synthetic biology.

Jeroen de Ridder is an assistant professor in the Delft Bioinformatics Lab. His research interest is pattern recognition in high-throughput molecular measurements in human disease.

Marcel J. T. Reinders is a professor in bioinformatics and heads the Pattern Recognition & Bioinformatics group that deals with pattern recognition, computer vision and bioinformatics. He is particularly interested in data-driven approaches toward computational biology.

to handle a variety of problems on (mostly) large-scale structured data [9]; and pattern recognition [10–13], taking a more engineering-based approach: how should data be modeled and algorithms be developed to effectively answer a certain question, given a set of measurements on objects and, optionally, a set of matching desired outputs (labels)?

Pattern recognition (related to, but not the same as, pattern matching, looking for specified patterns in sequences or pattern detection, looking for novel patterns in sequences) has been remarkably successful in helping to explore and exploit high-throughput measurement data. A good example is a seminal article on microarray-based breast cancer prognosis [14], which showed that dimensionality reduction and classification give insight into the molecular mechanisms underlying prognosis, but also aid translational medicine; microarray-based tests based on this work are now routinely used in the clinic [15]. More recently, the ENCODE consortium [16] depended heavily on a variety of classification and clustering algorithms in studies investigating the interplay between transcription, transcription factor binding, histone modification, chromatin accessibility, etc. [17].

As the analysis of high-throughput data becomes increasingly central to modern biology, it is essential that bioinformatics students learn about pattern recognition/machine learning. In this article, we review pattern recognition basics that should be part of any course on the topic. As space is limited, we will not supply much mathematical detail, but instead refer to a number of excellent textbooks on pattern recognition for further reading. We will focus on the generic principles underlying available methods and the main pitfalls in their application, rather than on the strengths and weaknesses of individual algorithms because correct application and particularly interpretation of results is often the hardest issue for pattern recognition students. We conclude with a brief overview of recent developments and give a list of available generic software tools for pattern recognition. Box 1 defines the key terms used throughout this article.

PATTERN RECOGNITION

Since its birth in the 1950s, a number of different views on pattern recognition have been taken, focusing on syntax, structure or statistics of the data studied. Statistical pattern recognition has become the predominant paradigm [18] and is mainly concerned with developing theory and methods for

- clustering—do objects form natural groups?
- dimensionality reduction—can we extract informative features out of the measurements?
- classification—can we predict labels of new objects?

Figure 1 illustrates these three problems. Regression, similar to classification but concerned with the prediction of real-valued outputs, is not considered as widely as the problems above [12, 13]. Note that there is a clear distinction between supervised and unsupervised pattern recognition problems. In the former, data extraneous to the measurement process, such as labels, are available, and in the latter, they are not. Clustering is an unsupervised problem, classification a supervised one and dimensionality reduction can be cast in both forms.

Pattern recognition applications follow a pattern recognition pipeline, a number of computational analysis steps taken to achieve the goal [11]. Figure 2 illustrates this for classification. The starting point of any application is the collection of a set of training objects, assumed to be representative of the problem at hand and thus for new objects to which the system will be applied later. The first stages then consist of translating raw measurements into data usable for further processing. Some pre-processing (steps A–B in Figure 2) is handled by measurement devices or accompanying software itself: next-generation sequencers deliver base calls (and quality estimates) extracted from raw trace data, and microarray scans are often normalized and summarized using device-specific algorithms, etc. Further pre-processing is usually specific to the problem at hand and depends on available prior knowledge. Quality inspection is also important: to avoid problems in subsequent analyses, erroneous measurements (outliers) should be detected and removed [19], and missing values imputed [20]. After pre-processing, measurements are adequately represented, usually as features (C). Then a subset of informative features is selected (D) and used to train a classifier, i.e. to set the parameters of an algorithm that predicts a label, given a set of measurements (E). Finally, the performance of the classifier is evaluated on test data not used before (F). In the remainder of this review, we will focus on steps C–F.

Box 1: A glossary of key terms used throughout this review

Object: the basic element of study, for example, a patient

Measurement: raw data obtained from an object, for example, a microarray probe intensity

Feature (vector): a (vector of) pre-processed and/or selected measurement(s), for example, gene expression level(s)

Label: the class membership of the object, to be predicted in classification, for example, 'diseased' or 'healthy'

(un)Supervised: problems in which the label does (not) play a role

Training: estimating the parameters of a pattern recognition model using a set of representative objects, the training set

Testing: evaluating the performance of a trained supervised pattern recognition algorithm

Prior knowledge: any knowledge about the problem that can be exploited to help solve it more efficiently and/or effectively

Clustering: grouping objects in a dataset based on some criterion, such as similarity

Dimensionality reduction: reducing an initial set of features into a smaller set of informative ones

Feature selection: selecting (combinations of) individual features based on some criterion, for example, the *t*-statistic

Feature extraction: combining features, linearly or nonlinearly, into new ones based on some criterion, for example, object distance preservation

Classification: predicting a label for an object, given its features, for example, diagnosing a patient, given a microarray measurement

(non)Parametric density estimation: estimating a probability density function (not) assuming it has a certain form

Prior probability: the probability of finding a certain label, before doing any measurements; for example, the probability of encountering a patient with a certain disease

Class-conditional probability: the probability distribution of feature value(s) of an object given that it has a certain label

Posterior probability: the probability distribution of an object having a certain label given its feature value(s)

Kernel: a function representing a kind of similarity between two objects, corresponding to an inner product between two feature vectors in some space; used in the kernel trick as replacement for inner products to turn linear algorithms into nonlinear ones

Decision boundary: line(s) dividing the feature space into parts in which all objects are assigned the same label

Discriminant function: a function fit directly to the feature values without estimating probability distributions; can be thresholded to obtain a decision boundary

Prediction error: the number of misclassifications by a classifier in testing

ROC curve: the receiver-operator characteristic curve, showing the trade-off between the ratio of false positives and false negatives in testing a classifier

AUC: the area under the ROC curve, a performance measure for a classifier

Precision: the fraction of true positives in all objects labeled positive by a classifier (also known as sensitivity or true positive rate)

Recall: the fraction of all positive objects labeled positive by classifier

Cross-validation: a method to evaluate performance, by dividing the objects into a number of subsets, iteratively holding out one subset for testing and training on the remaining part

Generalization: the ability to predict well for new unseen objects

Overfitting: adapting too much to the training set at hand, leading to poor performance on new objects

Model selection: the problem of selecting a model of appropriate complexity for a given problem

Peaking phenomenon: the problem that for a fixed number of training objects, adding features at some point degrades performance

Small sample size problem: the problem of generalizing based on a small set of training objects

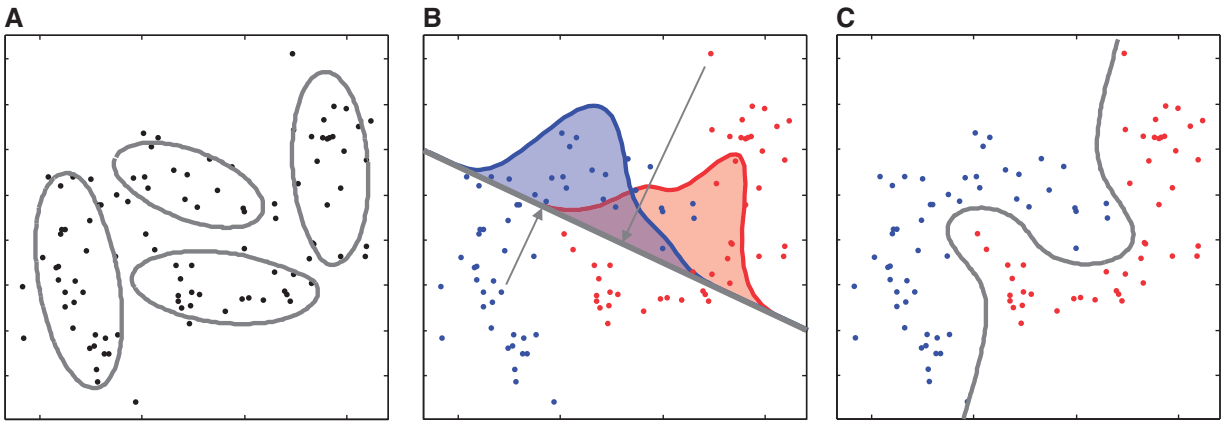

Figure 1: Three types of analysis illustrated on a dataset with two classes and two measurements. (**A**) Clustering: using a mixture of Gaussians, four clusters are fitted to the data. (**B**) Dimensionality reduction: in linear discriminant analysis, the data are projected on a line such that the classes are separated as well as possible. (**C**) Classification: the decision boundary of a support vector classifier (3rd degree polynomial kernel) separates the two classes.

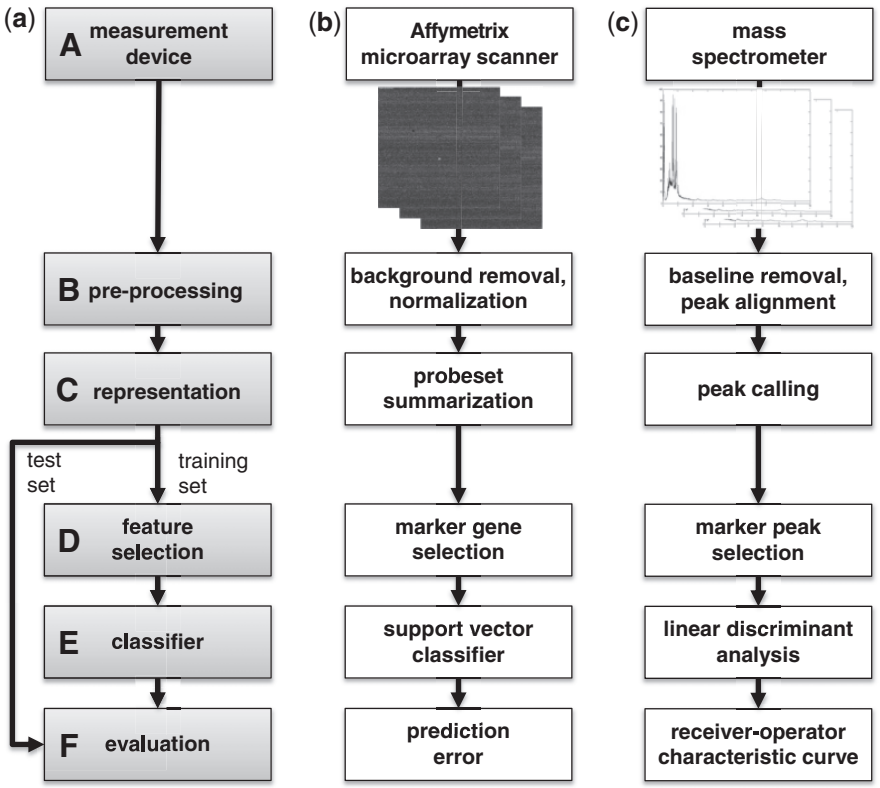

Figure 2: (**A**) The pattern recognition pipeline for classification. (**B** and **C**) Two examples in bioinformatics.

DATA REPRESENTATION

In step C, the measurements are represented in a format suitable for further processing. Having a good representation is perhaps the most important step toward satisfactorily solving a pattern recognition problem; no matter how advanced the algorithms in later stages, if too much information is lost in this step, good performance will be impossible to obtain. Most pattern recognition algorithms assume that an object is represented by a feature vector $\mathbf{x}$ of real-valued numbers. Often, this is straightforward: for microarray data, a vector of gene expression levels represents an object. Additional real-valued data, for example, clinical

data, can easily be added to such a feature vector. In some cases, prior knowledge on the problem or characteristics of the classifier make it useful to scale (e.g. to zero mean, unit standard deviation), transform (e.g. taking a log or square root) or combine (e.g. add or multiply) elements in feature vectors before further processing.

The representation of binary (0/1), ordinal (1/2/3/...), qualitative ('red'/'green'/'blue'), sequential ('ACTGAATA') or structural data is more complicated; either it is simply interpreted as real values (which usually leads to a loss of information) or it needs to be converted into a more suitable format. It is often natural to use a relative rather than an absolute representation of such data [21]. For example, while it is hard to represent a gene sequence by (a vector of) real values, it is quite natural to represent it by a vector of dissimilarity measures (e.g. BLAST *E*-values) to a set of representative sequences [22]. Likewise, gene expression profiles can be represented as correlations to expression profiles of other genes. Some algorithms, such as the nearest neighbor classifier, which assigns a test object the label of the most similar objects in the training set, can directly use such dissimilarities as input.

A relatively new development, driven to a large extent by applications in bioinformatics, is that of representing object pairs (X,Y) by kernels $K(X,Y)$ [23]. Kernels are functions that, under certain constraints, allow relatively simple algorithms to operate on complex representations of objects in high-dimensional feature spaces, without the need of explicitly calculating those representations. The development of the support vector classifier [24], which derives much of its power in applications from using kernels ('Classification' section), was followed by a large number of proposals for kernels and kernel-based methods for bioinformatics. For objects X,Y represented by feature vectors $\mathbf{x}$ and $\mathbf{y}$, there are, for example, polynomial kernels $K(\mathbf{x},\mathbf{y}) = (\mathbf{x}^{\mathrm{T}}\mathbf{y} + c)^d$ (the inner product plus a constant c raised to the power d) and radial basis function kernels, for example, the Gaussian $K(\mathbf{x},\mathbf{y}) = \exp(-c\|\mathbf{x}-\mathbf{y}\|^2)$.

However, kernels can also be defined for objects with other representations. For sets, e.g. of functional annotations or words in documents, there are set kernels; spectrum, motif and local alignment kernels are used to represent protein and gene sequences; *P*-kernels and Fisher kernels allow comparison of probabilistic models such as hidden Markov models (HMMs); and using graph and tree kernels, even complex structures can be compared [25]. Additionally, kernels representing different aspects of an object (say, a vector of concentration measurements, a set of functional annotations and a sequence of a protein) can easily be combined into a single kernel by simply adding the output of the individual kernel functions [26]. This makes kernel representations very powerful in bioinformatics.

PATTERN RECOGNITION ALGORITHMS

Clustering

A first natural question to ask when exploring a large dataset is whether it clusters, i.e. contains distinct subgroups of similar objects. Such groups may lead to new insights [27]. For example, genes whose expressions over a range of microarray experiments are similar may be functionally related, and the 'guilt-by-association' principle can be used to infer the function of uncharacterized genes. Likewise, proteins are clustered based on sequence similarity to learn about orthology, organisms based on genome similarity to find phylogenetic trees, patient samples based on high-throughput measurements to discover distinct disease subtypes, etc.

There are two main types of clustering algorithms: partitional and hierarchical. In the first, a certain (simple) cluster model is assumed, and the fit of a number of such models to the data is optimized. Examples of partitional methods are *k*-means and Gaussian mixture models (GMMs; Figure 1a), but there is a large body of literature on different clustering methods [28]. In hierarchical clustering, in contrast, a dendrogram is constructed, usually by iteratively grouping objects and clusters that are most similar (Figure 3a). This dendrogram can subsequently be cut at a certain level to end up with a specific number of clusters (Figure 3b). Hierarchical clustering has become popular in the microarray era to help create heatmaps [29].

Irrespective of the type of algorithm used, a user defines or assumes—perhaps implicitly—what number of clusters to look for, when objects are similar (e.g. using dissimilarity measures discussed in 'Data Representation' section), and/or what constitutes a cluster (e.g. in GMMs, a group of objects following a normal distribution). Similar assumptions can be used to estimate the number of clusters [30]

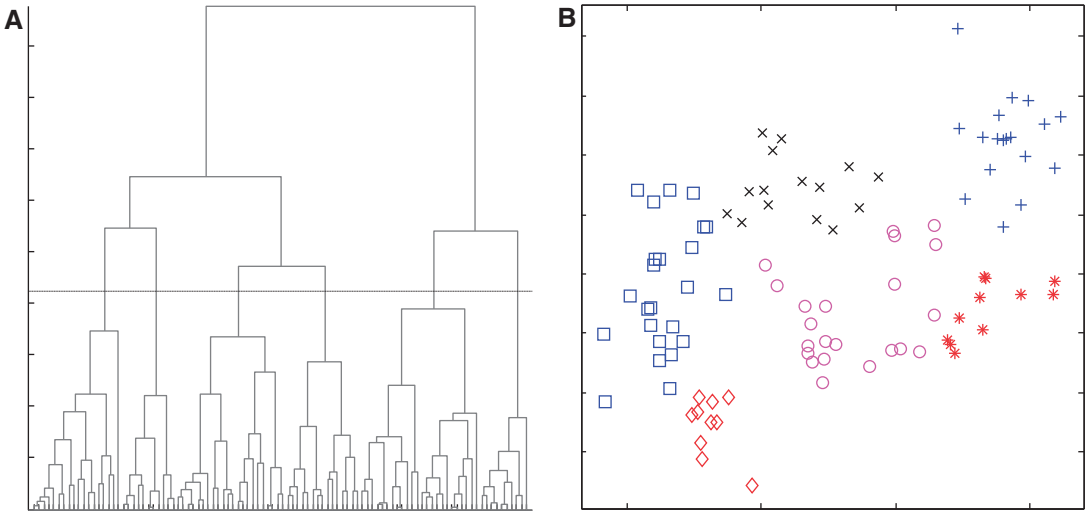

Figure 3: Hierarchical clustering. (**A**) The dendrogram joins objects and clusters; the height of the stem indicates their distance. (**B**) The clustering resulting when the dendrogram is cut at the dotted line.

or to evaluate a clustering [31]. It is important to realize that different assumptions can lead to very different groupings. A clustering is therefore never an objective result: finding that objects group is not proof (yet) of a relation, but at most a starting point for further experimentation.

A source of confusion is the distinction between clusters and classes, i.e. groups of objects that share a label. These can be quite distinct: a cluster may contain various classes if classes overlap; vice versa, a class may consist of various clusters, for example, when a single phenotypic label (e.g. 'diseased') is assigned to objects that show heterogeneity in the underlying molecular causes (e.g. various forms of that disease). As such, the term 'unsupervised classification' sometimes used for clustering is somewhat of a misnomer.

Dimensionality reduction

Where clustering summarizes the dataset by combining objects, likewise techniques are available to find a smaller number of features that adequately represent the original p measurements. An important application is visualization, reducing the number of features to two or three [32]. Dimensionality reduction is especially useful in determining sparse feature representations by discarding noisy or irrelevant measurements or by combining relevant measurements into a smaller number of features. This ensures fewer parameters have to be estimated (see 'Classification' section). An example of this is marker gene selection in microarray-based classification [33]. In fact, a counter-intuitive effect is that removing features may improve results, the so-called 'peaking phenomenon' [34] ('Fundamental Issues' section). Finally, the subset of features relevant to a problem can give insight into the biology underlying it. For example, subsequences important for protein secretion may give insight into protein sorting and trafficking [35]. However, dimensionality reduction is not aimed at improving acquiring novel biological insight *per se*. Supervised feature selection based on cross-validation ('Fundamental Issues' section), for instance, does not guarantee stable feature sets which are therefore not directly suitable for biological interpretation, a problem first encountered in investigating so-called disease 'gene signatures' found in microarray data [36–39].

Dimensionality reduction can be performed both supervised and unsupervised. For the former, the goal is to decrease the number of features while retaining as much information as possible about the classification or regression problem to be solved later. A major distinction here is between feature selection, selecting a subset of the original measurements, and feature extraction, combining measurements into new features, as illustrated in Figure 4. Feature selection [40] is a very computationally intensive problem, so a number of suboptimal methods have been proposed. The simplest method, filtering, orders all features using a criterion indicating how useful they are, for example, by a measure of class separation such as the t-statistic (Figure 4), and adds one feature at a time until performance stops improving. More complex forward and backward

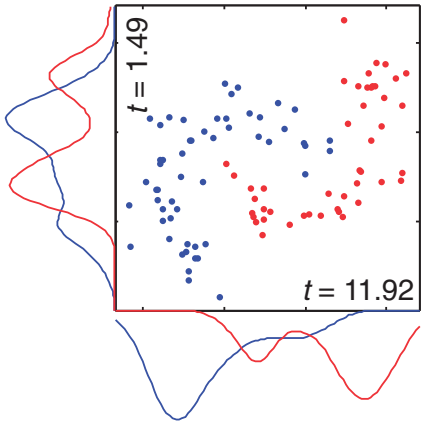

Figure 4: Feature selection. The feature on the *x*-axis separates the classes better than the feature on the *y*-axis, as indicated by the *t*-statistics.

search methods consider the possibility that two individually poor features may together be informative [41]. Feature extraction, in contrast, attempts to find a transformation $\mathbf{y} = f(\mathbf{x})$ for the original feature vector $\mathbf{x}$. A well-known supervised feature extraction method is 'linear discriminant analysis' [42], which takes on the simple form $\mathbf{y} = \mathbf{A}\mathbf{x}$, where matrix $\mathbf{A}$ is found to minimize within-class variation while maximizing between-class variation (Figure 1b). For both selection and extraction, care should be taken to avoid overfitting ('Fundamental Issues' section). In particular, in high-throughput measurements, it is often easy to find features predictive of the class label of individual training objects only as a result of noise; such features will provide little to no information for new objects.

Unsupervised dimensionality reduction uses criteria that do not depend on class labels or target outputs but rather on intrinsic properties of selected or extracted features. A well-known example is principal component analysis (PCA). It provides yet another linear transformation $\mathbf{y} = \mathbf{A}\mathbf{x}$, in which the orthogonal vectors in $\mathbf{A}$ are chosen to maximize the variation retained in the resulting $\mathbf{y}$ [43]. A similar but nonlinear transformation that attempts to preserve distances or dissimilarities between objects is multidimensional scaling (MDS) [44]. Both PCA and MDS are often used for visualization and 'visual clustering' of high-throughput measurement data: if groups of objects are distinct in a 2D/3D representation $\mathbf{y}$ of the original data, they are likely to correspond to interesting biological phenomena [32].

Classification

The most widely studied question in pattern recognition is how to design classifiers, algorithms assigning labels to previously unseen objects, given a training set of objects $\mathbf{x}$ with accompanying labels ω. The most obvious examples of classifiers in bioinformatics are those used for diagnosis and prognosis based on molecular measurements in medical applications, e.g. microarrays [14, 45]. Although less explicitly, classifiers are also used at the core of a variety of tools, some dating back to the 1970s [46]. For example, for gene annotation, a model of a gene sequence is trained, often an HMM, which is then applied to new genomes to detect initial open reading frames (ORFs) [47]. Similarly, proteins are assigned (putative) secondary structure, functions, locations and interactions using sequence-based classification [48].

There are two main distinct approaches to classification: either based on Bayesian decision theory or aiming to directly minimize some measure of the prediction error [49]. In the first, a classifier updates a prior probability of observing a certain label ω, $P(\omega)$, using the observation of a number of objects with that label. This results in a class-conditional probability distribution $P(\mathbf{x}|\omega)$ of observing certain values for features, $\mathbf{x}$, given the label ω. Using the prior probability $P(\omega)$ of observing a label ω, Bayes' rule is then applied to transform this into a posterior distribution: $P(\omega|\mathbf{x}) = P(\mathbf{x}|\omega)P(\omega)/P(\mathbf{x})$, i.e. the probability distribution of observing label ω, given feature values $\mathbf{x}$. Given a new object $\mathbf{x}$, the label ω that maximizes this posterior can then be assigned. As a result, the decision boundary (the line dividing regions of space in which objects are assigned to different classes) can be found where the posterior distributions for two classes are equal. This is illustrated in Figure 5. Although theoretically appealing, in practice, a choice will have to be made how to model $P(\mathbf{x}|\omega)$, leading to the name 'plug-in classifier'.

The problem of estimating class-conditional probability distributions from a limited set of objects is hard. In parametric approaches aimed at solving this problem, a certain functional form is assumed for each $P(\mathbf{x}|\omega)$, most often a Gaussian distribution [42]. As in many realistic bioinformatics problems, particularly those involving high-throughput measurements, the number of parameters p is far larger than the number of objects n (the '$p \gg n$' problem), estimating full distributions is often infeasible, and

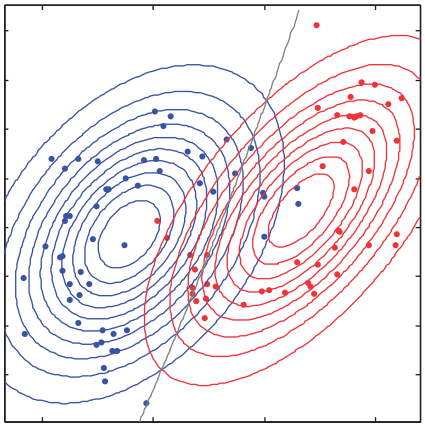

Figure 5: The plug-in classifier. In quadratic discriminant analysis, both classes are modeled by a Gaussian distribution.

simplifying assumptions are used [49], e.g. that covariances of all class-conditional distributions are identical (giving a linear decision boundary), that there is no covariance between features or even that all feature variances are equal (the nearest mean or nearest centroid classifier) [50]. Taking simplification even further, the 'nearest shrunken centroid classifier' [51] has feature selection built in. Note that it is also possible to use prior knowledge to construct more complex models for $P(\mathbf{x}|\omega)$, such as HMMs for gene sequences [47]. In nonparametric approaches, probability distributions are estimated without assuming a functional form, but rather by aggregating local density contributions of nearby objects. Well-known methods include histogramming, kernel density estimation [52] and the nearest neighbor method [49]. These bypass the parameter estimation problem, but are easily overfitted ('Fundamental Issues' section).

As an alternative, one can estimate the decision boundary directly, without first estimating densities. One of the most simple methods, Fisher's linear discriminant, assigns a target value of -1 to objects in one class and of $+1$ to objects in the other class, and attempts to find the weights $\mathbf{w}$ and b that minimize the difference between the output of a linear function $g(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x} + b$ and these targets [42, 49]. The decision boundary is then found where $g(\mathbf{x}) = 0$. A wide variety of classifiers are based on this principle. Logistic regression and perceptron classifiers assume a linear form for $g(\mathbf{x})$; feed-forward artificial neural networks have been widely used as universal function approximators for nonlinear $g(\mathbf{x})$ [53]. Recently, neural network research has been resurrected under the name deep learning and has already found application in bioinformatics [54]. Note that while these classifiers bypass the problem of estimating probability distributions, they are not immune to parameter estimation problems on relatively small datasets.

Two nonlinear classifiers are of particular interest in bioinformatics: support vector classifiers and decision trees. The support vector classifier is essentially a simple linear classifier that aims to minimize both the complexity of the classifier and the number of misclassifications on the training set, grounded in a solid theoretical framework called structural risk minimization [55]. The so-called kernel trick ('Data Representation' section) allows this framework to be used to learn nonlinear decision boundaries (as shown in Figure 1c) and to combine various sources of input data such as sequences, structures, etc. This flexibility makes the support vector classifier one of the most widely used algorithms in current bioinformatics [25].

Decision trees [56] take a different approach to constructing a classifier, by constructing a tree in which simple classifiers on single features (e.g. 'is the expression of gene $g > 2.5$?') are iteratively applied to select subsets of the data with the same label. Decision trees can model highly nonlinear decision boundaries. While decision trees are easily interpreted, they are prone to overfitting. To remedy this, random forests use many randomized trees and combine their output, e.g. by majority vote [57] (losing interpretability in the process). This is an example of combining classifiers: using different classifiers on the same data, or the same classifier on different aspects or subsets of the data, to achieve better performance or a more stable and trustworthy prediction [58].

PERFORMANCE EVALUATION

For supervised pattern recognition problems, it is essential to evaluate how well the trained system will perform on new objects. For classification, the criterion used most often is the prediction error, the expected percentage of misclassified objects. When costs of misclassification are available, for example, those of incorrectly diagnosing a patient with a certain disease and vice versa, the total expected cost could be calculated as well. Note that it is important to take prior probabilities into account in classifier design and evaluation whenever possible; in

particular, when they are highly skewed, for example, when detecting rare diseases.

In bioinformatics applications, prediction error is not always the most informative criterion, in particular for problems where the goal is to distinguish one target class [59]. Examples include the prediction of interactions (genetic, protein, protein–DNA, etc.), secreted proteins, relevant genetic variation, pathway membership, etc. A more natural choice in these cases is to calculate precision, the fraction of true positives (true interactions) in all predicted positives, and recall, the fraction of all positive objects predicted correctly by the classifier. Widely used is the receiver-operator characteristic (ROC) curve (Figure 6a), which shows the trade-off between the fraction of true positives and false positives as a function of a threshold on the output of the classifier, for example, the difference in the posterior probability. Each point on the ROC curve thus corresponds to a classifier. The ROC is often summarized by a single measure, the area under the curve (AUC). For a perfect classifier, the AUC reaches 1, and for a classifier that randomly assigns labels, it is 0.5. Note that as the axes reflect fractions rather than actual numbers, it may be wise to focus on only the (area under the) first part of the ROC [60].

Performance estimates based on the objects used to train a system will be too optimistic, i.e. biased: the system has seen these objects before, so they are no longer representative of new objects. Classifiers selected based on such estimates are likely to be heavily overfitted (adapted to the training data at hand) and will not generalize well on new data. Evaluation thus requires keeping apart objects that can serve as a test set, not used in any way to train the system, set parameters or make choices. In problems with limited datasets, this is wasteful. Therefore, often a form of cross-validation [61] is applied. In k-fold cross-validation, the dataset is randomly split into k roughly equal parts, and a classifier is trained on $k-1$ parts and tested on the remaining part k times. The cross-validation error estimate is then the average error over these k-folds. For $k=n$, this is called leave-one-out cross-validation. Note that care should be taken when applying cross-validation to small numbers of samples, as the variance of the resulting error estimate will make it unreliable. In this case, bootstrapping [62] is a better, although more biased, choice.

Performance evaluation is not only the last step in the pattern recognition pipeline, but also used for intermediate steps such as feature selection. To avoid overly optimistic error estimates [63], it is important to use a separate set of objects for such cases (called a validation set) or use cross-validation (or bootstrapping), leading to multiloop schemes with a feature selection cross-validation inside a training cross-validation [64].

FUNDAMENTAL ISSUES

A core problem in pattern recognition is generalization: how and under what circumstances can we use a limited set of data to predict well for unseen objects? A related question is model selection: what is the optimal model (e.g. classifier) complexity for a given problem? Given unlimited data, we could train highly complex models, fit their parameters well and achieve optimal performance. In practice, however, the number of training objects is severely limited by cost of measurement or lack of availability. This

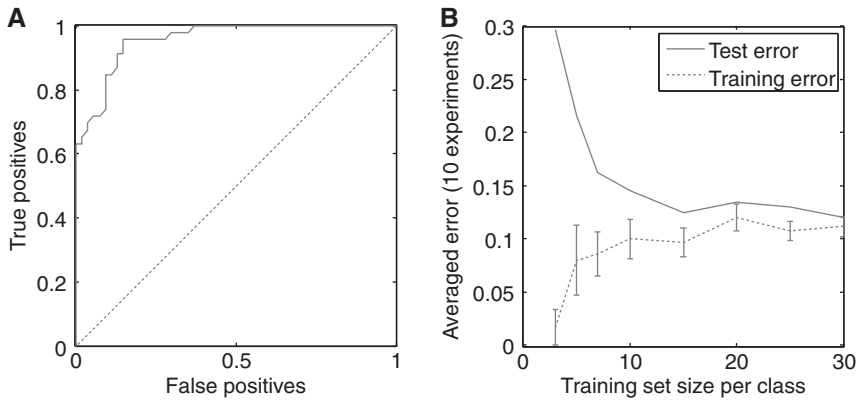

Figure 6: (**A**) An ROC curve for the classifier in Figure 5, where the left class is considered the positive class. The dashed line indicates the expected performance of a classifier that makes random decisions. (**B**) Learning curves for the same classifier.

so-called small sample size problem—often combined with large numbers of possibly informative features, the '$p \gg n$' problem—means we cannot train arbitrarily complex models. As a result, given a certain dataset size, there is a limit to the number of features we can exploit to train a classifier, the so-called peaking phenomenon [34] resulting from the curse of dimensionality [65].

To learn about generalization, learning curves (Figure 6b) help to estimate the minimum achievable error and can be extrapolated to see whether there is room for further improvement by increasing the number of objects available. For model selection, a number of theoretical approaches have been developed. At a high level, they all approach this problem as a trade-off between model complexity, i.e. the range of problems that a model can solve, and model descriptiveness, i.e. how well it solves the problem at hand. The optimal model is the one that is maximally descriptive with minimal complexity, i.e. is just complex enough to solve the given problem. The individual frameworks—Bayesian model selection [66], the minimum description length principle [67] and structural risk minimization [55]—start from different sets of assumptions, but lead to similar conclusions. While full Bayesian model selection is rarely used in practice, given its complexity, derived approaches such as regularization—i.e. penalizing high weights indicative of complex classifiers, or penalizing the number of features used—are routinely used to avoid overfitting [12, 68]. The structural risk minimization framework gives generalization bounds used in deriving the support vector classifier mentioned earlier, which attempts to minimize a combination of the weights of a classifier and the error it makes [24].

The most important message to convey in any course is that in practice, solving a pattern recognition problem remains a skill in which a number of different approaches are tried to learn what works well and what does not, always keeping in mind the recommendations given above to rigorously guard against overfitting and misinterpretation of results. Most classifiers have their niche, but proposals for novel classification schemes promising good performance should be taken with a grain of salt [69]. Often, the simplest approaches (such as hierarchical clustering, PCA and linear discriminants) work best, in particular with high-throughput measurement data. The best way for students to learn this is to experience it themselves in practice, for example, by ending a course with a small project in which they analyze their own dataset, or a dataset corresponding to a classic publication (e.g. [14, 45, 70]).

DEVELOPMENTS

New models and algorithms

Recent developments in pattern recognition and machine learning focus not as much on novel algorithms for existing problems as on finding solutions to (slightly) differently posed problems. Many of these algorithms are potentially useful in bioinformatics, but have not yet been extensively explored in literature. Examples are as follows:

- Bag-of-instances representations [71] of objects. In multiple instance learning, the problem is considered where such a bag is labeled positive if at least one instance is labeled positive [72]. This has already found application in drug discovery, relating possible structure of molecules to activity [73] and in predicting protein binding sites [74, 75].
- Semi-supervised learning [76], for cases where a large number of unlabeled objects are available besides a small set of labeled objects. This has already been exploited in prediction of gene function [77], in expression-based clustering [78, 79], prognosis [80, 81] and in the prediction of transcription factor binding sites [82]. A related development is that of positive unlabeled learning, assuming that some objects have a (single positive) label and the remainder is unlabeled [83], useful for protein–protein, genetic interaction data, etc. This has been used for predicting disease genes [84, 85] and delineating regulatory networks [86].
- Structured learning [87], predicting arbitrarily shaped output rather than a single label. Methods include HMMs [88] and, more recently, structured support vector machines and conditional random fields [89]. These have been applied for predicting gene structure (introns/exons) [46], protein secondary structure [90], drug activity [91], enzyme function [92] and interaction networks [93]. A particular case is multilabel learning, where several (hierarchically and ontologically) related labels are output, such as in predicting Gene Ontology annotations [94].
- In active learning [95], a classifier is used to decide which unlabeled object should be labeled next to

Table 1: An overview of general-purpose pattern recognition software

Name	Description	Environment	Free	GUI	URL
Weka	Full machine learning library with interactive GUI	Java	•	•	http://www.cs.waikato.ac.nz/ml/weka/
Orange	Interactive data analysis by scripting or visual programming	Python	•	•	http://orange.biolab.si/
Pattern Recognition Toolbox	Full range of algorithms for statistical pattern recognition	MATLAB™	•	○	http://www.newfolderconsulting.com/prt
STPRtool	Mostly classification routines, emphasis on linear and support vector classifiers	MATLAB™	•	○	http://cmp.felk.cvut.cz/cmp/software/stprtool/
MLR	Wrapper around R machine learning algorithms	R	•	-	http://mlr.r-forge.r-project.org/
Scikit-learn	Machine learning algorithms, emphasis on clustering and classification	Python	•	-	http://scikit-learn.org/
mlpy	Machine learning algorithms, including regression	Python	•	-	http://mlpy.sourceforge.net/
MILK	Limited set of classifiers	Python	•	-	http://packages.python.org/milk/
MDP	Pipeline-based collection of data analysis tools	Python	•	-	http://mdp-toolkit.sourceforge.net/
TOOLDIAG	Command line pattern recognition utilities	C	•	-	https://sites.google.com/site/tooldiag/
dlib	Library, focus on kernel methods	C++	•	-	http://dlib.net/
Waffles	Library, variety of learning algorithms	C++	•	-	http://waffles.sourceforge.net/
Shogun	Library focusing on kernel methods, interfaces to MATLAB™, Octave, R and Python	C++	•	-	http://www.shogun-toolbox.org/
Java-ML	Machine learning library, interfaces with Weka	Java	•	-	http://java-ml.sourceforge.net/
PRTools	Full range of algorithms for statistical pattern recognition, aimed at teaching and prototyping	MATLAB™	○	○	http://prtools.org/
PerClass	For pattern recognition systems design, from prototype to full implementation	MATLAB™	○	○	http://perclass.com/
SpotFire	Interactive data analysis, with some pattern recognition functionality, interface with R		-	•	http://spotfire.tibco.com/
SPSS	Statistical analysis package, including decision trees, neural nets		-	•	http://www.spss.com/

Note. The 'Free' column indicates whether the software can be used freely (•) or with some limitations (○: academic licenses, limited versions). The 'GUI' column indicates whether the software is based on a graphical user interface (•), is command-line driven with built-in visualization routines (○) or depends on third-party software for visualizing output (-). A more extensive overview can be found in [104].

best improve the classifier. Applications already explored include diagnosis [96], gene expression sampling [97], drug discovery [98] and predicting protein interactions [99] and transmembrane helices [100]. Active learning is often used implicitly, when classifier predictions are ranked and the most confident ones are verified experimentally first [101, 102]. Dedicated techniques could further enable current models to guide further experimentation [103].

Challenges

Next to the developments in pattern recognition itself, new challenges in biology also promise to place new demands on models and algorithms:

- It will be increasingly challenging to tie together various heterogeneous data sources in a single application. Pattern recognition algorithms will have to be more robust to missing data, better able to deal with various types of data and scalable to many more objects. Given limited storage and bandwidth, algorithms may have to be able to work on compressed or summarized data.
- As tools for measuring and particularly manipulating the cell become more widely available, pattern recognition should help close the systems biology loop by supporting researchers in setting up experiments. Given a limited experimental budget, which interventions together with which measurements are likely to increase our knowledge most? Active learning may prove very useful.
- It becomes increasingly important to help users to interpret why a pattern recognition algorithm predicts what it does. This calls for a move from black-box to grey-box models, which allows for a gain in biological knowledge. This requires a shift in emphasis from the performance of a trained predictor to its make-up, stability and uniqueness—are the parameters found meaningful or due to chance?

CONCLUSION

In this review, we presented the core elements of a pattern recognition course: data representation; the problems of clustering, dimensionality reduction and classification; performance evaluation; and model selection. We focused on the major issues and potential pitfalls in application and in interpretation of the results. Table 1 lists a number of software packages that can be used to solve pattern recognition problems; together with a textbook on pattern recognition [10–13] and data available online (for example, at http://www.ebi.ac.uk/arrayexpress/, http://www.ncbi.nlm.nih.gov/geo/, http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi), these could be the starting point for setting up a pattern recognition course.

Most of the examples discussed were rather simple applications of pattern recognition tools to solve straightforward prediction tasks in bioinformatics. It is good that students realize that pattern recognition is also widely used as part of larger bioinformatics tools. For example, software for detecting biological events in image and signal data uses classification algorithms; genome annotation pipelines depend, to a large extent, on detectors [105]; and visualization tools use clustering and dimensionality reduction [32]. Pattern recognition tools are also used as building blocks in the construction of systems biology models and networks, e.g. to predict and prioritize putative functional interactions between proteins [106], to learn regulatory modules [107], to link complex genotypes to phenotypes [106, 108], etc. Because it is unlikely that detailed mechanistic models of various cellular processes will become available soon, pattern recognition is likely to remain an essential tool in any bioinformatician's toolkit for some time.

Key Points

- Pattern recognition, concerned with algorithms that learn to solve a problem using a limited set of measurement data, is an essential part of bioinformatics education.
- The representation of the original measurements—as features, dissimilarities or kernels—is a decisive factor in obtaining good performance.
- Unsupervised analyses such as clustering can help to interpret the data and may give new insights, but never yield objective 'proof' of a finding.
- In performance evaluation, use a separate test set or cross-validation to guard against overfitting, a significant risk especially on high-throughput data.
- The core problem in pattern recognition is that of model selection: choosing a model with minimum complexity and maximal descriptiveness.

Acknowledgements

The authors are members of the Netherlands Bioinformatics Centre and the Kluyver Centre for Genomics of Industrial Fermentation, subsidiaries of the Netherlands Genomics Initiative.

References

1. Quackenbush J. Computational analysis of microarray data. *Nat Rev Genet* 2001;**2**:418–27.
2. Gstaiger M, Aebersold R. Applying mass spectrometry-based proteomics to genetics, genomics and network. *Nat Rev Genet* 2009;**10**:617–27.
3. Schuster SC. Next-generation sequencing transforms today's biology. *Nat Methods* 2008;**5**:16–18.
4. Luscombe ND, Greenbaum D, Gerstein M. What is bioinformatics? A proposed definition and overview of the field. *Methods Inf Med* 2001;**40**:346–58.
5. Butte A. The use and analysis of microarray data. *Nat Rev Drug Discov* 2002;**1**:951–60.
6. Bruggeman FJ, Westerhoff HV. The nature of systems biology. *Trends Microbiol* 2007;**15**:45–50.
7. Russell S, Norvig P. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall, 2009.
8. Witten IH, Frank E, Hall MA. *Data Mining: Practical Machine Learning Tools and Techniques*. Burlington, MA: Morgan Kaufmann, 2011.
9. Mohri M, Rostamizadeh A, Talwalkar A. *Foundations of Machine Learning*. Cambridge, MA: MIT Press, 2012.
10. Duda RO, Hart PE, Stork DG. *Pattern Classification*. Hoboken, NJ: Wiley-Interscience, 2000.
11. Webb AR, Copsey KD. *Statistical Pattern Recognition*. Hoboken, NJ: Wiley, 2011.
12. Hastie T, Tibshirani R, Friedman J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Heidelberg, Germany: Springer, 2009.
13. Bishop CM. *Pattern Recognition and Machine Learning*. Heidelberg, Germany: Springer, 2007.
14. van 't Veer LJ, Dai H, van de Vijver MJ, *et al*. Gene expression profiling predicts clinical outcome of breast cancer. *Nature* 2002;**415**:530–6.
15. Gokmen-Polar Y, Badve S. Molecular profiling assays in breast cancer: are we ready for prime time? *Oncology (Williston Park)* 2012;**26**:350–7, 361.
16. Dunham I, Kundaje A, Aldred SF, *et al*. An integrated encyclopedia of DNA elements in the human genome. *Nature* 2012;**489**:57–74.
17. Encode Consortium. Machine learning approaches to genomics. 2012. http://www.nature.com/encode/threads/machine-learning-approaches-to-genomics.
18. Jain AK, Duin RPW, Mao J. Statistical pattern recognition: a review. *IEEE Trans Pattern Anal Mach Intell* 2000;**22**:4–37.
19. Hodge V, Austin J. A survey of outlier detection methodologies. *Artif Intell Rev* 2004;**22**:85–126.
20. Donders ART, van der Heijden GJMG, Stijnen T, *et al*. Review: a gentle introduction to imputation of missing values. *J Clin Epidemiol* 2006;**59**:1087–91.
21. Duin R, Pekalska E, de Ridder D. Relational discriminant analysis. *Pattern Recognit Lett* 1999;**20**:1175–81.
22. Pekalska E, Duin RPW. *The Dissimilarity Representation for Pattern Recognition: Foundations and Applications*. Singapore: World Scientific Publishing, 2005.
23. Shawe-Taylor J, Cristianini N. *Kernel Methods for Pattern Analysis*. Cambridge, UK: Cambridge University Press, 2004.
24. Cortes C, Vapnik VN. Support vector networks. *Mach Learn* 1995;**20**:273–97.
25. Ben-Hur A, Ong CS, Sonnenburg S, *et al*. Support vector machines and kernels for computational biology. *PLoS Comput Biol* 2008;**4**:e1000173.
26. Gönen M, Alpaydın E. Multiple kernel learning algorithms. *J Mach Learn Res* 2011;**12**:2211–68.
27. MacCuish JD, MacCuish NE. *Clustering in Bioinformatics and Drug Discovery*. Boca Raton, FL: CRC Press, 2010.
28. Jain AK, Murthy MN, Flynn PJ. Data clustering: a review. *ACM Comput Surv* 1999;**31**:264–323.
29. Eisen MB, Spellman PT, Brown PO, *et al*. Cluster analysis and display of genome-wide expression patterns. *Proc Natl Acad Sci USA* 1998;**95**:14863–8.
30. Tibshirani R, Walther G, Hastie T. Estimating the number of clusters in a data set via the gap statistic. *J R Stat Soc Series B* 2001;**63**:411–23.
31. Handl J, Knowles J, Kell DB. Computational cluster validation in post-genomic data analysis. *Bioinformatics* 2005;**21**:3201–12.
32. Gehlenborg N, O'Donoghue SI, Baliga NS, *et al*. Visualization of omics data for systems biology. *Nat Methods* 2010;**7**:S56–68.
33. Davis CA, Gerick F, Hintermair V, *et al*. Reliable gene signatures for microarray classification: assessment of stability and performance. *Bioinformatics* 2006;**22**:2356–63.
34. Hughes GF. On the mean accuracy of statistical pattern recognizers. *IEEE Trans Inf Theory* 1968;**14**:55–63.
35. van den Berg BA, Reinders MJ, Hulsman M, *et al*. Exploring sequence characteristics related to high-level production of secreted proteins in *Aspergillus niger*. *PLoS One* 2012;**7**:e45869.
36. Michiels S, Koscielny S, Hill C. Prediction of cancer outcome with microarrays: a multiple random validation strategy. *Lancet* 2005;**365**:488–92.
37. Venet D, Dumont JE, Detours V. Most random gene expression signatures are significantly associated with breast cancer outcome. *PLoS Comput Biol* 2011;**7**:e1002240.
38. Ein-Dor L, Kela I, Getz G, *et al*. Outcome signature genes in breast cancer: is there a unique set? *Bioinformatics* 2005;**21**:171–8.
39. Ein-Dor L, Zuk O, Domany E. Thousands of samples are needed to generate a robust gene list for predicting outcome in cancer. *Proc Natl Acad Sci USA* 2006;**103**:5923–8.
40. Saeys Y, Inza I, Larranaga P. A review of feature selection techniques in bioinformatics. *Bioinformatics* 2007;**23**:2507–17.
41. Guyon I, Elisseeff A. An introduction to variable and feature selection. *J Mach Learn Res* 2003;**3**:1157–82.
42. Zhang MQ. Discriminant analysis and its application in DNA sequence motif recognition. *Brief Bioinformatics* 2000;**1**:331–42.
43. Ma S, Dai Y. Principal component analysis based methods in bioinformatics studies. *Brief Bioinformatics* 2011;**12**:714–22.
44. Tzeng J, Lu HH, Li WH. Multidimensional scaling for large genomic data sets. *BMC Bioinformatics* 2008;**9**:179.
45. Golub TR, Slonim DK, Tamayo P, *et al*. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* 1999;**286**:531–7.

46. Baldi P, Brunak S. *Bioinformatics: The Machine Learning Approach*. Cambridge, MA: MIT Press, 2001.
47. Korf I. Gene finding in novel genomes. *BMC Bioinformatics* 2004;**5**:59.
48. Juncker AS, Jensen LJ, Pierleoni A, *et al*. Sequence-based feature prediction and annotation of proteins. *Genome Biol* 2009;**10**:206.
49. Dudoit S, Fridlyand J, Speed TP. Comparison of discrimination methods for the classification of tumors using gene expression data. *J Am Stat Assoc* 2002;**97**:77–87.
50. Dabney AR. Classification of microarrays to nearest centroids. *Bioinformatics* 2005;**21**:4148–54.
51. Tibshirani R, Hastie T, Narasimhan B, *et al*. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proc Natl Acad Sci USA* 2002;**99**:6567–72.
52. Murakami Y, Mizuguchi K. Applying the Naive Bayes classifier with kernel density estimation to the prediction of protein-protein interaction sites. *Bioinformatics* 2010;**26**:1841–8.
53. Lancashire LJ, Lemetre C, Ball GR. An introduction to artificial neural networks in bioinformatics - application to complex microarray and mass spectrometry datasets in cancer studies. *Brief Bioinformatics* 2009;**10**:315–29.
54. Huma L. Computational biology perspective: kernel methods and deep learning. *Comput Stat* 2012;**4**:455–65.
55. Vapnik VN. *Statistical Learning Theory*. Hoboken, NY: Wiley, 1998.
56. Chen X, Wang M, Zhang H. The use of classification trees for bioinformatics. *Data Min Knowl Discov* 2011;**1**: 55–63.
57. Diaz-Uriarte R, Alvarez de Andres S. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics* 2006;**7**:3.
58. Kittler J, Hatef M, Duin RPW, *et al*. On combining classifiers. *IEEE Trans Pattern Anal Mach Intell* 1998;**20**:226–39.
59. Baldi P, Brunak S, Chauvin Y, *et al*. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics* 2000;**16**:412–24.
60. Hulsman M, Reinders M, de Ridder D. Evolutionary optimization of kernel weights improves protein complex comembership prediction. *IEEE-ACM Trans Comput Biol Bioinformatics* 2009;**6**:427–37.
61. Simon RM, Subramanian J, Li MC, *et al*. Using cross-validation to evaluate predictive accuracy of survival risk. *Brief Bioinformatics* 2011;**12**:203–14.
62. Braga-Neto UM, Dougherty ER. Is cross-validation valid for small-sample microarray classification? *Bioinformatics* 2004;**20**:374–80.
63. Ambroise C, McLachlan GJ. Selection bias in gene extraction on the basis of microarray gene-expression. *Proc Natl Acad Sci USA* 2002;**99**:6562–6.
64. Wessels LF, Reinders MJ, Hart AA, *et al*. A protocol for building and evaluating predictors of disease state based on microarray data. *Bioinformatics* 2005;**21**:3755–62.
65. Bellman RE. *Dynamic Programming*. Princeton, NY: Princeton University Press, 1957.
66. Wasserman L. Bayesian model selection and model averaging. *J Math Psychol* 2000;**44**:92–107.
67. Zhao W, Serpedin E, Dougherty ER. Inferring gene regulatory networks from time series data using the minimum. *Bioinformatics* 2006;**22**:2129–35.
68. Waldron L, Pintilie M, Tsao MS, *et al*. Optimized application of penalized regression methods to diverse genomic data. *Bioinformatics* 2011;**27**:3399–406.
69. Jelizarow M, Guillemot V, Tenenhaus A, *et al*. Over-optimism in bioinformatics: an illustration. *Bioinformatics* 2010; **26**:1990–8.
70. Khan J, Wei JS, Ringner M, *et al*. Classification and diagnostic prediction of cancers using gene expression. *Nat Med* 2001;**7**:673–9.
71. Ray S, Craven M. Supervised versus multiple instance learning: an empirical comparison. In: *International Conference on Machine Learning (ICML), 2005*, pp. 697–704. ACM, New York.
72. Dietterich TG, Lathrop RH, Lozano-Pérez T. Solving the multiple instance problem with axis-parallel rectangles. *Artif Intell* 1997;**89**:31–71.
73. Fu G, Nan X, Liu H, *et al*. Implementation of multiple-instance learning in drug activity prediction. *BMC Bioinformatics* 2012;**13(Suppl. 15)**:S3.
74. Pfeifer N, Kohlbacher O. Multiple instance learning allows MHC class II epitope predictions across alleles. In: Crandall K, Lagergren J (eds). *International Workshop on Algorithms in Bioinformatics (WABI)*. Heidelberg, Germany: Springer, 2008, 210–21.
75. Minhas F, Ben-Hur A. Multiple instance learning of Calmodulin binding sites. *Bioinformatics* 2012;**28**:i416–22.
76. Chapelle O, Schölkopf B, Zien A. *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2010.
77. Qi Y, Missiuro PE, Kapoor A, *et al*. Semi-supervised analysis of gene expression profiles for lineage-specific development in the *Caenorhabditis elegans* embryo. *Bioinformatics* 2006;**22**:e417–23.
78. Costa IG, Krause R, Opitz L, *et al*. Semi-supervised learning for the identification of syn-expressed genes from fused microarray and *in situ* image data. *BMC Bioinformatics* 2007;**8(Suppl. 10)**:S3.
79. Steinfeld I, Navon R, Ardigò D, *et al*. Clinically driven semi-supervised class discovery in gene expression data. *Bioinformatics* 2008;**24**:i90–7.
80. Shi M, Zhang B. Semi-supervised learning improves gene expression-based prediction of cancer recurrence. *Bioinformatics* 2011;**27**:3017–23.
81. Bair E, Tibshirani R. Semi-supervised methods to predict patient survival from gene expression data. *PLoS Biol* 2004; **2**:E108.
82. Ernst J, Beg QK, Kay KA, *et al*. A semi-supervised method for predicting transcription factor–gene interactions in *Escherichia coli*. *PLoS Comput Biol* 2008;**4**:e1000044.
83. Calvo B. *Positive Unlabelled Learning with Applications in Computational Biology*. Department of Computer Science and Artificial Intelligence. San Sebastian, Spain: University of the Basque Country, 2008.
84. Yang P, Li X, Mei JP, *et al*. Positive-unlabeled learning for disease gene identification. *Bioinformatics* 2012;**28**: 2640–47.
85. Mordelet F, Vert JP. ProDiGe: prioritization of disease genes with multitask machine learning from positive and unlabeled examples. *BMC Bioinformatics* 2011;**12**:389.
86. Cerulo L, Elkan C, Ceccarelli M. Learning gene regulatory networks from only positive and unlabeled data. *BMC Bioinformatics* 2010;**11**:228.
87. Baklr G, Hofmann T, Schölkopf B, *et al*. *Predicting Structured Data*. Cambridge, MA: MIT Press, 2007.

88. Eddy SR. Hidden Markov models. *Curr Opin Struct Biol* 1996;**6**:361–5.

89. Sutton C, McCallum A. An introduction to conditional random fields for relational learning. In: Getoor L., Taskar B (eds). *Introduction to Statistical Relational Learning*. Cambridge, MA: MIT Press, 2007.

90. Liu Y, Carbonell J, Weigele P, *et al*. Protein fold recognition using segmentation conditional random fields (SCRFs). *J Comput Biol* 2006;**13**:394–406.

91. Su H, Heinonen M, Rousu J. Structured output prediction of anti-cancer drug activity. In: Dijkstra TH, Tsivtsivadze E, Marchiori E, *et al* (eds). *Pattern Recognition in Bioinformatics*. Heidelberg, Germany: Springer, 2010;38–49.

92. Astikainen K, Holm L, Pitkänen E, *et al*. Structured output prediction of novel enzyme function with reaction kernels. In: Fred A, Filipe J, Gamboa H (eds). *Biomedical Engineering Systems and Technologies*. Heidelberg, Germany: Springer, 2011;367–79.

93. Geurts P, Touleimat N, Dutreix M, *et al*. Inferring biological networks with output kernel trees. *BMC Bioinformatics* 2007;**8(Suppl. 2)**:S4.

94. Rousu J, Saunders C, Szedmak S, *et al*. Kernel-based learning of hierarchical multilabel classification models. *J Mach Learn Res* 2006;**7**:1601–26.

95. Tong S, Koller D. Support vector machine active learning with applications to text classification. *J Mach Learn Res* 2001;**2**:45–66.

96. Liu Y. Active learning with support vector machine applied to gene expression data for cancer classification. *J Chem Inf Comput Sci* 2004;**44**:1936–41.

97. Singh R, Palmer N, Gifford DK, *et al*. Active learning for sampling in time-series experiments with application to gene expression analysis. In: *International Conference on Machine Learning (ICML)*, Bonn, Germany, pp. 832–9, 2005.

98. Warmuth MK, Liao J, Rätsch GR, *et al*. Active learning with support vector machines in the drug discovery process. *J Chem Inf Comput Sci* 2003;**43**:667–73.

99. Mohamed TP, Carbonell JG, Ganapathiraju MK. Active learning for human protein-protein interaction prediction. *BMC Bioinformatics* 2010;**11(Suppl. 1)**:S57.

100. Osmanbeyoglu HU, Wehner JA, Carbonell JG, *et al*. Active machine learning for transmembrane helix prediction. *BMC Bioinformatics* 2010;**11(Suppl. 1)**:S58.

101. Wong SL, Zhang LV, Tong AH, *et al*. Combining biological networks to predict genetic interactions. *Proc Natl Acad Sci USA* 2004;**101**:15682–7.

102. Moreau Y, Tranchevent LC. Computational tools for prioritizing candidate genes: boosting disease gene. *Nat Rev Genet* 2012;**13**:523–36.

103. King RD, Whelan KE, Jones FM, *et al*. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature* 2004;**427**:247–52.

104. Mikut R, Reischl M. Data mining tools. *Data Min Knowl Discov* 2011;**1**:431–43.

105. Holt C, Yandell M. MAKER2: an annotation pipeline and genome-database management tool for second-generation genome projects. *BMC Bioinformatics* 2011;**12**:491.

106. Szklarczyk D, Franceschini A, Kuhn M, *et al*. The STRING database in 2011: functional interaction networks of proteins. *Nucleic Acids Res* 2011;**39**:D561–8.

107. Segal E, Shapira M, Regev A, *et al*. Module networks: identifying regulatory modules and their condition-specific. *Nat Genet* 2003;**34**:166–76.

108. Jelier R, Semple JI, Garcia-Verdugo R, *et al*. Predicting phenotypic variation in yeast from individual genome sequences. *Nat Genet* 2011;**43**:1270–4.