Gene expression-based classifiers

Vlad Popovici[1,2]

[1] *Bioinformatics Core Facility, Swiss Institute of Bioinformatics, CH-1015 Lausanne, Switzerland; e-mail: vlad.popovici@isb-sib.ch*
[2]*Institute of Biostatistics and Analyses, Masaryk University, CZ-62500 Brno, Czech Republic*

Abstract

Whole genome profiling and decreasing costs of genome sequencing enable measuring the activity of tens of thousands of genes which can potentially be used for making predictions about patients' risk of relapse or response to a specific treatment. These predictions are based on mathematical models that combine the measurements from a selected set of genes into either a continuous score or a binary outcome. In order to build such models that can be used in clinical practice with real benefits for the patients, a rigorous methodological approach must be followed and the purpose of this chapter is to briefly describe some theoretical considerations and practical results in the field of gene expression-based classifiers.

Key words

Classifiers, biomarkers, performance estimation, model validation.

1. Introduction

The clinical practice has shown that many cancer treatments benefit only a small group of patients who received them. Lacking precise means of identifying the patients most likely to respond to a given treatment results in many patients being prescribed ineffective treatments, which puts a serious burden on them and on the health care systems. The *personalized medicine* addresses exactly this problem by trying to diagnose and treat a disease using information about patient's genes, proteins and environment. At the core of the diagnostic and treatment decisions are placed the *classifiers*, which are mathematical models assembling all the information into a system producing binary or multi-valued decisions. In this context, the new treatments are accompanied by diagnostic tests which are supposed to identify the most likely responders.

The problem of companion diagnostic tests is even more important in the case of *targeted therapies*, which are "drugs or other substances that block the growth and spread of cancer by interfering with specific molecules involved in tumor growth and progression" (NCI's Facts Sheet, http://www.cancer.gov/). As these drugs target specific molecular processes, such as cell growth signaling, angiogenesis, apoptosis, or stimulate the immune response, highly specific tests are needed to identify the right patient population.

1.1. What is a classifier in the context of genomic data

There are various names under which the classifiers appear in the literature related to gene expression-based diagnostics and prognostics. They may be called "(multigene) expression signature" or "(multigene) biomarkers" or simply "risk predictors/scores". In general, we talk about a classifier when we have in mind a model which produces a crisp decision (be it binary or multi-level). While a score can be converted into a decision (see further on in this chapter), and so “score” and “classifier” terms could be used interchangeably in some context, a gene expression signature is usually not enough to specify a classifier. A gene expression signature refers more to the genes selected to be specific to some phenotype, but it normally does not specify the way these genes should be combined to predict the phenotype in question. Also the term “biomarker” could be misleading, since it may also refer to some markers that can be mechanistically linked to a disease activity. In conclusion, we prefer the term “gene-based classifier” by which we mean a prediction model which combines the gene expressions (and maybe other variables) in a model. This classifier may have a score as an intermediate step towards decision.

2. Classifiers

Without any loss of generality, we will consider in the following the case of *binary classifiers*, constructed on continuous variables and we will denote the two alternatives (called *classes*) by “-1” and “+1”. Let $f\colon \mathbb{R}^p \to \mathbb{R}$ be a real-valued function which will map a vector $\mathrm{x} \in \mathbb{R}^p$ to a continuous *score*. A score $s = f(\mathrm{x})$ is converted into a *binary class label* by $h(s) = \mathrm{sign}(s)$. Some classifiers will directly produce the binary label (e.g. the basic Top Scoring Pairs algorithm, described later in this chapter), while others will firstly produce a score. As the labels are easily obtained from the scores and since using continuous functions is more convenient for modeling, we will generally focus on finding the function f rather than h. We can then state the problem of learning a classification rule to be the task of finding a real-valued function $f \in \mathcal{F}$ that maps each point of the input space (here considered to be $\mathbb{R}^p$) to a score that, after thresholding, will produce a binary label which will not differ in too many cases from the true label. This formulation is too vague to be of any practical utility as long as we do not specify

- which is the function space $\mathcal{F}$ in which we search for the solution;
- what we mean by ‘differ’, and
- how many misclassified cases is ‘too many’ for a classifier to be considered good.

The choice of the function space $\mathcal{F}$ is the first decision a data modeler has to take and, in most cases, it means defining a parametric form for the score function f. Let the parameters on which f depends be denoted by a r-dimensional parameter vector $\omega \in \Omega \subseteq \mathbb{R}^r$. The problem of *training* a classifier becomes an optimization problem in which one has to find the optimal vector ω^* such that the *expected risk of misclassification (expected prediction error)* is minimized:

$$\omega^* = \arg\max_{\omega} \int L\big(y, f(\mathrm{x})\big)\, dP(\mathrm{x}, y) \ ,$$

where L is a *loss function* penalizing the discrepancies between the predicted label $f(\mathrm{x})$ and the true label y. The integral is taken with respect to the probability density function P which is generating the population $\{(\mathrm{x}, y) | \mathrm{x} \in \mathbb{R}^p,\ y = \pm 1\}$. As the probability function is usually not available, the risk is estimated from a finite *training set (sample)* given as a pair of sets $X_n^t = \{\mathrm{x}_i, i = 1, \dots, n\} \subset \mathbb{R}^p$ and $Y_n^t = \{y_i = \pm 1,\ i = 1, \dots, n\}$ of points draw independently and identically distributed from the underlying probability. In this case, the prediction error can only be estimated from the finite sample, thus the estimation will become dependent on the particular training set. This observation justifies the introduction of various error estimation techniques, some briefly described in this chapter.

The loss function is of central importance in defining the form of the classifier and several ways of penalizing the errors have been proposed in the literature (Hastie et al, 2009; Duda et al, 2001). Here we will consider only the case of *squared error loss*,

$$L\big(y, f(\mathrm{x})\big) = \frac{1}{2}\big(y - f(\mathrm{x})\big)^2$$

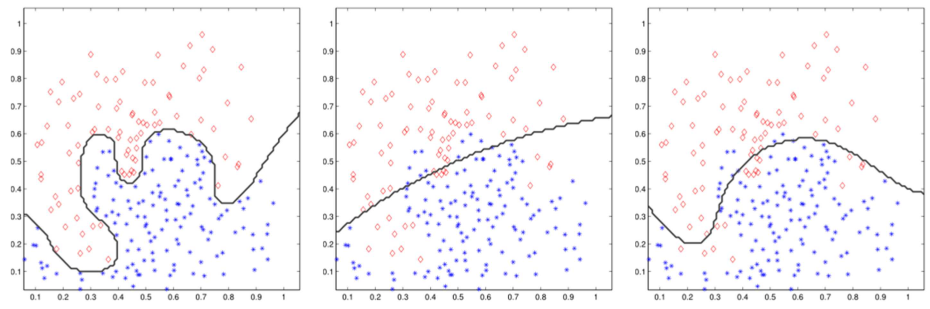
which is, by far, the most commonly used. In the case of a risk of misclassification estimated from a finite sample, we talk about *empirical risk (of misclassification)* and we estimate it by its mean value over the given sample:

$$\frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\mathrm{x}_i) \quad ,$$

which for squared error loss is simply the usual mean squared error, $\frac{1}{n}\sum_i\big(y_i - f(\mathrm{x}_i)\big)^2$.

Figure 1 depicts a possible scenario for a binary classification problem in the case of $p = 2$, with the solid line representing the *classification boundary* (i.e. the separation between the two classes), defined by the equation $f(\mathrm{x}) = 0$. Let us analyze the three proposed solutions: in the first panel, the classifier perfectly separates the two classes, while the other two solutions are simpler (smoother) classification functions, which misclassify some points. In practice, it turns out that a function that perfectly separates the training set will usually perform poorly on unseen data, i.e. the prediction will have high *variance* on different samples drawn from the same underlying distribution P as the training data. We say in this case that the first function *overfits* the training set. On the other hand, a too simplistic explanation as the one given by the second classifier will never be able to satisfactory fit the training data, i.e. the model chosen has a large *bias*. In this case we say that the model *underfits* the training set. The central problem of machine learning is to find that right tradeoff between underfitting and overfitting, that will generate functions f able to generalize well, i.e. their performance remains good on unseen data. We will further detail what we mean by good performance of a classifier. This problem is also known as bias-variance dilemma in classical statistics.

Figure 1. Three possible scenarios for a trained classifier: different degrees of *regularization* lead to different solutions, with various performances.



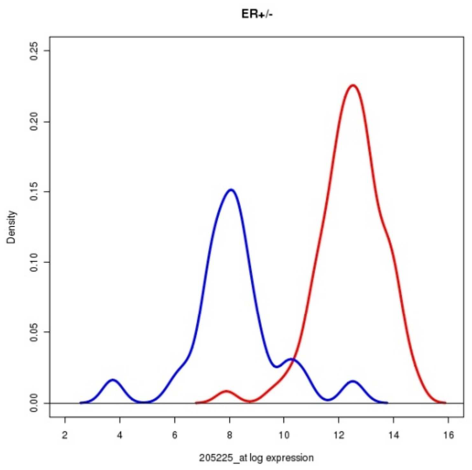
To conclude this introductory section, we note that the genomic applications of classifiers face a specific problem of fitting models in very high dimensional spaces (in general, $p \gg n$). Because of the high number of degrees of freedom of the learning problem, one can always find a classifier that perfectly fits the training set, for any possible labeling. Or, to put it in other terms, higher is the dimensionality of the space slower is the convergence of the estimators (of the parameters) – phenomenon called *curse of dimensionality*. It follows that one has either to use a very large learning set or to constrain the form of the classifier such that it fits only the most salient characteristics of the two classes. The first solution is not practically possible, so only the second approach remains feasible. Luckily, the variables (genes) are not all independent and a large proportion of them are usually not important for the classification task. This explains why, despite of the unfavorable settings, for many applications one can find proper classifiers with reasonable performance.

2.1. Bayesian decision theory

Let us consider for a moment the best case scenario in which the classification problem is completely specified by the probability functions:

- $P(y = +1)$ and $P(y = -1)$ are called *prior probabilities (priors)* and give the probability of either of the classes, when no other information is available. In general, for a binary classification problem, one excludes the possibility of observing any other class but one of the two ($y \in \{\pm 1\}$), so $P(y = 1) = 1 - P(y = -1)$.
- *class-conditional density functions*, $p(\mathrm{x}|y = 1)$ and $p(\mathrm{x}|y = -1)$, for $\mathrm{x} \in \mathbb{R}^p$, the probability density function of x, given that its label is “+1” (or “-1”).

Figure 2. Class conditional density functions for ESR1 gene expression as measured by one probeset: $p(205225\_\mathrm{at}\ | y = "ER+")$ and $p(205225\_\mathrm{at}\ | y = "ER-")$. The two classes are estrogen-positive (ER+, red line) and estrogen-negative (ER-, blue line).



Using Bayes’ rule, it is easy to obtain the *posterior* probability

$$P(y=\pm 1|\mathrm{x})=\frac{p(\mathrm{x}|y=\pm 1)\,P(y=\pm 1)}{p(\mathrm{x})}.$$

This shows that by observing the vector x (called evidence) and using information about priors and class conditional densities – called *likelihood* – one can obtain the posterior probability that the observed instance belongs to one of the classes. It follows naturally that, for minimizing the risk of misclassification one must assign x to the class with maximum posteriori probability (*Bayes decision rule*):

$$f(\mathrm{x})=\begin{cases}-1, & P(y=-1|\mathrm{x})>P(y=+1|\mathrm{x})\\ +1, & P(y=-1|\mathrm{x})\le P(y=+1|\mathrm{x})\end{cases}$$

It is sometimes convenient to consider this rule in terms of *log-ratio*: assign x to class "+1" if

$$\log\frac{P(y=+1|\mathrm{x})}{P(y=-1|\mathrm{x})}\ge 0,$$

and to class "-1" otherwise.

The Bayesian decision is optimal in the sense that it minimizes the probability of error, but it requires full information about priors and class-conditional densities to be available. However, this is not the case in real applications, and a plethora of approaches have been proposed to deal with more realistic scenarios. One can try, for example, to consider a parametric model for the probabilities (e.g. linear discriminant analysis, naïve Bayes classifier, etc. etc.) or to use nonparametric estimators of the densities.

2.2. Linear discriminants

Suppose that the class-conditional densities are multivariate Gaussians,

$$p(\mathrm{x}|y=\pm 1)=\frac{1}{(2\pi)^{p/2}\left|\Sigma_{\pm 1}\right|^{1/2}}e^{-\frac{1}{2}(\mathrm{x}-\mu_{\pm 1})^T\Sigma_{\pm 1}^{-1}(\mathrm{x}-\mu_{\pm 1})}$$

where $\mu_{\pm 1}$ are the mean vectors and $\Sigma_{\pm 1}$ are the covariance matrices of the two respective classes ($|\cdot|$ is the determinant operator). If the two classes have equal covariance matrices, $\Sigma_{-1}=\Sigma_{+1}=\Sigma$, the log-ratio of the posteriors becomes

$$\log\frac{P(y=+1|\mathrm{x})}{P(y=-1|\mathrm{x})}=\log\frac{P(y=+1)}{P(y=-1)}-\frac{1}{2}(\mu_{-1}+\mu_{+1})^T\Sigma^{-1}(\mu_{+1}-\mu_{-1})+\mathrm{x}^T\Sigma^{-1}(\mu_{+1}-\mu_{-1}).$$

The values for the class means and the covariance matrix have to be estimated from the training set, using the usual estimators. The priors are estimated by the class frequencies, $\hat{P}(y=+1)=n_{+1}/n$, and $\hat{P}(y=-1)=n_{-1}/n$, where $n_{+/-1}$ are the number of elements in each class. The above equation shows that the decision boundary between classes is a linear function of x (for equal covariance matrices). By introducing the discriminant functions

$$\delta_{\pm 1}(\mathrm{x})=\mathrm{x}^T\Sigma^{-1}\mu_{\pm 1}-\frac{1}{2}\mu_{\pm 1}^T\Sigma\mu_{\pm 1}+\log P(y=\pm 1)$$

the decision rule $y=\arg\max_{k=\pm 1}\delta_k(\mathrm{x})$ is equivalent to comparing the log-ratios of the posteriors with 0. As a final remark, we note that $d(\mathrm{x},\mu)=(\mathrm{x}-\mu)^T\Sigma(\mathrm{x}-\mu)$ is called *Mahalanobis distance* from x to μ (which becomes Euclidean distance if the covariance matrix is the unit matrix) and that, under equal covariances assumption, the decision rule assigns x to the class whose centroid (μ) is the closest in the sense of this metric.

The squared error loss function, mentioned in the introduction of this chapter, is intimately linked to LDA classifier as this can be derived from a linear regression model, where we fit a linear model to the label variable, considered this time a continuous variable.

2.3. Nearest neighbor and related classifiers

The intuition behind the nearest neighbor and related methods is that an observation should be assigned to the class containing other similar observations. The nearest neighbor classifiers employ a *voting scheme* for deciding the class membership of a sample x $\in \mathbb{R}^p$. The predicted (estimated) label is

$$\hat{y} = \operatorname{sign}\left(\sum_{x_i \in N_k(x)} y_i \right)$$

where N_k(x) is a neighborhood of k closest points to x. In other words, the predicted label $\hat{y}$ is the most common label among the k points in the neighborhood. If $\hat{y} = 0$ it means that the point lies on the decision boundary and it has equal number of points from each class in its neighborhood.

The notion of neighborhood implies the existence of a metric which, at its turn, is closely related to the notion of *similarity*, in the sense that more similar observations are closer to each other than observations less similar. In the case of $\mathbb{R}^p$, the natural metric is the Euclidean distance, but this is not necessarily the best choice. For genomic applications, in which the observations may be corrupted by high levels of noise, one may consider alternative distances, for example

- *correlation distance*: $d_{corr}(\mathrm{x},\mathrm{z}) = 1 - \rho(\mathrm{x},\mathrm{z})$
- *cosine distance*: $d_{cos}(\mathrm{x},\mathrm{z}) = 1 - \frac{\langle \mathrm{x},\mathrm{z} \rangle}{\|\mathrm{x}\|\|\mathrm{z}\|}$, where $\langle \cdot,\cdot \rangle$ denotes the scalar product of two vectors, and $\|\cdot\|$ the L_2 norm, respectively.

The parameter $k \geq 1$ has to be optimized for each problem, usually by cross-validation (see later on in this chapter). Smaller values will lead to a better fit of the training set, but may have an adverse effect on the generalization properties of the classifier. Also, there is a direct link between the parameter k and the smoothness of the decision boundary.

Instead of considering all the points in the data set in the decision rule, one may choose to select only a few "representative" patterns from each class and to compute the distances only to these points in order to classify a new observation. This is similar to LDA decision where, as we have seen above, one computes the Mahalanobis distance to the centers of the two classes and uses this information to classify the new observation. However, many other strategies of choosing the "centers" of the classes (like averaging all class members, or taking their median, for example) and distances to these centers can be employed, each leading to a slightly modified version of the algorithm. This class of *nearest centroid* classifiers is commonly employed in genomic applications because, despite not being necessarily the best in term of performance, it generally leads to simple classification rules that are readily interpretable and have reasonable performance.

2.4. Top scoring pairs

Top scoring pairs (TSPs) (Geman et al, 2004) are simple two-genes binary classifiers, in which the prediction of the class label is based solely on the relative ranking of the expression levels of the two genes. The rank--based approach to classification ensures a higher degree of robustness to technical variations and makes the rule easily portable across platforms. Also, the direct comparison of the expression level of the genes is easily interpretable in the clinical context, making the TSPs attractive for medical tests.

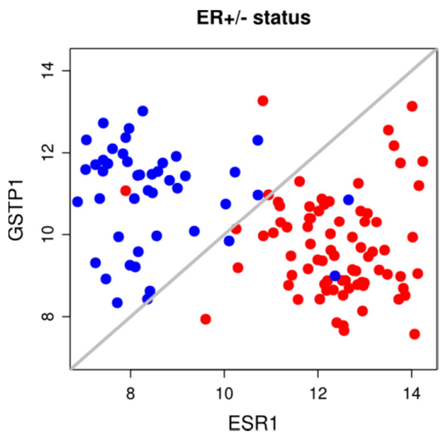
Let again $\mathrm{x} = [x_i]_{i=1,\dots,p} \in \mathbb{R}^p$ be a vector of measurements (e.g. gene expression) representing a sample and let the corresponding class label be $y = \pm 1$. Then, for all pairs of variables i and j, a score is computed,

$$s_{i,j} = P\big(x_i < x_j \big| y = 1\big) - P\big(x_i < x_j \big| y = -1\big),\ \ 1 \le i,j \le p$$

where P are conditional probabilities and the corresponding decision rule is: if $x_i < x_j$ then predict $y = 1$, otherwise $y = -1$. The pair with the highest score or the top k pairs are then considered for the final model (Geman et al, 2004; Tan et al, 2005).

Remarkably, this method does not require the optimization of any parameter and does not depend on any threshold. Figure 3 shows an example of a TSP predicting the estrogen receptor status. The decision boundary (in grey) is always a line with a slope of 1.

Figure 3. Predicting estrogen receptor status: if GSTP1 < ESR1, then the sample is considered ER+ (red dots), otherwise ER- (blue dots).



3. Performance parameters and performance estimation

In the context of clinical applications, a classifier is seen as a test and its continuous value $f(\mathrm{x})$ is called *score*. This score is discretized into two (binary tests) or more categories by using a number of thresholds (or cut-offs) and a prediction about the patient is made based on the predicted category. For example, a test can be used to predict if a patient has a given disease (binary test), or to which of a number of risk groups he/she belongs (e.g. low, medium and high risk groups – categorical tests). By convention, we will say that an individual which is predicted to have the disease to be positive for the test.

Several categories of medical tests are more common:

- *diagnostic* tests are designed to detect the 'diseased' condition in a patient;
- *prognostics* tests try to predict an outcome of interest, like 'recurrence' vs. 'no-recurrence';
- *predictive* tests are used to detect which patients may/may not respond to a treatment; and
- *screening* tests are usually applied to a large population of normally healthy individuals in which the disease has low prevalence, and are usually followed by other confirmatory tests.

Each of these tests is designed to work in specific settings. For example, we require a screening test to detect all (or, say 99%) of all diseased cases (must be sensitive), even if it will produce a relatively high rate of false alarms (false positives). In contrast, a diagnostic test must be sensitive and with low false positive rates. On the other hand, as we have seen from the Bayes decision theory, the prior distributions influence the final decision. A screening test is used in a population where the positive cases have a low prevalence: for example, in 2009 breast cancer in UK had an age-standardized incidence of 124.4 cases in 100 000 women, so we can set a prior $P(y = disease) = 0.125$. A diagnostic test which is applied to confirm a screening test will work on a population with a much higher incidence of the disease, so a possible prior would be $P(y = disease) = 0.75$. The screening and diagnostic tests could be the same, with the only difference being the value of the threshold for the score, above which we call a patient diseased. And this threshold is optimized based on the prior probabilities.

In the following, we will briefly review some of the performance parameters that are used for characterizing the classifiers. For a comprehensive treatment of the subject in the context of clinical applications, see (Pepe, 2003).

3.1. Threshold-dependent performance parameters

We will use the following convention for calling the classes:

- *true label* (disease status) is denoted by D:

$$D = \begin{cases} -1, & \text{if non-diseased} \\ 1, & \text{if diseased} \end{cases}$$

- *predicted label* is denoted by Y:

$$Y = \begin{cases} -1, & \text{if negative for the test} \\ 1, & \text{if positive for the test} \end{cases}$$

A continuous score $f(\mathrm{x})$ is converted into a prediction by $\mathrm{sign}(f(\mathrm{x}) - \theta)$, where θ is a threshold.

For a given observation x one of the following 4 situation may arise:

- $D = 1, Y = 1$: *true positive* – the prediction and the true label are both indicating a diseased case
- $D = -1, Y = -1$: *true negative* – the prediction and the true label are both indicating a non-diseased (healthy) case
- $D = 1, Y = -1$: *false negative* – the test fails to detect the disease status

- $D = -1, Y = 1$: *false positive* – the test predicts as diseased a healthy case

In assessing the performance of a classifier/test one is interested in estimating the probabilities of each of the above four events to occur. The estimation is done based on the respective frequencies in a test set. One usually constructs a *confusion matrix* containing counts of the observed occurrences (Table 1), from which the probabilities are estimated.

Table 1. The confusion matrix and the associated probabilities.

	True labels (gold standard)		
Predicted labels	$D = -1$	$D = 1$	Marginal probabilities
$Y = -1$	True negatives $P(Y = -1 \mid D = -1)$	False negatives $P(Y = -1 \mid D = 1)$	$P(Y = -1)$
$Y = 1$	False positives $P(Y = 1 \mid D = -1)$	True positives $P(Y = 1 \mid D = 1)$	$P(Y = 1)$
Marginal probabilities (priors)	$P(D = -1)$	$P(D = 1)$ (prevalence)	

The following performance parameters are some of the most commonly used criteria for judging a diagnostic test:

- *disease-centric* measures the performance predicting the disease: The *true positive/negative fractions (TPF, FPF)*:

  $$\text{TPF} = P(Y = 1 \mid D = 1), \text{FPF} = P(Y = 1 \mid D = -1)$$

  They are both needed to characterize the test and they are dependent on the chosen threshold. If one knows the disease prevalence ($P(D = 1)$), then the probability of error can be estimated by

  $$P(Y \neq D) = P(D = 1)(1 - \text{TPF}) + \big(1 - P(D = 1)\big)\text{FPF}$$

  A perfect test will have TPF = 1 and FPF = 0. The TPF is also called *sensitivity*, while $1 - \text{FPF}$ is called *specificity*. In the clinical testing literature, the two latter terms are more common than the first ones.

- *predicted values* are used to quantify the clinical value of a test (the likelihood of disease when the test is positive): The *positive/negative predicted values (PPV, NPV)* are defined as

  $$\text{PPV} = P(D = 1 \mid D = 1), \text{NPV} = P(D = -1 \mid Y = -1)$$

  A perfect test will have PPV = NPV = 1, while one totally uninformative, PPV = $P(D = 1)$ and NPV = $P(D = -1) = 1 - P(D = 1)$.

There is a simple connection between the two groups of measures and its derivation is left as an exercise to the reader.

Since the estimators for the above measures are random variables from a Bernoulli trial, one can compute *confidence intervals (CI)*, using any of the proposed methods (e.g. normal approximation, Wilson score, Agresti-Coull, and others (Newcombe, 1998)). Whatever method is used, the confidence intervals (usually 95% CIs) must be reported for a full characterization of the test.

As a final remark, we note that the CIs obtained based on binomial distribution refer to each of the measures individually and do not provide a *confidence region* for the joint distribution of the pairs (TPF,FPF) or (PPV,NPV). To obtain such confidence region, one can use the following result:

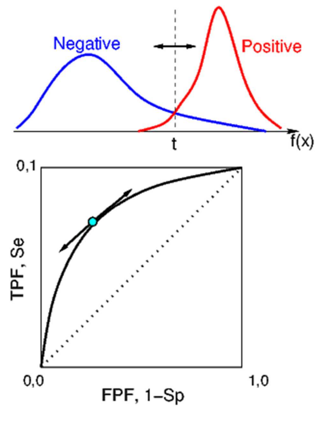
Proposition. If (P_{low}, P_{up}) and $Q_{low}, Q_{up})$ are $1-\alpha^*$ univariate confidence intervals for two binomial random variables P and Q, then the rectangle $\left(P_{low}, P_{up}\right) \times (Q_{low}, Q_{up})$ is a $(1-\alpha)$ confidence region for (P, Q), where $\alpha = 1-(1-\alpha^*)^2$.

For example, from two 95% univariate confidence intervals, one can construct a 90.25% confidence region for the joint variable.

3.2. Threshold-independent performance parameters

We have already noted that the performance measures described in the previous section depend on the chosen value of the threshold θ, and therefore we call them point estimates. However, these tests (classifiers) may need to work in different contexts, where one may want to select a different operating regimen (trade-off between sensitivity and specificity, or PPV and NPV). Moreover, when comparing two tests with different operating regimens, it is difficult to draw any conclusion. it is clear that we need a characterization of the test which is independent of the threshold. The *receiver operating characteristic (ROC) curve* serves exactly this purpose.

Figure 4. Varying the threshold *t* above which a score $f(\mathrm{x})$ leads to a positive test, generates a ROC curve in the (FPF, TPF) space.



By letting the TPF and FPF varying with the threshold,

$$\mathrm{TPF}(\theta) = P(f(\mathrm{x}) \geq \theta | D = -1)$$

$$\text{FPF}(\theta) = P(f(\text{x}) \geq \theta | D = 1)$$

we obtain the definition of the ROC curve:

$$ROC = \{(\text{FPF}(\theta), \text{TPF}(\theta)) \mid \forall\theta \in \mathbb{R}\}$$

It is easy to see that the ROC function is monotone increasing and that it is invariant to strictly increasing transformation of the scores. The parametric form of the curve is given by

$$ROC = \left\{\left(\alpha, \text{TPF}\left(\text{FPF}^{-1}(\alpha)\right)\right) \mid \forall\alpha \in (0,1)\right\}$$

A summary of the ROC curve is obtained by taking the area under the curve (AUC):

$$AUC = \int_0^1 \text{ROC}(\theta)\, d\theta$$

AUC is lower-bounded by 0.5 (corresponding to a totally uninformative test) and upper-bounded by 1. It can also be seen as the Mann-Whitney-Wilcoxon U-statistic: $\text{AUC} = P(Y_{D=1} > Y_{D=-1})$, i.e. the probability of correctly ordering a random pair of cases.

3.3. Performance estimation

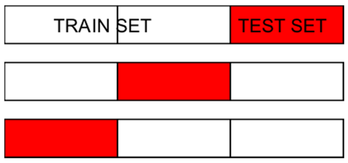
Once a classifier is trained, one has to estimate its performance on unseen data. Lacking access to the full data collection on which the classifier will be applied, one will have to rely on statistical estimates of the performance. The easiest estimate would be the one obtained by applying the classifier on the same data used for training it (plug-in estimate). Except for a few rare cases, this estimate will be optimistically biased, i.e. will underestimate the error rate. Furthermore, relying on the plug-in estimate will more often than not lead to overfitting the training set, i.e. one will find the parameters such that the classifier will have minimum error rate on the training set, but it will perform poorly on new data. The morale is that the estimation of performance has to be done on an independent data set, completely different than the one used for building the classifier.

One possible option would be to randomly split the data available into two disjoint subsets, one used for building the model and one for estimating its performance (*split sample validation* or *holdout validation*). While appealing, this method has at least two drawbacks: it does not use the available data in an optimal way and the training set is reduced drastically in comparison with the original sample size. However, the true validation of a classifier, diagnostic test remains its long run application on unseen data.

In order to better use the training data, several *resampling methods* have been proposed, among which: the k-fold cross-validation, Monte Carlo cross-validation, leave one out cross-validation, bootstrapping, etc. They all have in common the idea of repeatedly randomly partitioning the available into a training set and a validation set. The training set thus obtained is used for full model construction (including feature selection, meta-parameter optimization, model selection, etc.), while the validation set is used for obtaining intermediate estimates of the performance. At the end of the procedure, the intermediate estimates are aggregated into a final value (usually be averaging, but more sophisticated estimates can be used – see for example the .632 estimator below) and a measure of variability of the estimate is also computed (variance, standard error, confidence intervals). These methods differ in the strategy they use for partitioning the data. We will briefly describe some of them here, while for others the reader is referred to (Duda et al, 2001; Hastie et al, 2009).

- *k-fold cross validation* splits the data into k partitions and uses each of them in turn as validation set. Typical values for k are 5 and 10 and the choice represents usually a trade-off between a reasonable training set size and the computational burden, as the procedure is repeated k times. Note that any two models built in this setting share k-2 folds as training data. This means that the predictions are not totally independent so the variance of the estimates is usually underestimated. An improved performance estimation is obtained by repeating the k-fold cross validation on randomly shuffled versions of the original set (the so called *repeated k-fold cross validation*). The final estimate of the performance (e.g. error rate, sensitivity, specificity, etc.) is the average of the intermediate estimates.

  Figure 5. A 3-fold cross-validation scheme: each of the folds is used once and only once as validation set (the red block).

- *leave-one-out cross-validation* is an extreme case of k-fold cross-validation for the case k=1. The training/testing steps are repeated *n* times, where *n* is the sample size.

- *Monte Carlo cross-validation*: repeatedly splits randomly the data set into a training and validation set. For example, it retains 2/3 of the data in training and 1/3 for validation. The procedure is, in fact, a sequence of split-sample validations applied on random permutations of the data. Because of the random split of the data, the procedure does not ensure that all points are used for training and validation.

- *bootstrapping*, in contrast with the above methods, resamples *with replacement* from the original data set, generating new training sets (bootstraps) of the same size *n*. It means that the new training sets may contain duplicated training examples, while other samples are not included. On average, the bootstraps contain 0.632 of the original set. The procedure is repeated B times.

  The .632 estimator of the error rate (or other performance measure) is given by

  $$\hat{E}_{.632} = 0.368\,\hat{E}_0 + 0.632\frac{1}{B}\sum_{b=1}^{B}\hat{E}_b$$

  where $\hat{E}_0$ is the plug-in error rate on the full training set and $\hat{E}_b$ are the error rate obtained at repetition *b* by applying the classifier on the left out data. The empirical distribution of $\hat{E}_b$ can be used for estimating the confidence intervals (for example, the 0.025 and 0.975 quantiles of this distribution are good estimates for the lower and upper limits of the 95% confidence interval).

All these resampling procedures for performance estimation can be implemented to preserve the proportions of the classes from the original data set. In this case, they are called *stratified* since, indeed, the sampling takes place within strata (levels) of the class label variable.

4. Guidelines for gene-based classifier development

Developing gene-based classifiers poses several specific problems, in addition to the "classical" issues that arise when building predictive models. Some of the specific issues are methodological, while others relate to the utility and relevance of the classifiers built.

4.1. Methodological issues

During the last decade thousands of new gene-based classifiers have been published, covering a large palette of applications. The US Food and Drug Administration, which is responsible for approving new diagnostic tests for medical applications, set up a series of projects to investigate the reproducibility and reliability of decision models built on gene expression data. These projects, gathered under the acronym of MAQC (MicroArray Quality Control) have shown that the technology is mature enough to be used in clinical practice. The second phase (MAQC-II) dealt specifically with classification models (Shi et al, 2010) and put forward a number of recommendations, some of which are mentioned below.

As mentioned before, the data points lies in a high dimensional space where the number of dimensions greatly outnumbers the data set cardinality ($n \ll p$). This makes the problem to be ill-posed, in the sense that, theoretically at least, there may be an infinite number of solutions to a classification problem. This is why building a classifier requires a proper feature (variable) selection before training the model per se, but the methods for performing feature selection are not discussed here. We only mention that the feature selection can be done either independently or jointly with training the classifier (may be embed into the process of classifier training, as in the case of penalized logistic regression, for example), but in any case it is a mandatory step.

In the early phases of development of a new classifier, one usually tries many different algorithms before narrowing the selection to a few of them. The initial set of classifiers to be tried should be rich enough such that a suitable model can be found. MAQC-II has shown that in most cases the simpler methods perform as well as the more sophisticated ones on gene expression data. In this project, more than 30,000 models have been assessed and the conclusion was that the major factor impacting the performance of the models is the problem difficulty and not the complexity of the algorithms thrown at it (Shi at al, 2010). Once the initial exploratory phase completed, the list of candidate models should be short (2-3 models). These candidates should be evaluated on new data and the final model selected. The final model can then be trained and its performance estimated (either by resampling methods or on other independent data). This approach requires a fairly large amount of data but will likely produce a robust model that will not be overfitted to the training data.

The performance estimation is usually the most prone to methodological errors task in building a classifier. In theory, *all the steps performed from raw data to final model* must be included in the cross-validation (or whatever resampling method) loop. However, this is not always feasible: for example, microarray data normalization usually inspects the whole batch of raw samples for producing the normalized data. This means that any input vector for the classifier will be influenced by the information from other vectors in the data set, so the data normalization step has to be performed inside the cross-validation. On the other hand, the normalization step can be quite computationally demanding and repeating it at each iteration will slow down the process of model assessment. While this issue is not well studied in the literature, the common consensus is that the performance estimates are marginally impacted by the inclusion or not of the data normalization step in the cross-validation. That is why the overwhelming majority of studies leave the normalization outside the cross-validation. Nevertheless, apart from the normalization step, all the other processing steps must be included in the cross-validation (ideally, even the model selection step – if a model is selected at the end). Failing to obey this rule will lead to an optimistically biased estimation of performance (Varma and Simon, 2006). By far, the incorrect performance estimation for prediction models is the most common error (Dupuy et al, 2007).

Finally, a question that still lacks a definite answer refers to the samples size needed for developing a gene-expression classifier. Sample size estimation can be done under some parametric assumptions: for example (Dobbin and Simon, 2005) assume a normal multivariate distribution of the classes and derive mathematical formulae for computing the sample size for classifier development and validation. Assumption-free approaches exist and relies on simulations: (Popovici et al, 2010) shows how using the learning curves can be used to estimate if increasing the sample size would bring any benefit for classifier training and what would be the required sample size to achieve a predefined performance.

4.2. Clinical utility and relevance

The final goal of gene based classifiers is to answer a clinical or biology research need, so they have to compete with the current predictive models used in the respective fields. Thus, for the development and validation of clinically-relevant genomic tests a number of key stages must be successfully fulfilled (Simon, 2006):

- identify an important therapeutic decision which would need improvement;
- the target patient population should be homogeneous enough and treatment uniform, so that the results would be therapeutically relevant. Also, the economic considerations should not be overlooked: the treatment options and costs of misclassification should be such that the resulting classifier/test would be likely to be used in clinical practice (the test itself would incur some costs as well);
- develop the classifier and perform internal validation to assess whether the classifier appears to be sufficiently accurate relative to standard prognostic factors currently used. This means that initial analysis should prove the superiority (performance and/or costs at equal performance) of the new test with respect to current practice;
- translate the classifier to a platform likely to be used in practice. For example, a classifier relying on the use of several (in the order of tens) genes, even though it could be develop from microarray data, it is more likely that it implementation on qPCR would be more appealing to the clinicians/laboratories;
- demonstrate reproducibility of the results;
- independent validation of the complete test in prospective clinical trials.

5. Examples of gene-based classifiers

A simple search on PubMed (www.pubmed.com) portal for scientific literature lists hundreds of papers proposing new gene expression classifiers (sometimes called biomarkers), reflecting the importance these tools gained in the biomedical research. It would be a futile and inherently subjective attempt to list here "the most representative" results in the field. Therefore we will limit ourselves to mention just three such classifiers and some of their applications, each of them having something particular that makes them to stand out of the crowd.

5.1. Golub's ALL vs AML classifier

Golub's classifier (Golub et al, 1999) represents one of the first classifiers built in the early days of the microarrays. It was designed to distinguish between acute lymphoblastic leukemia (ALL) and acute myeloid leukemia (AML), and was addressing the need for a standardized test to establish the diagnostic. Their training set consisted of $n = 38$ cases (27 ALL, 11 AML) profiled on an early Affymetrix chip ($p = 6817$ genes). They identified 50 genes correlated with the class distinction (based on a signal-to-noise ratio measure) and combined the genes into a score by "weighted vote" (i.e. linear combination of genes' expression values). And then they validated their predictor on an independent collection of 34 samples. By today's standards, this represents and easy problem, nevertheless the merit of this first system was to prove that building classifiers on gene expression is not only feasible but could solve important diagnostic problems. The fact that their classifier relied on know oncogenes (like c-MYB, HOXA9) strengthen the confidence in such decision systems.

5.2. Compound covariate predictor

In (Radmacher et al, 2002) a generalization of Golub's classifier was proposed. Again, for a specimen i a score is computed as a weighted sum of the expression values of a number of genes,

$$s_i = \sum_j t_j \, x_{ij}$$

where the weights t_j are the signed t-statistics measuring the association of gene j with the class to be predicted. The sign is indicating if the gene is positively or negatively associated with the class. The score is then compared to a threshold computed as the average of the mean scores of each class. This *compound covariate predictor* is prototypical for large number of classifiers based on gene expression. It has the appealing property of being easily understandable as each gene contributes to the score proportional to its fold change between the two classes.

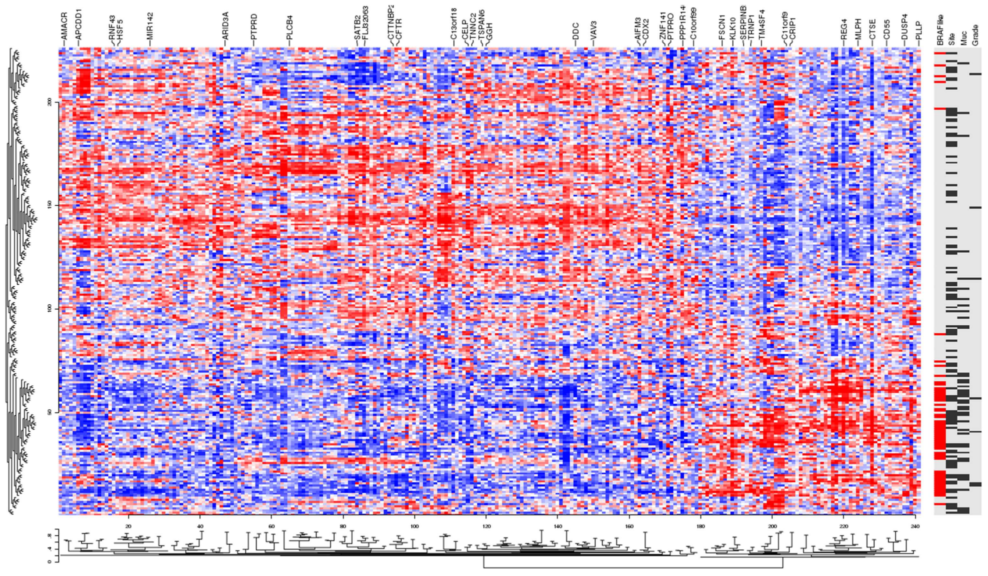
5.3. Top scoring pairs

The final example of gene based classifier is represented by Geman's Top Scoring Pairs (TSP) classifier (Geman et al, 2004), described in section 2.4. The striking feature of this classifier is its simplicity: for easier classification problems, it suffices to compare the expression levels of only two variables (genes) for taking a decision. However, as this is seldom enough for most of the problems, extensions of this algorithm have been proposed in which the top pairs are combined by majority vote (Tan et al, 2005) or by weighted combinations (Popovici et al, 2011). Despite its apparent simplicity, the classifier performs remarkably well on a large number of problems. Moreover, as the decision is taken by comparing the relative order of two genes, the classifier is extremely robust to noise and translates well from one platform to another.

Recently, this classifier was used to build a predictive model for identifying the colorectal cancer patients harboring a BRAF mutation (Popovici et al, 2012). In this study, the authors used the TSP to build a 64-gene-based classifier (32 pairs) to distinguish the BRAF mutant patients from those BRAF wild type and KRAS wild type. The training set consisted in 431 cases (of which 47 were BRAF mutants). Despite the highly imbalanced settings, the classifier's estimated performance (using repeated 5-fold cross-validation) was extremely good (sensitivity 95.8% and specificity 86.5%). The proper use of cross-validation procedure led to an accurate estimation of the performance, as the independent validation has shown: on three external data sets, the aggregated performance was: sensitivity 96.0% and specificity 86.24%. The classifier has demonstrated its good robustness, as the external validation sets were originating from different microarray platforms than the training set.

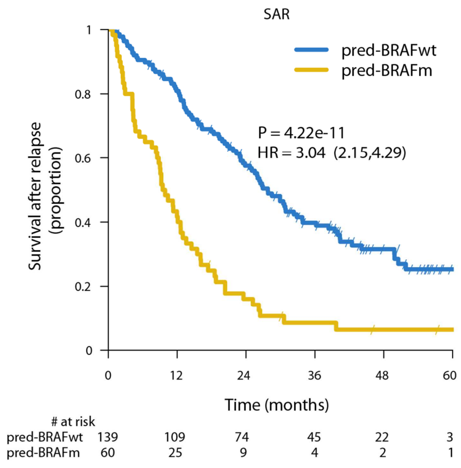
While the original purpose of the classifier was to predict the BRAF mutant patients, when applied to KRAS mutant population (which was not part of the training set) it segregated it into two subpopulations with clearly different gene expression patterns (on selected differentially expressed genes) – Figure 6.

Figure 6. Heatmap showing the different expression patterns within the KRAS mutant population, between BRAF-mutant-like patients (those predicted by the classifier, marked in red in the right column) and the rest of KRAS mutants.



The classifier had also strong prognostic value, i.e. it predicted the high risk patients. For examples, Figure 7 shows the Kaplan-Meier curves for the two populations predicted by the classifier (pred-BRAFm stands for "predicted BRAF mutant", while pred-BRAFwt for "predicted BRAF wild type"). This discovery is of clinical relevance, since it identifies a larger population at risk than initially considered by the clinical practice. Also, it opens new interventional avenues which would target specific pathways active only in this "BRAF-mutant-like" population. Finally, it is of importance also for the design of clinical trials since it clearly shows that the KRAS mutant population is not homogeneous and extra stratification factors should be taken into account.

Figure 7. Survival after relapse: patients predicted to be "BRAF mutant" form a high risk group, with a median survival time of about 12 months.



6. Some concluding remarks

In this chapter we tried to briefly present a number of key concepts for understanding the classifiers in general and the specific issues arising from their application in the context of gene expression data. While for optimal application of classification algorithms intimate knowledge of the theory underlying their development is needed, for making good use of them a more superficial understanding of the principles of rigorous classifier development is enough. What remains extremely important is to understand the risks resulting from improper validation and performance estimation: the classifiers will never perform as expected.

References

Duda RO, Hart PE, Stork DG. 2001. Pattern classification. 2nd edition. John Wiley and Sons

Dupuy A, Simon R. 2007. Critical review of published microarray studies for cancer outcome and guidelines on statistical analysis and reporting. Journal of National Cancer Institute 99:147-157

Geman D, D'Avignon C, Naiman DQ, Winslow RL. 2004. Classifying gene expression profiles from pairwise mRNA comparisons. Statistical Applications in Genetics and Molecular Biology 3(1):19

Golub TR, Slonim DK, Tamayo P, Huard C, et al. 1999. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science vol. 286.

Hastie T, Tibshirani R, Friedman J. 2009. The elements of statistical learning. 2nd edition. Springer Verlag

Newcombe RG. 1998. Two-sided confidence intervals for the single proportion: comparison of seven methods. Statistics in Medicine 17:857-872

Pepe MS. 2003. The statistical evaluation of medical tests for classification and prediction. Oxford University Press

Popovici V, Chen W, Gallas BG, Hatzis C, Shi W, Samuelson FW, Nikolsky Y, Tsyganova M, Ishkin A, Nikolskaya T, Hess KR, Valero V, Booser D, Delorenzi M, Hortobagyi G, Shi L, Symmans WF, Pusztai L. 2010. Effect of training sample size and classification difficulty on the accuracy of genomic predictors. Breast Cancer Research 12(1):R5

Popovici V, Budinska E, Delorenzi M. 2011. Rgtsp: a generalized top scoring pairs package for class prediction. Bioinformatics 27(12):1729-1730

Popovici V, Budinska E, Tejpar S, Weinrich S, Estrella H, Hodgson G, Van Cutsem E, Xie T, Bosman FT, Roth AD, Delorenzi M. 2012. Identification of a poor-prognosis BRAF-mutant-like population of patients with colon cancer. Journal of Clinical Oncology 30:1288-1295

Radmacher MD, McShane LM, Simon R. 2002. A paradigm for class prediction using gene expressino profiles. Journal of Computational Biology 9(3):505-511

Shi L, MAQC consortium. 2010. The MicroArray Quality Control (MAQC)-II study of common practices for the development and validation of microarray-based predictive models. Nature Biotechnology 28(8)

Simon R. 2006. Roadmap for developing and validating therapeutically relevant genomic classifiers. Journal of Clinical Oncology 23:7332-7341

Tan AC, Naiman DQ, Xu L, Winslow RL, Geman D. 2005. Simple decision rules for classifying human cancers from gene expression profiles. Bioinformatics 21(10):3896-3904

Varma S, Simon R. 2006. Bias in error estimation when using cross-validation for model selection. BMC Bioinformatics 7:91