Feature selection via Boolean Independent Component analysis.

Bruno Apolloni\textsuperscript{a} Simone Bassis\textsuperscript{a} Andrea Brega\textsuperscript{b}

\textsuperscript{a}Dipartimento di Scienze dell’Informazione Università degli Studi di Milano Via Comelico 39/41, 20135 Milano, Italy.

\textsuperscript{b}Dipartimento di Matematica “F. Enriques” Università degli Studi di Milano Via Saldini 50, 20133 Milano, Italy

Abstract

We frame feature selection as a representation problem within a wider task of clustering feature vectors, and root its solution on a special procedure for extracting from them a set of boolean components that we expect to be independent. The overall clustering procedure is based on a divide et impera strategy: first give data a suitable representation then compute an assignment function. With the former we aim to find components of the feature vector minimizing – with the help of a special Schur-concave function – the mutual information between data and cluster features. We assess a subsymbolic tool to implement the optimization process and wisely use clustering algorithms to complete the procedure.

We adopt the crucial problem of feature selection of DNA microarray data in cancer diagnosis as a benchmark to toss various aspects of the procedure.

Key words: Feature selection, BICA, Independent Component Analysis, clustering, DNA microarray classification.

1 Introduction

Large databases such those containing DNA microarray data represent typical instances where technology goes beyond the capability of their exploitation. A microarray probe produces hundreds of thousands of real valued data that we know to be related to the functional status of the body cells they refer to, yet we do not know how they are related. We may face similar scenarios when we

Email addresses: apolloni@dsi.unimi.it (Bruno Apolloni), bassis@dsi.unimi.it (Simone Bassis), brega@mat.unimi.it (Andrea Brega).
are called to analyze as much huge benches of data available for instance with stock market or bank accounts, and that may be described by very long records contrasted with poor tools for their effective comprehension. A main request with these records is to discriminate the information that is relevant to the goal of the current analysis from the rest of the data, that constitute a confusing garbage that overload computations in the particular instance. If we associate information with specific fields in the record, the fields go to represent features of the phenomenon the records refer to and the above discrimination translates into the common task of feature selection. It is a much sophisticated task whose achievement depends on the peculiarities of the questioned features. In this paper we will have the microarray gene analysis as a leading scenario. Because of its complexity and the current challenging diagnostic problems it raises, this task asks for integrated methodologies that may be exported in similarly challenging application fields, such as the mentioned stock market and credit assignment.

To frame our goal, we recall that different approaches have been proposed for relating gene expression patterns with diseases, clinical outcome and treatment response to drugs. They range from methods based on classical statistics, such as t-like statistics [1], correlations [2], sparse logistic regression with Bayesian regularization [3] and other Bayesian-based feature selection approaches [4] to pure subsymbolic methods, such as Neural Networks [5] or genetic algorithm-based schemes [6]. The fact is that we have a mix of two hard problems to solve: i) a suitable quantization of the signals produced by the target probe hybridization in order to capture the essential traits of the gene expression, and ii) the relation between genotypic and phenotypic expressions, that is to say between gene expression and diagnosis. Now, on the one hand the huge amount of data per probe imposes the use of either statistical or subsymbolic methods for getting rid of the mess of data. On the other, the relative low number of examples available for a given analysis (a many data-few samples coupling that is generally denoted the curse of dimensionality [7]) and the complexity of their interpretation requires injecting in the processing as much symbolic knowledge as is available to obtain a final model of the data. Combining the two aspects leads the researcher into the realm of hybrid systems having diagnosis as a learning task.

A divide et impera strategy has usually been adopted by researchers to face the complexity of the data analysis: first make them manageable, then try to draw a diagnosis from them. The initial phase aims at reducing the number of boiling up variables, either by selecting a subset of probe data, playing the role of relevant features of the gene expression, or by mapping the data into new variables that may simply represent a quantization of the probe data or completely new ones variously connected to the original data. The former is the favorite strategy, giving rise to feature selection methods intended to select the most ”relevant” features [8]. The expected benefit is twofold [9].
On the one hand, by reducing the dimensionality of the problem we may improve the performance of classification algorithms for diagnosis and outcome prediction. On the other, since gene expression data represent activation values of single different genes, we may select subsets of genes mostly related to a specific disease. This has relevant applications in bio-molecular medicine and pharmacogenomics [10]. A typical broad taxonomy of the methods commonly employed to reduce dimensionality in data analysis splits into the following three families [8]:

- **Filter methods**: they are preprocessing methods. They attempt to assess the merits of features from the data, ignoring the effects of the selected subset on the clustering performance. Examples are methods that select variables by ranking them using compression techniques (like PCA), or by computing correlation with relevant parameters, or by using information theoretic ranking criteria [11]. These approaches are usually univariate and implicitly assume independence between features. For statistical methods to assess in particular the significance of ranked genes see [12].

- **Wrapper methods**: these methods assess subsets of variables according to their usefulness to a given predictor [13]. They conduct a search for a good subset using the clustering efficacy itself as part of the evaluation function. The problem boils down to a problem of stochastic state space search. One example is the stepwise methods proposed in linear regression analysis [13].

- **Embedded methods**: they perform variable selection as part of the clustering procedure and are usually specific to given learning machines. Examples include: i) classification trees [14], ii) regularization and shrinking techniques to trim the hypothesis space by constraining the magnitude of the parameters [15], and iii) SVM and kernel-based methods [16]. With DNA microarray data, regularization is mostly a part of many classification algorithms for feature selection rather than a stand alone method [17].

The second step of the *divide et impera* approach has recently been implemented through ensemble methods adapted for the specific task: they range from boosting [18,19], bagging [20], Error Correcting Output Coding [21], to Random Subspace ensemble methods [22]. In particular, SVM and kernel methods within an ensemble of learning machines will be part of the procedure we propose in this paper.

Although we are unable to overcome the *divide et impera* dichotomy, our approach starts by considering the true goal of the analysis underlying feature selection that is a clustering of the feature vectors – in particular gene expression data – in order to classify them in view of a specific application – for instance for cancer diagnosis. Hence the processing of the data is primarily finalized to this target. More precisely, we map feature data into arrays of independent bits as the most essential form of information compression that we will use to classify the examples (hence we adopt the second strategy for first
making data manageable). We call BICA (Binary Independent Components Analysis) the procedure assessed to this aim that we implement on a specially featured neural network. In spite of an obvious propensity for Boolean formulas, we found the most efficient classification tool in the ensembles of SVMs on a space augmented through the kernel trick. Our final target is, however, not classifying data in a way that ultimately proves subsymbolic. Rather, we search for a real explanation of the clustering in face of the feature data. Hence we aim at a selection of a small number of features on which to build up simple formulas resuming the clustering. Though the whole contrivance is designed as a general purpose procedure, the latter is a hard task that find specific solutions in the single instances in a human-centric way. Namely, we analyze with the help of the field specialists the features mapping into the Boolean variables that prove more statistically relevant to determining the hyperplanes of the SVM ensemble. With reference to our leading example, in this way we obtain a semi-authoromatic procedure to select features that may explain the phenotypic expression of the probes under analysis.

To illustrate our procedure, the paper is organized as follows. In the next section we face the problem of compressing information so as to get a reduced number of working variables, i.e. the BICA procedure. Then in Section 3 we describe the ensemble of classifiers concluding the clustering. In Section 4 we complete the procedure with steps devised to identifying and interpreting subsets of features that are expectedly responsible for the discrimination operated with the clustering. The effectiveness of the whole procedure is tossed on two benchmarks in Section 5, and in Section 6 we make some concluding remarks.

2 The Representation Problem

The reason why we deal with hundreds of thousands of data produced by probes is that we want to classify the patient the probe refers to either as healthy or ill (namely, belonging to some specific illness category). Analogously a bank clerk manages large client databases to discover the most reliable ones, an investor the most promising stock options, etc. All these objectives call for classification rules that are beneficial of skillful clustering tasks.

A key parameter referring for the difficulty of these tasks is the ratio between the number of processed records ant their length. Thus, starting with a few dozen of already classified examples, we want to discover a classification rule that promises to be suitable also for new cases. This results in the solution of a clustering problem constrained to output the known cluster label on each probe. At the outset, given a set of $m$ objects that we identify through a set of vectorial patterns $y^m = \{y_1, \ldots, y_m\}$, the clustering problem is to group them into $n$ clusters $\{d_1, \ldots, d_n\}$ in a suitable way. We denote by $d_i$ the $i$-
th class, by $\Delta$, the decision of attributing a generic pattern $y$ to it, and by $l(\Delta_i, d_j)$ the comparative cost (loss function) of attributing $y$ to $d_i$ in place of $d_j$. In this framework, we identify the problem of establishing clusters of patterns with that of maximizing the cost $C$ of an incorrect attribution of patterns to clusters, i.e. of misclassification once we decide to actually use clusters as classes. Hence, we want to partition the pattern space $\mathcal{Y}$ into $n$ subsets through a decision rule $\Delta(y): \mathcal{Y} \to \{d_1, \ldots, d_n\}$ such that

$$\Delta = \arg\max_{\Delta} C(\Delta) = \arg\max_{\Delta} \left\{ \sum_{y \in \mathcal{Y}} \sum_{j=1}^{n} l(\Delta(y), d_j) \right\}$$

(1)

whose solution depends on the shape of $l$, i.e. we want to sharply discriminate the cluster on the basis of the loss function. For instance, for $l$ identified with the Euclidean distance, i.e. with the $l_2$ metric

$$l(\Delta(y), d_j) = \begin{cases} 0 & \text{if } \Delta(y) = d_j \\ (y - \mu_{d_j})^T(y - \mu_{d_j}) & \text{otherwise} \end{cases}$$

(2)

where $\mu_{d_j}$ plays the role of representative of class $d_j$, then the solution is

$$\Delta(y) = \arg\min_j \left\{ (y - \mu_{d_j})^T(y - \mu_{d_j}) \right\}$$

(3)

$$\mu_{d_j} = \frac{1}{\nu_j} \sum_{i=1}^{\nu_j} y_{i_j}$$

(4)

being $\nu_j$ the number of objects with index $i$ attributed to the $j$-th cluster. 1

Denoting by $H(X)$ and $H(X|Z)$ the entropy of $X$ and the conditional entropy of $X$ given $Z$ respectively, for $Y$ normally distributed around the representative of its cluster the above rule consists in the minimization of conditional entropy $H(Y|D)$ of the data given the distribution of the clusters 2, rather, of its sample estimate. Indeed, by definition

$$H(Y|D) = - \sum_{d_i} p_{d_i} H(Y|d_i)$$

(5)

where $p_{d_i}$ is the probability measure of cluster $d_i$. With this notation

$$f_Y(y) = \sum_{i=1}^{n} p_{d_i} f_{Y|D=d_i}(y) = \sum_{i=1}^{n} p_{d_i} \frac{1}{(2\pi)^{n/2}} \exp \left[ -(y - \mu_{d_i})^T(y - \mu_{d_i}) \right]$$

(6)

1 Note that rule (3) and templates (4) comes from the conventional Bayesian approach to clustering [11] as well. Our enunciation of the problem however does not requires the existence apriori of true classes of data. Rather, they come from the use as a suitable way of organizing data, in the thread of Algorithmic Inference [23].

2 By default capital letters (such as $U$, $X$) will denote random variables and small letters ($u, x$) their corresponding specifications.
Hence

\[ H(Y|D) = \sum_{d_i} p_{d_i} E[(Y - \mu_{d_i})^T(Y - \mu_{d_i})] + a \]  

(7)

where \( a \) is a constant and \( E \) denotes the expectation operator. By denoting with \( I(X|Y) \) the mutual information between \( X \) and \( Y \) and rereading in true entropic terms our clustering strategy, the general goal of any useful mapping from \( Y \) to \( D \) is to ensure a high mutual information [24]. Now, in the case where the mapping realizes a partition of \( Y \) range, hence in the case of univocal mapping, we have the following expression of the mutual information.

**Lemma 1** For any univocal mapping from \( Y \) to \( D \),

\[ H(Y|D) = H(Y) - H(D) \]  

(8)

\[ I(Y, D) = H(D) \]  

(9)

**PROOF.** Let us prove the claim in the case of discrete \( Y \), hence specifying into patterns indexed by an integer \( k \), and \( D \equiv L \) ranging in \( \{0, 1\} \), the extension to the general case being trivial. With these conditions we have

\[ H(Y) = - \sum_k p_k \ln p_k = - \sum_{i \in A} p_i \ln p_i - \sum_{i \notin A} p_i \ln p_i = - \left( \sum_{i \in A} p_i \right) \ln \left( \frac{p_i}{\sum_{i \in A} p_i} \right) - \left( \sum_{i \notin A} p_i \right) \ln \left( \frac{p_i}{\sum_{i \notin A} p_i} \right) = - p_A \ln p_A - p_{\bar{A}} \ln p_{\bar{A}} = H(Y|L) + H(L) \]  

(10)

where \( A \) is the set of the indices of \( Y \) patterns mapping to 1, \( \bar{A} \) its complement, and \( p_x \) is the probability of the pattern or set denoted by the index \( x \). Solving the equality in \( H(Y|L) \), we obtain the first claim. Coupling this result with the definition \( I(Y, L) = H(Y) - H(Y|L) \) we obtain \( I(Y, L) = H(L) \) which brings us to the second claim.

Claim (9) says that a clustering so more preserves information of patterns the more the entropy of clusters is higher, i.e. the more they are detailed. Claim(8) denotes the gap between entropy of patterns and entropy of clusters that is managed by the clustering algorithm. As \( H(Y) \) is not up to us, the algorithm may decide to group the patterns so as to increase \( H(D) \), in line with the claim (9) suggestion.
In a case such as ours where the labels of the patterns are given and no other information about them is granted, we are harmless with respect to the $H(D)$ management. Rather, our problem is to find an algorithm that reproduces the given labeling. If the task is hard, we may try to split it into two steps: i) improve the pattern representation so that their label may be more understandable, and ii) find a clustering algorithm with this new input. This corresponds to dividing the gap between $H(Y)$ and $H(D)$ in two steps and, ultimately, looking for an encoding $Z$ of $Y$ minimizing $H(Z|D)$, i.e. the residual gap $H(Z) - H(D)$. We have two constraints to this minimization. One is to maintain the partitioning property of the final mapping. Hence

**Definition 2** Consider a set $A$ of $Y$ patterns, each affected by a label $d \in D$. We will say that an encoding $Z$ of $Y$ is correct if it never happens that two patterns of $A$ with different labels receive the same codeword. \(^3\)

The second constraint is strategic: as we do not know the algorithm we will invent to cluster the patterns, we try to preserve almost all information that could be useful to a profitable running of the algorithm.

The second constraint is somehow fuzzy, since *vice versa* our final goal is to reduce the mean information of the patterns exactly to its lowerbound $H(D)$. A property that is operationally proof against the mentioned fuzzyness is the independence of the components of $Z$. The following claim is connected to a special way of minimizing $H(Z)$. This is why we speak of operationally proofness. It is however a very natural way in absence of any further constraint on $Z$.

**Lemma 3** Consider a set $A$ of $Y$ patterns and a probability distribution $\pi$ over the patterns. Assume that for any mapping from $Y$ to $Z$, denoting $H(Z)$ w.r.t. the corresponding distribution over the corresponding $Z$ patterns, there exists a mapping from $Y$ to $Z'$ with $H(Z') = H(Z)$ such that $Z'$ are independent. Then the function

$$
\tilde{H}(Z) = - \sum_k p_k \ln p_k - \sum_k (1 - p_k) \ln (1 - p_k)
$$

(11)

with $k$ spanning the image of $A$, has minima over the above mappings in $Z$s having independent components.

**Proof.** $H(Z) = \tilde{H}(Z) - I(Z)$ where $I(Z)$ denotes the overall mutual information between $Z$ components, a term that is always positive and 0 only when the components are independent. Hence for each $Z$ with non independent components $Z'$, we have $H(Z) > \tilde{H}(Z) - I(Z) > \tilde{H}(Z') - I(Z')$. This proof is similar to the one of the previous claim.

\(^3\) of course we will check this property on the available patterns, with no guarantee as to any future pattern we will meet.
ponents there exists a $\mathbf{Z}'$ with independent components and $\tilde{H}(\mathbf{Z}') < \tilde{H}(\mathbf{Z})$ by hypothesis.

Finally, with the aim of lowering the vagueness of our entropic goal, we decide in this paper to have Boolean $\mathbf{Z}$, as a way of forcing some reduction of data redundancy and information as well, in the direction of taking a discrete decision, ultimately, the clustering. This produces the side benefit of a concise description of both patterns and clustering formulas as a nice premise for a semantic readability of them. To this end, and limiting ourselves to a binary partition of the patterns, we assume as cost function of the single pattern $s$ in our coding problem the following Schur-concave function [25] which we call edge pulling function:

$$E_s = \ln \left( \prod_{k=1}^{n} z_{s,k}^{-z_{s,k}} (1 - z_{s,k})^{(1 - z_{s,k})} \right)$$

(12)

where $z_{s,k}$ is the $k$-th components of the encoding of the pattern $s$. In line with the general capability of Schur-concave functions of leading to independent component located on the boundaries of their domain [26], we may prove that minimizing over the possible encodings the sum $\tilde{H}$ of the logarithm of $E_s$ over all patterns leads us to a representation of the patterns that is binary and with independent bits, which we call BICA representation.

Lemma 4 Any $\mathbf{Z}$ mapping that is correct according to def. 2 and meets assumptions in Lemma 3 while minimizing the edge pulling function (12) is expected to produce Boolean independent components minimizing (5) as well.

PROOF. This function has a minimum when each $z_{s,k}$ is Boolean (see Fig. 1 for $n = 2$). This is why we call it edge pulling function. Moreover, from Jensen inequality [27] on the function $g(x) = x \ln x$ we have that

Fig. 1. Graph of the function $E_s$ with $n = 2$. 
\[ \hat{H} \equiv \frac{1}{m} \sum_s E_s \leq \sum_k \left[ -\sum_s \frac{z_{s,k}}{m} \ln \left( \sum_s \frac{z_{s,k}}{m} \right) \right. \\
\left. - \left( 1 - \sum_s \frac{z_{s,k}}{m} \right) \ln \left( 1 - \sum_s \frac{z_{s,k}}{m} \right) \right] \equiv \hat{H} \quad (13) \]

where \( \hat{H} \) is an estimate of \( H(Z) \), i.e. the entropy of the \( Z \) patterns we want to cluster computed with the empirical distribution law \([28]\) in the assumption that \( Z \) components are independent. Finally, for \( z_{s,k} \) close to either 0 or 1 and binary vectors almost orthogonal (so that also \( \sum_s \frac{z_{s,k}}{m} \) is close to 0), \( g(x) \) behaves in (13) almost linearly, making the inequality almost an equality. Thus, within this range of values we are minimizing \( \hat{H} \) as well, getting an encoding with components that are independent by lemma 3 and Boolean as well.

**Remark 5** The above lemma is only part of the history. In view of minimizing \( \hat{H} \), another lever is represented by the number of bits into which we want to encode the patterns. We will see in the next section that this number is fixed feasibly small so as to limit the inconsistencies of the coded vectors.

### 2.1 The BICA algorithm

We commit to a neural network the task of *correctly* minimizing the edge pulling function (12). Namely, we look for a vector \( v \) of Boolean variables, possibly independent, whose assignments reflect the relevant features of the original data pattern \( x \), where Boolean assignments may coincide only when they code data patterns having the same value of the 0/1 label.

In order to let relevant features emerge, we ask the network to mirror the input pattern, in such a way that the mirroring learning process could help the intent of computing the wanted Boolean variables. In summary, on the one hand, we want to extract independent components of the signals, as the *noble part* of their information content. On the other, stressing the fact that independence is a property of the representation of the data that we use, we search for this property precisely on a Boolean representation of them, suitable for correctly partitioning the data into positive and negative inputs of our decision rule. We get both goals through a specially featured multilayer perceptron.

#### 2.1.1 The architecture

The whole process is done by a neural network with the architecture shown in Fig. 2, sharing the same input and hidden layer with the two output seg-
ments A and B, computing the Boolean assignments and a copy of the input, respectively.

Part A: Propositional Variable Vector \( \mathbf{v} = (v_1, v_2, \ldots, v_n) \)

Part B: Mirroring of Pattern Vector

Hidden Layer

Pattern Vector \( \mathbf{y} = (y_1, y_2, \ldots, y_n) \)

Fig. 2. Layout of the neural network mapping features to symbols.

2.1.2 The learning algorithm

We train this network with a backpropagation algorithm [29] whose characteristics are specified below.

Error backpropagation in part B. Mirroring is a usual functional requirement for an MLP [30]. We structured our network as a three-layer network with the same number of units in both input and output layers and a smaller number of units in the hidden layer. Therefore the hidden layer constitutes a bottleneck which collects in the state of its nodes a compressed representation of the input. This part of the network is trained according to a quadratic error function. Hence the error \( \delta_{s,j} \) which is backpropagated to the hidden layer from each unit \( j \) of this part upon presentation of \( s \)-th input pattern is:

\[
\delta_{s,j} = f'_{\text{act}}(\text{net}_{s,j})(y_{s,j} - z_{s,j})
\]  

where \( \text{net}_{s,j} \) is a weighted sum of the inputs to the unit \( j \) upon presentation of \( s \)-th pattern \( \mathbf{y}_s \), suitably normalized, \( z_{s,j} \) the corresponding output, and \( f_{\text{act}} \) is the sigmoid function [15].

Error backpropagation in part A. Things are different for the units of part A of the output. In this case we require that the network minimizes an error represented by the edge pulling function (12). The error which is backpropagated from the units of part A is:

\[
\delta_{s,k} = f'_{\text{act}}(\text{net}_{s,k})\alpha_{s,k}
\]

where

\[
\alpha_{s,k} = -\frac{\partial E_s}{\partial z_{s,k}} = \ln \left( \frac{z_{s,k}}{1 - z_{s,k}} \right)
\]
2.1.3 Directing the mapping

Using $\alpha_{s,k}$ as in (16), we let the network decide independently about the values of part A to which it will converge for each input pattern. Of course, parameters such as the learning rate, the initial weights and the influence from part B (through the hidden nodes) play an important role in this decision and at the same time constitute the source of randomness of our compression. But we want to govern this sub-symbolic process also with syntactic feedbacks. The general idea is to insert into the $\alpha$ expression an extra term which has the form of ‘directed noise’ added to the initial value of $\alpha$ when we are not satisfied with the correctness of the result. The effect is to ‘shake’ the network in order to search for a new equilibrium. We activate this ‘punishment’ each time we find a pair of patterns with different labels to be coded with binary vectors having the Hamming distance below a given threshold (say 1 or 2). Namely, a positive random term $\tau_{s,k}$ is generated in correspondence to the incorrectly coded pattern $s$, specifically in those nodes (possibly all nodes) that do not increase the Hamming distance from the mate. Its value contributes to $\alpha_{s,k}$ with the following function $\theta_{s,k}$:

$$\theta_{s,k} = (1 - 2\Gamma(z_{s,k})) \tau_{s,k}$$  

where $\Gamma$ is a threshold function. The first term in the brackets specifies the sign of $\theta_{s,k}$ so that the contribution to the network parameters goes in the opposite direction from the one the unit is moving in. Finally, using a tuning parameter $\pi_A$ to balance the mutual relevance of corrections coming from parts B and A, we get the complete expression of $\alpha_{s,k}$ which reads:

$$\alpha_{s,k} = \pi_A \left( \theta_{s,k} + \ln \left( \frac{z_{s,k}}{1 - z_{s,k}} \right) \right)$$

3 The discriminating function

From a strictly algorithmic point of view the value of the coded boolean vectors lies in their ability to cluster the feature vectors. It depends in turn on the efficiency of the clustering algorithm as well. Focusing on a simple partition task – such healthy-ill patient, reliable unreliable client, fruitful - garbage stock option, etc. – the problem specializes in the learning of a suitable discriminating function between the two classes. Despite of many attempts to use the vectors as assignments of propositional variables within a Boolean formula to be learnt through various procedures, such as decision trees [31] or PAC meditation [32], in our leading example we found the most efficient discriminating function to be constituted by an ensemble of SVMs. Namely, a single SVM draws a hyperplane to linearly separate positive vertices from
negative vertices of the Boolean hypercube leaving the most margin between points and plane. This operation is carried out very efficiently by current algorithms solving the problem in a dual space. We do not expect, however, Boolean vectors representative, for instance, of healthy and ill patients to be linearly separated. Thus we augment SVM power in two directions:

- From linear to nonlinear separators. Since the binarization step guarantees that there is a Boolean formula as a separator that is not necessarily a linear separator, we augmented the discriminating power of the hyperplanes produced by SVMs with the help of the kernel trick. As well known, all boils down to the fact that the search for the most margin hyperplane translates into an optimization problem in a dual space whose object depends on the points only through the inner products such as \( z_i \cdot z_j \). We may assume it as a special issue of a symmetric function \( k(z_i, z_j) \) – the kernel – and repeat the computation for any other issue of this function intended as the inner product \( x_i \cdot x_j \) with \( x_i = \phi(z_i) \), and we obtain a fitting of the points according to a linear function on \( x \), hence a possibly non linear function on \( z \).

- From one to many SVMs. BICA vectors are not a univocal representation of the original data. Rather, we have different representations if we start from differently initialized hourglass networks. Moreover, given the randomness of the network parameters we use for initializations, we expect that different BICA vectors may enhance different features of the data to be clustered, so that some features are more appropriate for discriminating a certain subset of them, while other features serve the same purpose for other subsets. Hence on each BICA representation we train a SVM, considered a base learner. Then we suitably combine the classification proposed by each SVM. Namely, we compute for each example the frequency with which base learners answer 1, and we gather frequencies corresponding to either positive or negative samples. In the lose assumption that frequencies in each group follow a Gaussian distribution we locate a threshold at the cross of the c.d.f. of the right distribution with the complement to 1 of the c.d.f. of the left distribution [11], i.e.

\[
t = \frac{\hat{\mu}_-\hat{\sigma}_+ + \hat{\mu}_+\hat{\sigma}_-}{\hat{\sigma}_- + \hat{\sigma}_+}
\]

where \( \hat{\mu}_- \) and \( \hat{\sigma}_- \) are, respectively, the sample estimates of parameters \( \mu_- \) and \( \sigma_- \) of the negative distribution; idem for the positive distribution (see Fig 3). With this threshold we classify new records. We submit any record to each base learner of the ensemble and count the frequency with which it receives label 1. We classify the record positive if this frequency overcomes the threshold, negative otherwise. This is a variant of majority voting rule (corresponding to a threshold=0.5) that minimizes the misclassification risk still in the above Gaussianity hypotheses.
4 FSP: the Feature Selection Procedure

Dealing with a satisfactory classification, the true goal of this paper is to understand why. For instance, to assume our healthy-ill discrimination goal fully accomplished, we search for features of the gene expressions directly in the probe data, i.e. we want to isolate columns of these data, hence genes, that actually determined the classification. The feature selected by our proposed method on the one hand need to be validated for its plausibility, but on the other can be interpreted as working hypotheses for users.

As for the former, we we may point out the features that most frequently intervened in the computation of the BICA variables. To have a clear indication from frequencies, first of all we managed to reduce the number of boiling up variables through a pair of default procedures:

(1) still in light of remark 5, we reduce the length of the Boolean vectors by removing components multiplied by small coefficients in the hyperplane formulas until the error over probes does not get worse,

(2) we retrain the BICA neural networks in order to reproduce the code of every example with a small number of input nodes, completely abandoning the mirroring task and training the network in a supervised mode. Hence the network is deprived of connections from the hidden layer to part B of the output layer, thus assuming a funnel shape. Then we use standard pruning techniques to remove useless connections to compute the reduced binary code with the aim of completely disconnecting also some input nodes from the hidden layer.

We may apply these procedures to a selected number of BICA encodings. Namely, ordering the hyperplanes in a decrease ordering of their percentages of correct classification we select the former, which jointly correctly classify the
1. For a sufficient number of times do:

// data compression

1.1. Find a compressed representation of the data with Boolean independent components through the BICA neural network

1.2. Train a SVM to classify patterns using the compressed representation of the data obtained in step 1.

// A. Ensemble classification  |  // B. Feature selection

1.3.A Build a discriminating SVM.  |  1.3.1.B Remove nonessential variables from SVM

2.A Combine the SVM by the "biased" majority voting.  |  1.3.2.B Prune the BICA network and output the surviving input features,  

2.B Identify the "consensus" features.

Fig. 4. A sketch of the BICA-based procedure. Steps 1.1 and 1.2 are common to both ensemble classification and feature selection.

samples with a majority rule. Getting each time the set of features surviving the pruning procedure, we obtain a "consensus" set as the intersection of the sets.

The overall procedure for ensemble classification and feature/gene selection is summarized in Fig. 4.

5 Numerical Results

We toss our procedure on a significant number of benchmarks available in the literature [33,34].

As visualized in Fig. 5, the general scheme of experiments is the following: according to the multiple hold-out scheme, given a dataset, we randomly split it in two equally-sized training and test sets. We repeat this process 50 times. We train 50 hourglass networks for every partition, obtaining 50 BICA encodings, where the length of the encoded vectors is tentatively fixed to one hundredth of the number of the genes in the probes. On each encoding of a training set we train an SVM getting an ensemble of 50 SVMs decreeing the health status of each probe in the training set. Hence we pick up the most efficient subset of SVM capable of jointly classifying correctly the whole set of the training probes (three in both study cases below) and find the genes consensus set from them, as mentioned before 4.

4 In this way we avoid selection bias traps: we used only training data to select
We test the procedure at two levels. As the extraction of BICA components and their clustering through an ensemble of SVM is a crucial task *per se*, whose success highly affects the subsequent feature selection, we preliminary tossed these steps on two specific benchmarks. In particular, Table 1 reports their performance in terms of length of the compressed data and clustering accuracy on the test sets.

The benchmarks concern binary classification problems. Namely, the Ionosphere database consists of 351 patterns of 34 real variables each. 126 of them are labeled ood and the remaining ad.

With the Sonar database the goal is to discriminate sonar signals denoting a mine among 208 patterns, each made up of 60 continuous features between 0 and 1. The patterns are grouped in 13 random sets of 16 patterns each.

We compare the parameters obtained with our method with those deriving from

- the well known C4.5 method [Quinlan, 1993], where, a decision tree in terms of IF-THEN-ELSE rules is drawn directly by iterated partitioning of the genes without using any information about the test data [35].
sampled data on the basis of mutually exclusive tests on their range, and
• a multilayer perceptron trained with backpropagation method, as a template
of subsymbolic algorithms.

In a conservative way, we conventionally attribute a length of 4 bits to continu-
ous variables to account the lengths of the data to be compared with those
compressed by BICA. We assume indeed that these bits are sufficient to the
considered methods to discriminate the data w.r.t. the classification problem
they are questioned on. The entire procedure lasts a few seconds on....... From
the table we see.....

<table>
<thead>
<tr>
<th></th>
<th>BICA</th>
<th>Neural Network</th>
<th>C4.5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Length</td>
<td>Correct %</td>
<td>Length</td>
</tr>
<tr>
<td>µ</td>
<td>σ</td>
<td></td>
<td>µ</td>
</tr>
<tr>
<td>Iono</td>
<td>42</td>
<td>91.6 1.90</td>
<td>136</td>
</tr>
<tr>
<td>Sonar</td>
<td>50</td>
<td>81.2 3.7</td>
<td>240</td>
</tr>
</tbody>
</table>

Table 1
BICA experimental profile in comparison with other information management meth-
ods.

µ: mean value when σ is available, single trial value when σ is not available (n.a.)
σ: standard deviation where available. Correct%: percentage of correctly classified
patterns of the test set.

In order to toss the entire procedure, here below we discuss two paradigmatic
case studies in the field of microarray data processing, the former represent-
ing a relatively easy to classify benchmark, the latter a widely acknowledged
difficult instance. Namely, the first benchmark is the Colon adenocarcinoma
[33] data set composed of 2000 genes and 62 samples: 40 tumoral and 22 nor-
mal colon tissue samples. The compressed patterns are 20 bit long and allow
a classification accuracy as in Table 2 denoting a good performance of the
procedure. Actually in the literature we find classification scores that refer to
leave one out cross validation technique. Using this technique for comparison’s
sake, we obtain similar classification error rates around 10% [2]. In particu-
lar, Fig. 6 compares the error percentages on the instances of the multiple
hold-out scheme. They have been obtained on each record of the data set with
three different methods, namely Random Subspace Ensemble (RSE) [36] ex-
ploiting random projections to lower dimensional subspaces, Genet [37] wisely
adapting the training of a single SVM in a semi-automatic mode, and BICA
ensemble. The picture shows that there are records for which the error rate is
high for every method we consider. From a biological perspective, this may be
due to the fact that, as explained in [33], most normal samples are enriched
in muscle cells, while tumor samples are enriched in epithelial cells. Now the
above samples consistently misclassified by all methods present an “inverted”
tissue composition: normal samples are rich in epithelial cells, tumor samples
are rich in muscle cells. These considerations seem to confirm that the separa-
tion between normal and tumoral samples are made also on the basis of tissue
composition, as observed in [9].
Fig. 6. Frequencies $\phi$ of error of the patterns indexed with $n$ within the multiple hold-out scheme. Curve parameter: the classification procedure.

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Colon</th>
<th>Adenocarcinoma</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train $\mu$</td>
<td>0.98</td>
<td>0.865</td>
</tr>
<tr>
<td>Train $\sigma$</td>
<td>0.02</td>
<td>0.05</td>
</tr>
<tr>
<td>Test $\mu$</td>
<td>0.792</td>
<td>0.549</td>
</tr>
<tr>
<td>Test $\sigma$</td>
<td>0.09</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Table 2

Classification accuracy on cancer datasets. $\mu$: average, $\sigma$: standard deviation of the accuracy over training sets (Train) and test sets (Test) from the datasets heading the columns.

Concluding that our classifier is comparable to other methods, we searched for the features that are essentially responsible for the classification, as the main additional benefit. Namely, we selected the most efficient hyperplanes which, in numbers of 3, jointly correctly classify the entire training set. Then we intersected the corresponding three sets of input features (whose cardinality ranges from 107 to 127) that survived the pruning of the funnel nets implementing the mapping from the original features to the 20 Boolean variables in input to the hyperplanes. In this way we identified 49 genes as statistically relevant. Finally we performed an extensive search in biomedical databases in order to understand which of the statistically relevant genes selected by our methods are known to be relevant in the context of cancer diseases. We discovered that half of them may be associated with carcinogenic processes, according to bio-medical literature. Some examples are:

- T74906 CYTOCHROME P450 IIA7: Decreased expression of cytochrome P450 protein in patients with colonic adenoma [38];
- T94579 Human chitotriosidase precursor mRNA. Cancer cells overproduce chitinases to survive apoptosis. Selected in colon tumors also by [9].
- X02492 INTERFERON-INDUCED PROTEIN 6-16 PRECURSOR: is ex-
pressed in gastric cancers and inhibits mitochondrial-mediated apoptosis in gastric cancer cell line [39]

- L38503 Homo sapiens glutathione S-transferase theta 2: its activity is up-regulated in transitional cell carcinoma of urinary bladder [40]
- T76972 RNA-BINDING PROTEIN FUS/TLS: involved in t(12;16) in malignant liposarcoma (from GEO database).

The second case study concerns the adenocarcinoma data set [34] composed of 3801 genes previously filtered from 12600 according to Whitthead Institute protocol [41]. We work on 62 samples again, now partitioned into 31 adenocarcinoma good prognoses and 31 fatal prognoses. The genes are mapped in a Boolean space of dimension 40. With these binary variables we obtain a relatively satisfactory classification accuracy that does not sensibly improve if we work with a greater number of variables (see Table 2). The same percentage in the testing phase is definitely poor, just a bit higher than pure chance. Rather, we may argue that the small positive shift from 0.5 we obtain is due to dummy relations stated during the training between variables actually representing noise. Indeed, after applying reverse mapping with the funnel-net we obtain a subset of 48 genes that are statistically relevant but biological meaningless in general. Only 2 of them seem to be related to the fatal clinical outcome:

- neurotensin receptor 2: a prostate cancer cell receptor that plays a role in carcinogenic processes [42],
- ceruloplasmin (ferroxidase): ceruloplasmin levels used to predict recurrence of breast cancer [43]

These negative results confirm we may find relevant genes only if the data may be related with high reliability to the separation between healthy and diseased patients.

6 Conclusions

In this paper we propose a procedure that neither is automatic nor guarantees results. Nevertheless, it has the own value of integrating the use of both symbolical and subsymbolical tools for managing information to cluster data in a non trivial way. The key point is to isolate the information that is suitable for addressing the classification algorithm to correct solutions. In absence of a priori knowledge we code this information into binary vectors which we extract through and shrink with symbolic and connectionist methods. Then a reverse mapping identifies the original genes at the basis of the classification. To wisely compose this contrivance we assessed a theory at the basis of the BICA procedure for compressing data in a supervised mode; then we used
advanced tools from neural networks and SVM techniques to build a reliable classifier. Finally, the completion of the procedure and the check of its effectiveness undergoes thorough the end user analysis. As for case study, the correlation between genes and a specific disease is in most cases an open problem. We know from biomedical literature only a subset of the genes related to a specific disease. Yet in many cases we may have only some evidence, with a certain degree of uncertainty, of relationships of a given gene with a specific tumoral disease or with tumoral processes in general [44]. Nevertheless these correlations are crucial in seeking to discover effective cures for malignant diseases. Hence this case study represents a paradigmatic instance where we are pushed to use all our conceptual equipement to grasp any possible knowledge. As a result, we succeed in extracting a set of features that are semantically relevant for classifying gene expressions in DNA microarray data.

References


[38] I. Bergheim, C. Bode, A. Parlesak, Decreased expression of cytochrome P450 protein in non-malignant colonic tissue of patients with colonic adenoma., BMC Gastroenterol 5 (34).


