A wrapper approach to supervised input selection using simulated annealing

M. Filippone, F. Masulli, S. Rovetta
Dipartimento di Informatica e Scienze dell'Informazione,
Università di Genova, and CNISM, Via Dodecaneso 35, I-16146 Genova, Italy
email: {filippone, masulli, rovetta}@disi.unige.it

12th June 2006

Technical report DISI-TR-06-10

Abstract

Genomic data, and more generally biomedical data, are usually characterized by high dimensionality. A feature selection procedure can attain the two objectives of highlighting the relevant variables (genes) and improving classification results. In this paper we propose a wrapper approach to gene selection in classification of gene expression data using Simulated Annealing coupled with supervised classifiers. The proposed approach can perform global combinatorial searches through the space of all possible input subsets, can handle cases with numerical, categorical or mixed inputs, and is able to find (sub-)optimal subsets of inputs giving very low classification errors. The method has been tested on publicly available bioinformatics data sets, and also on mixed type data, using Support Vector Machines or Classification Trees. Moreover we propose some heuristics able to speed-up the convergence. The experimental results highlight the ability of the method to select minimal sets of relevant genes.

1 Introduction

Genomic data are often characterized by small cardinality and high dimensionality and can include some features that are not relevant for the discrimination among classes. This is the case, e.g., with gene expression data obtained from DNA microarrays where each dimension or feature corresponds to a gene expression data. Usually, some (or most) genes are not relevant for discrimination between classes and subsets of genes are mutually redundant. This is inherent in the experiment design: a lot of candidate genes are probed in a microarray experiment, and those related to the phenomenon under study are to be identified.

For those data, a gene selection procedure could highlight the relevant genes and improve the classification results at the same time.

Gene selection is a specific instance of a more general problem, which is called input or variable selection [15]. Note that our main goal is selection of relevant subsets of genes rather than performance optimization or dimensionality reduction. Therefore *features* (derivative variables obtained by transformation of raw input variables) are not of interest. Input selection algorithms can be broadly divided into two categories [5, 18]: *filters* and *wrappers*. Filters evaluate the relevance of each input (subset) using the data set alone, while wrappers invoke a learning algorithm to evaluate the quality of each input (subset). Both approaches, filters and wrappers, usually involve combinatorial searches (often only local) through the space of possible input subsets. Wrappers are usually more computationally demanding, but they can be superior in accuracy when compared with filters.

The strategy for variable selection, and the underlying assumptions about the input variables themselves, is also a design choice. Variables can be selected as a subset with aggregate discriminative power [29, 24], or ranked for their individual relevance [36, 14, 21]. In the latter case variables are assumed to be weakly correlated, so that their individual importance can be unambiguously assessed. In the former case, it is assumed that all possible interaction patterns can occur, and this forces a much more complex search space. However, ranks can also be used as an indication to evaluate a subset selection process, in an in-between approach.

The definition of relevance itself can be subject to variations [18], and the goal of the procedure can also be different, with some approaches aiming at comprehensive set (find all significant variables [16, 20, 24]) and others at explanatory sets (this is generally the case with all gene selection tasks, where one wants to identify the most important genes only, as, e.g., in [13]). Again, an in-between approach is possible [36] when an explicit cost function includes both performance and complexity (number of variables) terms. In this case, a continuum of possible balances is provided by the relative weighting of these terms.

To summarize, the gene selection problem is here stated as the problem of selecting small subsets of input variables achieving high discriminating power but with good explanatory properties.

These hypotheses form the basis of the method we are presenting in this paper, which is based on optimizing the combination of performance and complexity costs. We propose a wrapper approach to gene selection in classification of gene expression data. The combinatorial search is performed using the Simulated Annealing (SA) method [17] which is a global search method technique derived from Statistical Mechanics and is based on the Metropolis algorithm [22], while the learning algorithms employed in the paper are the Support Vector Machine [8] and the Decision Tree [6].

In the next section, we present the Simulated Annealing technique and describe how we applied it to input selection problem. In Sect. 3, some measures of the input relevance are illustrated. The

experimental validation of the proposed input selection method and some heuristics for its speeding-up are shown in Sect.s 4 and 5. Sect. 6 concerns the discussion and conclusions.

2 SA for gene selection

Table 1: Simulated Annealing Input Selection (SAIS) algorithm.

1. Initialize the parameters;
2. Initialize the binary mask **g** and the temperature T;
3. Train and test the classifier and evaluate the generalized system energy E;
4. **do**
5. Initialize $f = 0$ (number of iterations), h=0 (number of success);
   (a) **do**
   (b) Increment the number of iterations f;
   (c) Perturb the binary mask **g**;
   (d) Train and test the classifier and evaluate the generalized system energy E;
   (e) Generate a random number rnd in the interval [0,1];
   (f) **if** $rnd < P(\Delta E)$ **then**
      i. Accept the new binary mask **g**;
      ii. Increment the number of success h;
   (g) **endif**
   (h) **loop until** $h \leq h_{min}$ **and** $f \leq f_{max}$;
6. update $T = \alpha T$;
7. **loop until** $h > 0$;
8. end.

The method for input selection we propose makes use of Simulated Annealing (SA) technique [17] that is a global search method technique derived from Statistical Mechanics. SA is based on the Metropolis algorithm [22] proposed to simulate the behavior and small fluctuations of a system of atoms starting from an initial configuration, by the generation of a sequence of iterations. In the Metropolis algorithm each iteration is composed by a random perturbation of the actual configuration and the computation of the corresponding energy variation (ΔE). If $\Delta E < 0$ the transition is unconditionally accepted, otherwise the transition is accepted with probability given by the Boltzmann

Table 2: Parameters of SAIS algorithm and their values selected for the experiments presented in Sect.s 4 and 5.

Symbol	**Meaning**	**Exper A**	**Exper B**	**Exper C**
s_0	Number of inputs initially selected	20	20	5
p	Number of initializations of $\mathbf{g}$ for estimating the initial value of T	10000	10000	10000
$[w_{\min}, w_{\max}]$	Interval for w	$[1, s]$	$[1, s]$	$[1, s]$
$[v_{\min}, v_{\max}]$	Interval for v	$[1, s_0/2]$	$[1, s_0/2]$	$[1, d-s]$
λ	Regularization coefficient	10^{-2}	$2 \cdot 10^{-3}$	10^{-4}, 10^{-2}
f_{max}	Maximum number of iterations for each T	10000	2000	100
h_{min}	Minimum number of successes for each T	1000	200	30
α	Cooling parameter	0.9	0.9	0.9
γ	Aging constant	0.98	0.98	0.98

distribution:

$$P(\Delta E) = e^{-\Delta E/KT} \tag{1}$$

where K is the Boltzmann constant and T the temperature.

In SA this approach is generalized to the solution of general optimization problems [17, 25] by using an *ad hoc* selected cost function (*generalized energy*) instead of the physical energy. SA works as a probabilistic hill-climbing procedure searching for the global optimum of the cost function [28]. The temperature T takes the role of a control parameter of the search area (while K is usually set to 1), and is gradually lowered until no further improvements of the cost function are noticed. SA can work in very high-dimensional searches, given enough computational resources.

SA has been already applied to classification of gene expression data from DNA microarray [1] with the aim of training perceptrons. In this paper we apply SA to the input selection problem with the aim of aggregating an ideally minimal subset of inputs with strong discriminative power. The approach we adopted is to constrain the search space to subsets of variables, and to evaluate a compound cost function combining performance and complexity scores, as previously indicated. The method is described in the following, whereas in Tab. 1 a step-by-step outline of the proposed Simulated Annealing Input Selection (SAIS) algorithm is presented.

Let d be the dimensionality of the input space and $\mathbf{g} = (g_1, g_2, \ldots, g_d)$ be a binary mask representing the system state (configuration), where each bit g_i (with $i = 1, \ldots, d$) corresponds to either selection ($g_i = 1$) or deselection ($g_i = 0$) of an input. The number of bits of $\mathbf{g}$ set to 1 is denoted by s (i.e., $s \equiv |\mathbf{g}| = \sum_{i=1}^{d} g_i$) .

At Steps 1, 3, and 5, the classifier is trained in the sub-space of selected inputs as defined by the vector mask $\mathbf{g}$ and the *Classification Error* ε is evaluated using a cross-validation technique, e.g., *leave-one-out* or *k-fold validation* [31]. The generalized energy E is defined as a linear combination

of ε and of the number of selected inputs s:

$$E = \varepsilon + \lambda\, s \tag{2}$$

Note that the introduction of the number of selected inputs s in the computation of E penalizes situations in which the number of selected inputs is too high. The trade-off between size of input space and accuracy is controlled by the parameter λ (*penalization coefficient*). If $\lambda = 0$ we favor solutions with low classification error ε, without taking the dimensionality of the input space into account, while high values of λ can led to solutions with few input variables, but with a large classification error.

Moreover, due to the redundancy of groups of input variables [19] and to curse of dimensionality problem [4], often a good tuning of the penalization coefficient can allow us to find a very small input space supporting a small classification error.

The vector mask $\mathbf{g}$ (Step 2) is initialized by randomly setting s_0 bits to 1 and leaving the remaining $d - s_0$ to 0.

Moreover, as suggested in [25], at Step 2 the initial temperature T is obtained as the mean variation of generalized energy (ΔE) over an assigned number p of random initializations of $\mathbf{g}$.

A perturbation or move (Step 5c) is obtained in the following way: w bits of $\mathbf{g}$ set to 1 are switched to 0, and v bits of $\mathbf{g}$ set to 0 are switched to 1, where the values of w and v are extracted with uniform distributions, respectively in the (integer) intervals $[w_{\min}, w_{\max}]$ and $[v_{\min}, v_{\max}]$.

In Tab. 2, the list of the parameters to be initialized at Step 1 is presented together with the values selected for the experiments that we will describe in Sect. 4. Namely, s_0 and p are used at Step 2 for the initialization of $\mathbf{g}$ and T; $[w_{\min}, w_{\max}]$ and $[v_{\min}, v_{\max}]$ constrain the size of move (Step 5c); λ is used for computing the Generalization Energy E (Steps 3 and 5d); f_{max}, h_{min}, α are used for the SA scheduling (Steps 5 and 6); and γ is used for estimating the input relevances (see Sect. 3).

As already stated, the SAIS algorithm aims to find a small subset of variables, with high discriminant capability, by exploiting the redundancy of subsets of variables and penalizing solutions with high input dimensionality. To this aim s_0 should be selected of the order of the estimated input dimensionality, while intervals $[w_{\min}, w_{\max}]$ and $[v_{\min}, v_{\max}]$, regulating the variability of perturbation, can be changed during the algorithm to favor solutions with small input space.

SAIS is a computationally intensive algorithm, but it is able to work with both numerical and categorical inputs. It is also worth noting that, due to the random nature of SA, each time we run the SAIS algorithm we can find a new subset of s inputs from the original d.

3 Evaluating input relevance

As noticed in Sect. 1, many definitions of input relevance have been presented in the literature, with different meaning, motivation, and depending on the numerical or categorical type of inputs [18, 16, 20, 24, 13, 36]. Here we present some new definitions that apply both to numerical and categorical variables.

SA is an algorithm implementing a stochastic time-varying dynamical system where the state vector evolves in the direction of the minima of the generalized energy function. In our case during the evolution of the SAIS algorithm, the bits set in the state vector $\mathbf{g}$ will be related to the more relevant inputs with increasing probability.

The inputs which are more relevant for classification should appear soon in the set of bits of $\mathbf{g}$ set to 1 and will be as more frequent as the temperature decreases. In order to estimate the relevance of inputs, we can include in SAIS an aging algorithm. To this end, we can define a vector $\mathbf{r} = (r_1, r_2, \ldots, r_d)$. At Step 1 of the SAIS algorithm, we set $r_i = 0\ \forall i$. Every time a perturbation is accepted according to the Boltzmann distribution (Step 5.f), we update $\mathbf{r}$ as follows:

$$\mathbf{r} = \gamma\mathbf{r} + \mathbf{g} \tag{3}$$

where γ is the aging constant chosen in the interval [0,1].

At the end of the SAIS the vector $\mathbf{r}$ measures how long each input has been selected in the last few successful moves of the algorithm. We give to vector $\mathbf{r}$ an interpretation as vector of input relevances (*aged relevance*).

It is worth noting that measures based on similar aging algorithms are used also in other contexts such as implementation of policies in computer operating systems [32], sensor networks [12], and chaotic systems dynamics [3].

We also introduce the concepts of *voted relevance* and of *soft-voted relevance* defined as the sums of the input masks $\mathbf{g}$ and of vectors $\mathbf{r}$ obtained at the ends of an assigned number m of SAIS runs, i.e., as, respectively, $\sum_j \mathbf{g}^j$ and $\sum_j \mathbf{r}^j$ (with $j = 1, 2, ..., m$).

It is worth noting that none of those relevance notions are exploited for making decisions in the SAIS algorithm.

4 Experimental validation of SAIS

SAIS algorithm has been implemented in R [27] under the Linux operating system. The experimental validation consisted of three experiments using publicly available data sets with numerical inputs (Experiments A and B) and mixed numerical and categorical inputs (Experiment C) that will be described in the next subsections.

In the former two cases we used as the classification algorithm the *Support Vector Machine* (SVM) [35], that is one of most popular classifier, as implemented by Chung et al. [7]. We chose to work with linear kernels and with a cost parameter fixed to $C = 1$ in the SVM functional in order to avoid model selection on several parameters and to make it possible to obtain a clear biological interpretation of results. In the latter case we used the RPART decision tree algorithm as implemented in R by Ripley [6][26].

Tab. 3 lists the principal characteristics of data sets, and the learning machine used in each experiment. The selected values for the initialization of parameters (Step 1 of SAIS) are shown in Tab. 2.

In all the experiments described in this section, the w bits of $\mathbf{g}$ set to 1 to be switched to 0 and the v bits of $\mathbf{g}$ set to 0 to be switched to 1, at the move step of SAIS (Step5c), are selected using a uniform distribution in the (integer) interval $[1, s]$, i.e., with probability

$$p_i = \frac{1}{s} \quad \textit{(Uniform Selection)}, \tag{4}$$

and, similarly, the v bits of $\mathbf{g}$ set to 0 to be switched to 1, are selected using an uniform distribution in the interval $[1, d - s]$, i.e., with probability

Table 3: Characteristics of data sets and of the learning machines used in the experiments.

	Exper A	**Exper B**	**Exper C**
Number of numerical inputs	7129	2000	6
Number of categorical inputs	0	0	7
Number of instances	38	62	303
Learning machine	Linear SVM	Linear SVM	RPART

$$p_i = \frac{1}{d-s} \quad \textit{(Uniform Selection).} \tag{5}$$

In Sect. 5 we will show some modifications to this basic approach to system state perturbation, aimed at accelerating the convergence of SAIS.

4.1 Experiment A

The first data set we considered is *Leukemia* data by Golub et al. [13][1]. The Leukemia problem consists in characterizing two forms of acute leukemia, Acute Lymphoblastic Leukemia (ALL) and Acute Mieloid Leukemia (AML). The original work proposed both a supervised classification task (*class prediction*) and an unsupervised characterization task (*class discovery*). The data set contains 38 instances for which the expression level of 7129 genes has been measured with the DNA microarray technique (the interesting human genes are 6817, and the others are controls required by the technique). Of these instances, 27 are cases of ALL and 11 are cases of AML. Moreover, it is known that the ALL class is composed of two different diseases, since they are originated from different cell lineages (either T-lineage or B-lineage). In the data set, ALL cases are the first 27 objects and AML cases are the last 11. Therefore, in the presented results, the object identifier can also indicate the class (ALL if id ≤ 27, AML if larger). Using those data (with dimensionality $d = 7129$) Golub et al. [13] selected a set of 50 most relevant genes.

Given the scarcity of instances in the sample, for the computation of the Classification Error ε we applied the SAIS algorithm using a leave-one-out resampling technique.

In Tab. 4 we show the results of 10 independent runs of SAIS using the assumptions in Tab. 2 and SVM with linear kernel, as already noticed. For each run we started SAIS with a mask $\mathbf{g}$ with 20 bits set to 1 and ended with a set of two genes (ranked in Tab. 4 in order of aged relevance) with perfect discriminant ability (i.e., $\varepsilon = 0$). Those results have been obtained with a penalization parameter $\lambda = 10^{-2}$. In the next experiments (B and C) we will be not able to find a value of λ with those characteristics, but we will have to select λ in order to handle the trade-off between accuracy and input space size.

In Fig. 1(a1) and Fig. 1(a2) we plot the behavior of the classification error ε and the number of selected genes versus the iteration number of the algorithm in a run of SAIS. Each iteration corresponds to a different value of temperature T (i.e. Step 5 and Step 6 in Tab. 1). Those graphs

[1] http://www.broad.mit.edu/cancer/software/genepattern/datasets/.

Table 4: Genes selected in each run of the SAIS algorithm on the Leukemia data set. ε is the classification error and m is the number of misclassified instances.

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**
ε; m	0; 0	0; 0	0; 0	0; 0	0; 0
Gene 1	M55150_at	U50136_rna1_at	M23197_at	X95735_at	Y12670_at
Gene 2	HG3523-HT4899_s_at	M58603_at	X85116_rna1_s_at	M10321_s_at	U29607_at
	Run 6	**Run 7**	**Run 8**	**Run 9**	**Run 10**
ε; m	0; 0	0; 0	0; 0	0; 0	0; 0
Gene 1	M63138_at	X95735_at	M23197_at	X95735_at	M55150_at
Gene 2	U29091_at	D14659_at	M24470_at	HG3454-HT3647_at	HG3523-HT4899_s_at

Table 5: Genes selected in each run of the SAIS algorithm on the Colon data set. For each run we show also ε (classification error) and m (number of misclassified instances).

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	**Run 7**	**Run 8**	**Run 9**	**Run 10**
ε; m	0.048; 3	0.048; 3	0.048; 3	0.081; 5	0.048; 3	0.032; 2	0.016; 1	0.065; 4	0.081; 5	0.016; 1
Gene 1	J02854	R36977	Z50753	X12369	Z50753	R87126	H08393	H64489	X05276	H49870
Gene 2	M24069	M95627	X66365	D21261	L06175	R88749	M84490	U37012	L37112	M28219
Gene 3	R49719	U05875	H20505	T47601	T69748	M11220	L06175	H65355	X13810	X66503
Gene 4	T74896	T40637	T51493	T56690	L22214	R27813	M23254	J03824	T74257	R54097
Gene 5	X79683	H67764	R54818	L10413	R74208	M95678	L09159	–	R49565	T74257
Gene 6	M22488	R38636	X56597	U10117	–	H49515	T81492	–	M23115	M13450
Gene 7	H81802	R90908	H05966	H02630	–	T86444	H06061	–	U33849	L06111
Gene 8	R01755	–	U31525	M38690	–	M63239	D26129	–	M76378	D14663
Gene 9	H13292	–	M83664	D12686	–	U30498	U27699	–	R55778	R07007
Gene 10	K03474	–	U33849	L14076	–	H05814	T74257	–	–	T89164
Gene 11	R49565	–	M94250	–	–	T54341	X78817	–	–	R36977
Gene 12	H07899	–	–	–	–	–	–	–	–	U07802
Gene 13	T95063	–	–	–	–	–	–	–	–	M23419
Gene 14	X77548	–	–	–	–	–	–	–	–	T89175
Gene 15	U09646	–	–	–	–	–	–	–	–	T85165
Gene 16	T67433	–	–	–	–	–	–	–	–	T78489
Gene 17	–	–	–	–	–	–	–	–	–	H09137
Gene 18	–	–	–	–	–	–	–	–	–	U37012
Gene 19	–	–	–	–	–	–	–	–	–	X73424
Gene 20	–	–	–	–	–	–	–	–	–	R37276
Gene 21	–	–	–	–	–	–	–	–	–	H56077

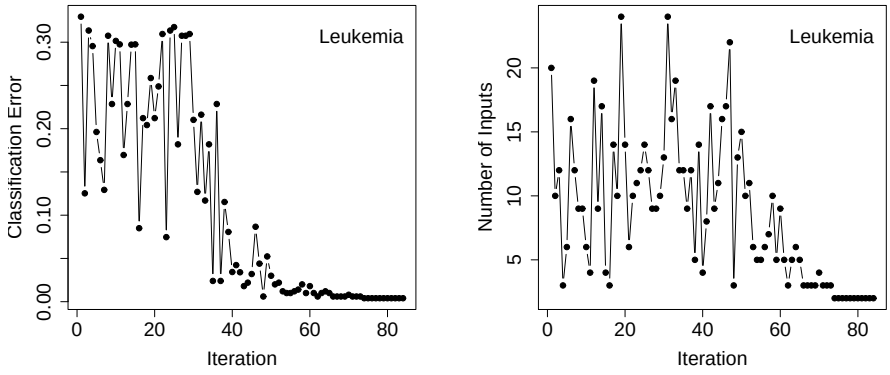
illustrate the ability of SAIS to minimize both the Classification Error ε and the number of relevant inputs s.

We notice that each selected pair of genes (Tab. 4) contains at least one gene found by Golub et al. [13]. Moreover, ranking the genes following their soft-voted relevance degrees, we found in the first three positions X95735_at, M23197_at and M55150_at, that are also in the set selected by Golub et al. [13]. The the most relevant gene (i.e., X95735_at) has been also highlighted by Guyon et al. [14].

4.2 Experiment B

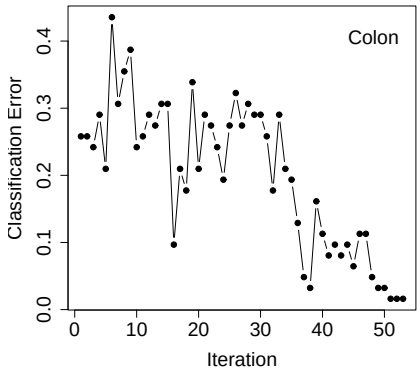
The second data set on which we performed input selection is the *Colon* data set by Alon et al. [2]. This is an oligonucleotide microarray analysis of gene expression in 40 tumor and 22 normal colon tissue instances, used to characterize the role and behavior of more than 6500 human genes in colon adenocarcinoma. The normal instances were obtained from a subset of the tumor instances, so that they are well paired to the corresponding positive instances. The actual data set used in the experiments[2], contains only the 2000 genes most clearly expressed in the experiments, those with the

[2]http://microarray.princeton.edu/oncology/affydata/index.html.

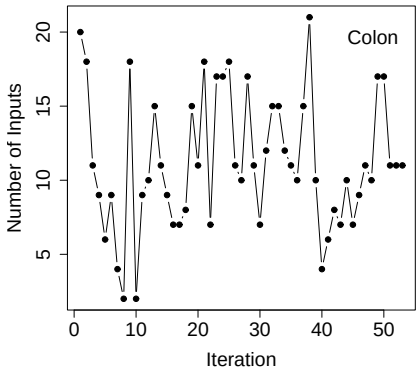


(a1)

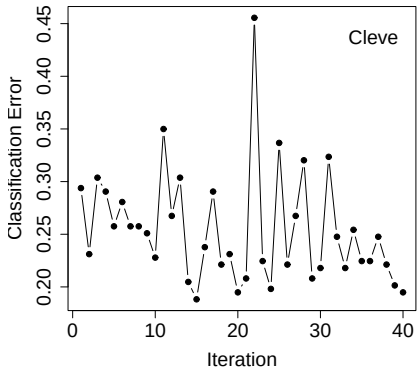
(a2)



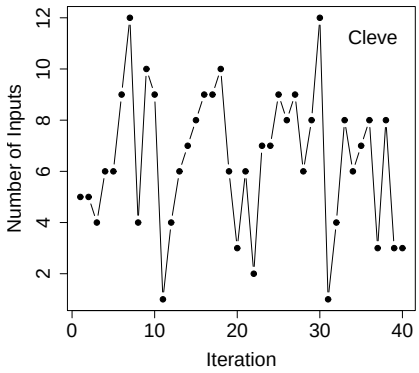
(b1)



(b2)



(c1)



(c2)

Figure 1: Classification error ε and Number of selected inputs s versus the iteration number for Run 4 of Experiment A on the Leukemia data set ((a1), (a2)), Run 7 of Experiment B on the Colon data set ((b1), (b2)), and Run 1 of Experiment C on the Cleve data set ((c1), (c2)).

Table 6: Inputs selected in each run of the SAIS algorithm on Cleve data set using as penalization parameter $\lambda = 10^{-2}$ in the generalized energy. For each run we show also ε (classification error) and m (number of misclassified instances).

	Run 1	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	**Run 7**	**Run 8**	**Run 9**	**Run 10**
ε; m	0.18; 55	0.18; 55	0.18; 54	0.18; 55	0.18; 55	0.15; 45	0.18; 55	0.18; 54	0.18; 55	0.18; 55
Input 1	ncol	ncol	ncol	ncol	ncol	ncol	ncol	ncol	ncol	thal
Input 2	thal	chest	ches	thal	thal	thal	thal	chest	chest	ncol
Input 3	chest	thal	induced	chest	chest	induced	chest	induced	thal	chest
Input 4	–	–	oldpeak	–	–	Sex	–	oldpeak	–	–
Input 5	–	–	–	–	–	Age	–	–	–	–
Input 6	–	–	–	–	–	bps	–	–	–	–

Table 7: Inputs selected in each run of the SAIS algorithm on Cleve data set using as penalization parameter $\lambda = 10^{-4}$ in the generalized energy. For each run we show also ε (classification error) and m (number of misclassified instances).

	Run 11	**Run 12**	**Run 13**	**Run 14**	**Run 15**	**Run 16**	**Run 17**	**Run 18**	**Run 19**	**Run 20**
ε; m	0.16; 48	0.15; 44	0.14; 42	0.17; 50	0.16; 49	0.14; 42	0.15; 44	0.15; 45	0.14; 42	0.15; 44
Input 1	ncol	ncol	ncol	ncol	ncol	ncol	ncol	ncol	ncol	ncol
Input 2	induced	thal	Age	induced	induced	induced	thal	thal	induced	thal
Input 3	Sex	Sex	induced	Sex	thal	Sex	induced	induced	thal	induced
Input 4	bps	induced	thal	bps	ecg	thal	Sex	ecg	Sex	Sex
Input 5	thal	Age	Sex	slope	Sex	Age	ecg	fbs	Age	ecg
Input 6	Age	bps	bps	–	oldpeak	bps	Age	Sex	bps	bps
Input 7	chol	ecg	–	–	Age	–	bps	bps	–	Age
Input 8	–	–	–	–	bps	–	–	Age	–	–

highest minimal intensity across the 62 tissue instances.

In all runs of SAIS on this data set, the classification error ε has been evaluated using 6-fold cross-validation, as in this case we have a larger (although still small in absolute terms) data base than in previous experiment.

In Tab. 5 we show the results of 10 independent runs of SAIS using the assumptions in Tab. 2 and SVM with linear kernel, as already noticed. For each run we obtained a different small set ranging from 4 to 21 genes with good discriminant ability, that in Tab. 5 are presented ranked following their aged relevance degree.

In Fig. 1(b1) and Fig. 1(b2) the behavior of the classification error ε and the number of selected genes are plotted versus the iteration number of the algorithm.

For Colon data set the selection of a penalization parameter $\lambda = 10^{-3}$ allows us to obtain a small classification error ε. Greater values of λ lead to the selection of a smaller input space at the cost of an higher ε.

Run 7 selected 11 genes and made only one misclassification on the data base. Moreover, the gene H08393, obtained in this run, has been also highlighted by Guyon et al. [14].

4.3 Experiment C

The third experiment has been performed on the *Cleve* mixed data set which is modified from the Detrano's heart disease data set which contains 76 clinic attributes for each patient. Cleve data set is composed by 303 instances (patients) with six categorical and eight numerical inputs[3]. We focus

[3]http://www.ics.uci.edu/ mlearn/databases/heart-disease/cleve.mod.

Table 8: Classification error ε obtained by several algorithms on Cleve data set.

Model	**mean**	**standard deviation**
C5.0 [11]	.272	.005
MoE I [33]	.232	.039
MLP [30]	.209	.037
MoE II [33]	.204	.029
C5.0+Boost [11]	.202	.005
Rprop [30]	.182	.040
SAIS $\lambda = 10^{-2}$	**.178**	**.010**
ADTree [11]	.170	.006
5-NN OS [34]	.170	.010
Stump Boost [11]	.166	.008
MoE [30]	.161	.036
SAIS $\lambda = 10^{-4}$	**.149**	**.010**

on the problem of finding a set of features able to distinguish only between health and disease.

We chose this data base, even if it is not a Bioinformatics data set, as it has been well studied in the literature and can allow us to make a deeply comparison on a mixed data set of the performances of SAIS with those of previously studied models.

In all the runs of SAIS on the Cleve data set, the classification error ε has been evaluated using 10-fold cross-validation, as even in this case the size of the data base is small. In Tab.s 6 and 7 we show the results of 20 independent runs of SAIS using the assumptions in Tab. 2 and a RPART classification tree [6, 26] as a classifier. The classification tree has been employed here as it is able to handle both numerical and categorical inputs.

In the runs of SAIS shown in Tab. 6 we chose a penalization coefficient $\lambda = 10^{-2}$, while in those of Tab. 7 we used $\lambda = 10^{-4}$. In both tables the inputs selected are ranked following the valuation of their aged relevance.

Using the value of the penalization coefficient $\lambda = 10^{-2}$, we obtained solutions with higher classification error ε and few selected inputs. With $\lambda = 10^{-4}$, we obtained solutions with better accuracy and higher dimension of the selected input space. In the former case the first inputs, ranked using both soft voted and voted relevances, were: *ncol* (number of vessels colored), *thal* (norm, fixed, rever), *chest* (chest pain type, namely: angina, abnang, notang, asympt), and *induced* (exercise induced angina). In the latter case the first inputs, ranked using voted relevance, were: *sex*, *induced*, *ncol*, and *ecg* (resting ecg, namely: norm, abn, hyper); while, ranking according to soft voted relevance, the first inputs were: *ncol*, *chest*, *induced*, and *thal*.

In Fig. 1(c1) and Fig. 1(c2) the behavior of the classification error ε and the number of selected inputs are plotted versus the iteration number of the algorithm in run number 11.

In Tab. 8 we compare the accuracies obtained with SAIS with those of other models reported in the literature, namely C5.0 [11], online Mixture of Expert classifier (MoE I) [33], Multi-Layer Perceptron using a softmax activation function (MLP) [30], Mixture of Expert with softmax self-organizing gate function (MoE II) [33], C5.0 with boosting [11], perceptron trained using Resilient Backpropagation (RProp) [30], Alternating Decision Tree boosting algorithm (ADTree) [11], 5-NN with Optimal Scoring (5-NN OS) [34], boosting algorithm generating trees with a single layer of

Table 9: Time duration in hours of input selection methods on Leukemia database, averaged on 10 runs (with the exception of RFE). Results obtained on a Pentium IV 1900 MHz personal computer.

Method	**mean**	**standard deviation**
RFE [14]	2.3	–
SAIF US	14.2	4.0
SAIF SPS ($\varphi = 0$)	10.9	3.4
SAIF SPS ($\varphi = 0.25$)	13.4	3.8
SAIF SPS ($\varphi = 0.50$)	13.0	6.6
SAIF SRS ($\alpha = 2$)	9.1	2.2
SAIF SRS ($\alpha = 3$)	5.3	1.6
SAIF SRS ($\alpha = 4$)	5.7	1.8
SAIF STS ($c = 2$)	6.0	3.5
SAIF STS ($c = 5$)	4.6	1.1
SAIF STS ($c = 8$)	4.1	1.2

decision nodes (StumpBoost) [11], and Mixtures of local Experts (MoE) [30]. The shown results demonstrate that the proposed input selection procedure allows a simple classification tree to perform better than most ensemble methods.

5 Speeding-up SAIS

Optimization techniques are evaluated both on the precision in finding the solutions to the problem, and on their converging speed. Concerning SAIS algorithm, it shows a time cost sometime comparable with other wrapper methods, even if Simulated Annealing is computationally intensive.

Let us consider, e.g., the method by Guyon et al. [14] which performs feature selection by Recursive Feature Elimination (RFE). At each iteration a new linear SVM is trained and the inputs with the smallest weight is eliminated. In order to rank the entire set of thousand of features it must run an SVM for each dimension of the input space starting from thousands to one. In SAIS, instead, when we work on only numerical inputs and using SVMs, we have much more (hundreds of thousand) SVMs to train, but each learning procedure is fast, as it must perform classification in a space of small dimension (only some tens of inputs).

In Tab. 9 we present a comparison of convergence times obtained using RFE [14], SAIS with Uniform Selection (US), already illustrated in the previous sections, and some heuristics we have implemented to speed-up SAIS and we shall show in this section. The results are obtained on Leukemia data by Golub et al. [13] on a Pentium IV 1900 MHz personal computer and averaged on 10 runs of each algorithm.

The heuristics we propose apply to the case of numerical inputs and learning methods based on linear discriminant $f = \mathbf{w} \cdot \mathbf{x}$ (where $\mathbf{x}$ is the vector of inputs and $\mathbf{w}$ is the vector of parameters or *weights*), such as linear SVM.

In this case the sensitiveness of the discriminant function to the input x_i, can be evaluated as:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = \frac{\partial (w_i x_i)}{\partial x_i} = w_i \quad (\textit{Absolute input sensitiveness}), \tag{6}$$

or, better, as:

$$\frac{w_i^2}{\sum_j w_j^2} \quad (Normalized\ input\ sensitiveness). \tag{7}$$

Inputs sensitiveness is often exploited by wrapper methods for feature selection as notion of input relevance [36, 14, 21]. As noticed in Sect. 1, this implies that variables are assumed to be weakly correlated, so that their individual importance can be unambiguously assessed, and no input redundancy of subsets of features is taken into account.

SAIS implements a combinatorial search on the all possible input subspaces on the basis of the penalized classification error that is an evaluation more robust both to noise data and on numerical approximations than those of input sensitiveness. However, in case of numerical inputs and linear discriminant classifiers, we can embedded input sensitiveness evaluations in the algorithm, as a broad knowledge on the solution of the problem.

In particular, advice from input sensitiveness can be useful in the perturbation step of SAIS (Step 5c). For example, we can make use of input normalized sensitiveness for selecting the w bits of $\mathbf{g}$ in the interval $[1, s]$ to be switched for 1 to 0 in the Step 5c of SAIS, instead of using Uniform Selection as in Sect. 4. We have implemented the following selection techniques borrowed from Genetic Algorithms literature (see, e.g., [23]):

1. Sensitiveness Proportionate Selection (SPS). A bit of mask g set to 1 is select to be switched to 0 with probability

$$p_i = \frac{\varphi}{s} + \frac{1-\varphi}{s-1}\left(1 - \frac{w_i^2}{\sum_{j=1}^{s} w_j^2}\right), \quad \text{with} \ \ 0 \leq \varphi \leq 1. \tag{8}$$

2. Sensitiveness Ranked Selection (SRS). A vector of ranks of inputs $\rho = (\rho_1, \rho_2, ..., \rho_d)$ is obtained by sorting on the ground of their normalized relevance. Then each bit of mask $\mathbf{g}$ set to 1 is select to be switched to 0 with probability

$$p_i = \frac{\alpha^{\rho_i}}{\sum_{j=1}^{s} \alpha^{\rho_j}}, \quad \text{with} \ \ \alpha \geq 0. \tag{9}$$

3. Sensitiveness Tournament Selection (STS). To select a bit of mask $\mathbf{g}$ set to 1 to be switched to 0, c bits (competitors) are sampled from those set to 1 using an uniform probability distribution. Then, only the bit corresponding to the input with the lowest normalized sensitiveness is selected and switched to 0.

In SPS if $\varphi = 0$ we have the basic perturbation with Uniform Selection used in Sect. 4, while if $\varphi = 0$ we have the pure roulette-wheel algorithm used in Genetic Algorithms. When $0 < \varphi < 1$, the selection is done in an intermediate way. Nevertheless, as shown in Tab. 9, the best speeding-up has been obtained for the pure roulette-wheel algorithm (i.e., $\varphi = 0$).

We can notice that using SPS, the values of normalized input sensitiveness, after few iterations tend to become very similar for all inputs, independently on the value assigned to φ. In Genetic

Algorithms literature, this phenomenon is called the problem of *stagnation*, and is known to slow-down convergence, as best solutions are favored only slightly with respect to the worst ones. Remedies to stagnation proposed in the literature are ranked and tournament selections.

Using STS, stagnation is avoided at the cost of a reordering overhead, as it happens in Genetic Algorithms. From Tab. 9, we notice a significant speeding-up with respect SPS, especially with $\alpha = 3$ or 4. Note that when $\alpha = 1$, STS becomes the the basic perturbation with Uniform Selection used in Sect. 4.

STR can be thought of as a noisy version of rank selection. Even in this case we can can avoid stagnation, but no global reordering is required. STR permits to obtain the fastest runs of SAIS, as shown in Tab. 9, especially with a large number of competitors (c).

We point out that when we apply those heuristics to Step 5c of SAIS, we make use of a generic knowledge on the solutions of the combinatorial search problem. If the advice is correct, we obtain a speeding-up of SAIS, otherwise the move will be rejected at Step 5f. However, the results obtained with those heuristics are comparable with those obtained with SAIS US and illustrated in Table 4.

6 Discussion and Conclusions

This paper describes SAIS, a variable subset selection method of the wrapper type, with application to the problem of gene selection. The method is based on Simulated Annealing [17] and therefore it implements a global search strategy. The output of the SAIS algorithm is a relevant subset of genes, where relevance is experimentally assessed by using a supervised classifier during the evaluation phase of the algorithm [8][6] and in addition it provides estimates of individual relevances of genes within the output subset.

The obtained results compare very favorably with the literature. On the 7129-dimensional Leukemia data set by Golub et al. [13] the proposed input selection method is able to find for each run a subset of two genes, that is sufficient to achieve null classification error with the leave-one-out procedure. Different pairs of genes were obtained from different runs, and the most frequent ones are also in the list found by Golub.

On the 2000-dimensional Colon data set by Alon et al. [2] the proposed input selection method finds solutions with a low number of selected genes and a very good discriminant capability.

Even on the Cleve mixed data set, the proposed input selection method gives good accuracy in classification, performing better than many ensemble of classifiers.

These results can be obtained semi-automatically with few parameter selection iterations, and the method performs classification and input selection and ranking jointly.

Due to the type of problem addressed, which implies long data acquisition and preparation times, the relatively long run times required by a SA procedure are not considered a limiting factor for the method. Of course, this does not apply to optimal solutions (which are found in theoretically unbounded time), but to good suboptimal solutions. In this realistic case, the optimization can be stopped according to one of the usual criteria (threshold on the objective function, number of iterations, or others).

Let's consider, e.g., the method by Guyon et al. [14] which performs feature selection by recursive feature elimination. At each iteration a new linear SVM is trained and the inputs with the smallest weight is eliminated. In order to rank the entire set of thousand of features it must run an SVM for

each dimension of the input space starting from thousands to one. In SAIS, instead, we have many more classifiers to train. However, each one performs classification in a space of small dimension (some tenths of inputs). Moreover, the convergence SAIS can be accelerated using the heuristics inspired from the Genetic Algorithms we have presented.

Moreover, it is worth noting that the proposed algorithm can work either with numerical or categorical inputs depending on the associated learning machine and on the application. For example in [10] we run SAIS method associated to Fuzzy c-means on the Leukemia data set by Golub et al. [13] obtaining minimal sets of inputs able to perform unsupervised clustering with null representation error.

The stochastic nature of the SA method implies that at each run a different subset of genes can be found, provided that it complies with some quality requirement, in our case implemented by the classification performance of a suitable classifier. Having more than one gene sets at the output, the method can possibly find other applications, e.g. in systems biology, where interaction networks and cause-effect relationships are investigated.

References

[1] A.A. Albrecht, S.A. Vinterbo, and L. Ohno-Machado, An epicurean learning approach to gene-expression data classification. *Artificial Intelligence in Medicine*, vol.28, no.1, pp. 75–87, 2003.

[2] U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine, Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays, *Proc. Natl. Acad. Sci.*, USA, vol.96 no. 12 pp. 6745–6750, 1999.

[3] E. Barkai, Aging in Subdiffusion Generated by a Deterministic Dynamical System, *Phys. Rev. Lett.* vol. 90, 104101, 2003.

[4] R.. Bellman, *Adaptive Control Processes: A Guided Tour*, Princeton University Pressv, 1961.

[5] A. Blum and P. Langley, Selection of Relevant Features and Examples in Machine Learning, *Artificial Intelligence*, vol. 97, nos. 1–2, pp. 245–271, 1997.

[6] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth & Brooks, Pacific Grove, CA., 1984.

[7] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001; Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm.

[8] C. Cortes and V. Vapnik, Support vector networks, *Machine Learning*, vol. 20 pp. 273 – 297, 1995.

[9] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.

[10] M. Filippone, F. Masulli, and S. Rovetta, Unsupervised gene selection and clustering using simulated annealing, In I. Bloch, A. Petrosino, and A. Tettamanzi, ed.s, *WILF*, volume 3849 of *Lecture Notes in Computer Science*, pp. 229–235, Springer, 2005.

[11] Y. Freund and L. Mason, The alternating decision tree learning algorithm, *Proc. 16th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA, pp. 124–133, 1999.

[12] D. Ganesan, B. Greenstein, D Perelyubskiy, D. Estrin and John Heidemann, An Evaluation of Multi-resolution Storage for Sensor Networks, *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, pp. 89–102,ACM, 2003.

[13] T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller,M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, E Lander, Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring, *Science* vol. 286, pp. 531–537, 1999.

[14] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, Gene selection for cancer classification using support vector machines, *Machine Learning*, vol. 46 no.s 1–3, pp. 389–422, 2002.

[15] I. Guyon, A. Elisseeff, *An introduction to variable and feature selection*, *Journal of Machine Learning Research*, vol. 3 pp. 1157–1182, 2003.

[16] K. Kira and L. Rendell, The Feature Selection Problem: Traditional Methods and a New Algorithm, Proc. 10th Nat l Conf. Artificial Intelligence (AAAI-92), pp. 129–134, 1992.

[17] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, vol. 220, pp.661–680, 1983.

[18] R. Kohavi and G. John, Wrappers for Feature Subset Selection, *Artificial Intelligence*, vol. 97, nos. 1–2, pp. 273–324, 1997.

[19] D. Koller and M. Sahami, Toward Optimal Feature Selection, *Proceedings of the Thirteenth International Conference (ICML '96)*, Lorenza Saitta, Ed., pp.284–292, Morgan Kaufmann, 1996.

[20] I. Kononenko, *Estimating Attributes: Analysis and Extensions of RELIEF*, Proc. Seventh European Conf. Machine Learning, pp. 171–182, 1994.

[21] F. Masulli and S. Rovetta, Random Voronoi ensembles for gene selection, *Neurocomputing*, vol. 55, no.s 3-4, pp. 721-726, 2003.

[22] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculations for fast computing machines. *Journal of Chemical Physics*, vol. 21, pp. 1087–1092, 1953.

[23] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin Heidelberg New York, 3. edition, 1998.

[24] C. Moneta, G.C. Parodi, S. Rovetta, and R. Zunino, Automated diagnosis and disease characterization using neural network analysis, in *Proceedings of the 1992 IEEE International Conference on Systems, Man and Cybernetics - Chicago, IL, USA*, pp. 123-128, 1992.

[25] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, *Numerical Recipes in C*, (2nd Edition), Cambridge University Press, 1992.

[26] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge university press, Cambridge, 1996.

[27] R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005. ISBN 3-900051-07-0. http://www.R-project.org

[28] F. Romeo and A. Sangiovanni-Vincentelli. *Probabilistic Hill-Climbing Algorithms: Properties and Applications*. Computer Science Press, Chapell Hill, NC, 1985.

[29] N. Slonim and N. Tishby, Agglomerative information bottleneck, in *Advances in Neural Information Processing Systems*, pages 617-623, 2000.

[30] H.Stern, Improving on the mixture of experts algorithm, Internet publication *http://flame.cs.dal.ca/ stern/papers/stern-improving-on-the-mixture-of-experts-algorithm-2003.pdf.*

[31] M. Stone, Cross-validatory choice and assessment of statistical predictions, *Journal of the Royal Statistical Society*, B, vol. 36, no. 1, pp.111–147, 1974.

[32] A. Tanenbaum, *Modern Operating Systems*, (2nd Edition), Prentice Hall, 2001.

[33] B. Tang, M.I. Heywood, and M. Shepherd, Input partitioning to mixture of experts, *In 2002 International Joint Conference on Neural Networks, Honolulu, Hawaii*, pp. 227–232, 2002.

[34] E. Tuv and G.C. Runger, Scoring levels of categorical variables with heterogeneous data, *IEEE Intelligent Systems*, vol. 19, no. 2, pp.14–19, 2004.

[35] V. N. Vapnik, *The nature of statistical learning theory*, Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[36] J.Weston, A. Elisseff, B. Schoelkopf, and M. Tipping, Use of the zero norm with linear models and kernel methods, *Journal of Machine Learning Research*, vol. 3, pp. 1439-1461, 2003.