Noname manuscript No.
(will be inserted by the editor)

Simulated Annealing for Supervised Gene Selection

M. Filippone · F. Masulli · S. Rovetta

Received: date / Accepted: date

Abstract Genomic data, and more generally biomedical data, are often characterized by high dimensionality. An input selection procedure can attain the two objectives of highlighting the relevant variables (genes) and possibly improving classification results. In this paper, we propose a wrapper approach to gene selection in classification of gene expression data using Simulated Annealing along with supervised classification. 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 low classification errors. The method has been tested on publicly available bioinformatics data sets using Support Vector Machines, and on a mixed type data set using Classification Trees. We also propose some heuristics able to speed up the convergence. The experimental results highlight the ability of the method to select minimal sets of relevant features.

Keywords Input Selection · DNA Microarrays · Gene Selection · Support Vector Machines · Classification Trees · Simulated Annealing

PACS 07.05.Kf · 02.60.Pn · 02.50.Sk

Mathematics Subject Classification (2000) 62-07 · 62-04 · 62H30

M. Filippone
Department of Computer Science, University of Sheffield, United Kingdom
E-mail: m.filippone@dcs.shef.ac.uk

F. Masulli
Department of Computer and Information Sciences, University of Genova, Italy;
CNISM Genova Research Unit, Genova, Italy;
Sbarro Institute for Cancer Research and Molecular Medicine, Center for Biotechnology,
Temple University, Philadelphia, PA, USA
E-mail: masulli@disi.unige.it

S. Rovetta
Department of Computer and Information Sciences, University of Genova, Italy;
CNISM Genova Research Unit, Genova, Italy
E-mail: rovetta@disi.unige.it

1 Introduction

Genomic data are often characterized by small cardinality and high dimensionality, and can include some inputs that are not relevant for class discrimination. This is the case, e.g., of gene expression data obtained from DNA microarrays where each dimension or input corresponds to a gene expression data. Usually, some (or most) genes are not relevant to discriminate among classes and subsets of genes are mutually redundant. This is inherent in the experiment design: several candidate genes are probed in a microarray experiment, and those related to the phenomenon under study must 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 one of the main problems in pattern recognition, which is very general and variously termed input, attribute, or variable selection [18]. This problem concerns the search for small relevant subsets of inputs in high dimensional data sets.

Input selection procedures [18] can improve the knowledge on the problem, e.g., finding the most relevant genes expresses in relation to a pathology in DNA microarrays data sets, or finding the most relevant keywords from sets of text documents. Therefore, we are not interested in methods for *feature* extraction, as they can reduce the dimensionality by transforming input variables into a smaller set of derivative variables, thus not allowing to have a direct interpretation of the role played by the raw inputs.

An useful by-product of input selection procedures is also the possibility of alleviating the *curse of dimensionality* problem [8] arising in the usual situation of data sets characterized by high dimensionality and low cardinality. In fact, after input selection, learning machines can usually improve their generalization ability.

Input selection algorithms can be broadly divided into two categories [9,25]: *filters* and *wrappers*. Filters evaluate the relevance of input subsets using the data set alone, while wrappers invoke a learning algorithm in order to do that. Both approaches, 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 to 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 [39,33], or ranked by their individual relevance [46,17,30]. In the former case, it is assumed that all possible input interaction patterns can occur, thus forcing a more complex search of the configuration space. In the latter case, instead, variables are assumed to be weakly correlated, so that their individual importance can be unambiguously assessed. We notice, however, that in the former case ranks can be used as a guideline to evaluate a subset selection process, in an in-between approach.

A class of powerful variable selection wrapper methods is based on global search strategies, such as Evolutionary Computation [38,28,20,6,29,5], Swarm Intelligence [19,1], and Simulated Annealing [22,41,42,21].

The definition of relevance itself can be subject to different interpretations [25], and the goal of the procedure can also be different, with some approaches aiming at comprehensive set (find all significant variables [23,27,33]) 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 [16]). Again, an in-between approach is possible [46] when an explicit cost function includes both a measure of complexity and performances. In this case, a continuum

of possible balances is provided by the relative weights given to these two terms. A direct measure of complexity, for instance, can be given by the number of selected inputs.

A problem that may arise in the incorrect use of wrapper approaches is the so called *selection bias*. It occurs when the learning machine is tested on the same data set used in the first place by the learning machine to select the inputs, or when the cross-validation is internal to the selection process. Ambroise and McLachlan in [4] pointed out this problem and recommended using *M-fold external cross-validation* or *bootstrap* for wrapper approaches to input selection.

In M-fold cross-validation [40], the training set is divided into M non overlapping subsets of equal size. A learner is trained on the union of $M-1$ of these subsets and then applied to the remaining subset to obtain an estimate of the prediction error. This process is repeated in turn for each of the M subsets, and the cross-validation error is given by the average of the M estimates of the prediction error thus obtained.

The application of M-fold cross-validation to a wrapper input selection method entails that the same input-selection method must be used in training the learner on the $M-1$ subsets combined at each stage of an (external) cross-validation of learner for the selected subset of variables (genes). In this way, for each fold, we might select a different subset of variables. Therefore, the subsets of relevant variables have to be aggregated with appropriate heuristics.

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

These hypotheses form the basis of the method we are presenting in this paper, which is based on optimizing a 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 [24] which is a global search method technique derived from Statistical Mechanics, while the learning algorithms employed in the paper are the Support Vector Machine [12] and the Classification Tree [10].

In the next section, we present the Simulated Annealing technique and describe how we applied it to the 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 speeding it up are shown in Sect.s 4 and 5. In Sect. 6 we draw the conclusions.

2 Simulated Annealing for Input Selection algorithm

The method for input selection we propose makes use of Simulated Annealing (SA) [24] that is a global search technique derived from Statistical Mechanics. SA is based on the Metropolis algorithm [31] proposed to simulate the behavior 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 current 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 distribution:

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

where K is the Boltzmann constant and T the temperature.

In SA, this idea is generalized to solve general optimization problems [24, 34] by using *ad hoc* selected cost functions (also known as *generalized energy functions*) instead of the physical energy. SA works as a probabilistic hill-climbing procedure searching for the global optimum of the cost function [37]. K is usually set to 1, while the temperature T plays the role of a control parameter of the search area, and is gradually lowered until no further improvements of the cost function are noticed. SA can be employed in very high-dimensional search spaces, given enough computational resources.

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, named *Simulated Annealing Input Selection* (*SAIS*), is described in the following of this section with reference to a step-by-step outline shown in Algorithm 1.

Algorithm 1 (SAIS - Simulated Annealing for Input Selection)

1. Initialize the parameters;
2. Initialize the binary string $\mathbf{g}$ and the temperature T;
3. Train and test the classifier and evaluate the generalized system energy E;
4. **do**
5. Initialize $f = 0$ (f is the number of iterations), h=0 (h is the number of successes);
   (a) **do**
   (b) Increment the number of iterations f;
   (c) Perturb the binary string $\mathbf{g}$;
   (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 string $\mathbf{g}$;
      ii. Increment the number of success h;
   (g) **endif**
   (h) **loop while** $h \leq h^*$ **and** $f \leq f_{max}$;
6. update $T = \alpha T$;
7. **loop while** $h > 0$;
8. end.

Let d be the dimensionality of the input space and $\mathbf{g} = (g_1, g_2, \ldots, g_d)$ be a binary string representing the system state (or 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 string $\mathbf{g}$.

The generalized energy E is defined as a linear combination of the *Classification Error* ε (i.e., the empirical risk) and of the number of selected inputs s:

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

Note that the term λs penalizes situations in which the number of selected inputs is high. The trade-off between size of input space and accuracy is controlled by the parameter λ (*penalization coefficient*). If $\lambda = 0$, solutions with low classification error ε are favored, regardless of the dimensionality of the input space. For high values of λ, instead, few input variables would be selected at the expense of some accuracy.

Due to possible redundancies of groups of input variables [26] and to the curse of dimensionality problem [8], often a good tuning of the penalization coefficient can allow to find a small set of inputs achieving a low classification error.

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

As suggested in [34], 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 the algorithm there are two nested cycles. The inner one (Steps 5(a)-5(h)) runs with constant temperature value T, and terminates at most after f_{max} iterations, or when we obtain h^* successes; the external cycle (Steps 4-7) updates the temperature value T and terminates when the inner cycle ends after f_{max} iterations and without any success, i.e., with $h = 0$.

As already pointed out, the SAIS algorithm seeks 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 $[v_{\min}, v_{\max}]$ regulate the variability of perturbation.

The ability of the method to explore the space of input configurations is controlled by parameters $w_{\min}$, $w_{\max}$, $v_{\min}$, and $v_{\max}$. The upper extremes, $w_{\max}$ and $v_{\max}$, control the extent of the variation, since they control the maximum number of flips to 1 and to 0, while the lower extremes, $w_{\min}$ and $v_{\min}$, when different from zero, impose a minimum number of mandatory flips, ensuring diversity in the explored configurations. Note that in any case the distribution of the dimensionality of the explored configurations is concentrated around the suggested value, although, if better performing configurations are found, they will be accepted by the optimization procedure.

These parameters can be used to tune the available moves at each iteration, and may be varied during the search. For instance, we may require a high diversity in the first phases of optimization, where several alternative configurations are searched, and reduce it in the subsequent iterations.

In order to avoid the selection bias problem, we have used the SAIS algorithm with M-fold external cross-validation, as suggested by Ambroise and McLachlan in [4]. With this approach we obtain a different subset of selected input variables for each fold. This is due to both the different data subsets used for training the learner for each fold, and to the intrinsic randomness of the SAIS method. To aggregate these subsets of relevant variables we define some ad-hoc voting techniques.

Definition 1 *Hard-Voted Relevance (HVR).* The *hard voted relevance* of an input is the count of its occurrences in the results obtained on the M folds.

Definition 2 *Vector of Hard-Voted Relevance (*$\mathbf{G}$*).* The vector $\mathbf{G} \equiv \sum_j \mathbf{g}^j$ ($j = 1, 2, ..., M$), with $\mathbf{g}^j$ being the string obtained by SAIS on the j-th fold, contains the HVR of all input variables.

Definition 3 *Vector of Aged Relevances (*$\mathbf{R}$*).* The vector $\mathbf{R} = (r_1, r_2, \ldots, r_d)$ is defined as follows:

– 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 $\mathbf{R}$:

$$\mathbf{R} = \gamma \mathbf{R} + \mathbf{g}, \qquad with \;\; \gamma \in [0, 1] \tag{3}$$

At the end of the SAIS the vector $\mathbf{R}$ measures how often each input has been selected in the last few successful moves of the algorithm and contains the *Aged Relevances* of inputs (γ is called *aging parameter*).

Definition 4 *Soft-Voted Relevance (SVR).* The *soft-voted relevance* of an input l is the sum of values of r_l, the end of SAIS in the M different folds.

Definition 5 *Vector of Soft-Voted Relevances (*$\mathbf{S}$*).* The *vector of soft-voted relevances* is the sum of vectors of aged relevances $\mathbf{R}^j$ obtained by SAIS on the j-th fold, namely $\mathbf{S} \equiv \sum_j \mathbf{R}^j$ (with $j = 1, 2, ..., M$).

Note that measures based on similar aged indexes are used also in other contexts, such as the implementation of policies in computer operating systems [43], sensor networks [15], and chaotic systems dynamics [7]. It is worth noting that in this paper the indexes of relevance are only used for evaluating and integrating the results of the SAIS algorithm.

In Table 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. 3. 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 Generalized Energy E (Steps 3 and 5d); f_{max}, h^*, α are used for annealing schedule (Steps 5 and 6); and γ is used to estimate the input relevances.

SAIS is a computationally intensive algorithm, but as we shall see in Sect. 4 it is faster than other wrapper methods, and it can be accelerated using some heuristics that we will present in Sect. 4.

It is worth noting that global search strategies for input selection such as SAIS and others already cited based, e.g., on Genetic Algorithms, Particle Swarm Optimization, and Simulated Annealing, can work with both numerical and categorical inputs. We note also that, due to the dependence of these techniques on random decisions, for each independent run on the same training set they can find a new subset of s inputs from the original d.

3 Experimental validation of SAIS

The SAIS algorithm has been implemented in the R language and statistical environment [36]. The experimental validation consisted of four experiments using a synthetic data set for evaluating the method in Experiment A , and using publicly available data sets (Experiments B, C and D). Experiments B and C apply the method to two popular Bionformatics data sets, while Experiment D employs a medical data sets with numerical and categorial inputs with the aim to test SAIS with data sets with mixed input.

In the first three experiments, we used the popular *Support Vector Machine* (SVM) [44] as a classifier, as implemented in Chang and Lin's LIBSVM [11]. 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 an additional parameter. Moreover, to avoid selection bias in the evaluation of classification performances of the SVM trained on the input selected, we implemented an external 10-fold cross validation. We chose also to have one unique run

Table 1 Data sets used in the experiments.

Experiment	**A**	**B**	**C**	**D**
Data set	**Synth**	**Leukemia**	***Colon***	**Cleve**
Number of numerical inputs	1000	7129	2000	6
Number of categorical inputs	0	0	0	7
Number of instances	50	38	62	303

Table 2 Parameters of SAIS-US algorithm and their values selected for the four experiments.

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

of SAIS for each fold. In experiment D we used Decision Trees as classifiers, due to the presence of categorical inputs.

In the experiments described in this section, in Step 5c of SAIS, we used a uniform probability distribution in the interval $[1, s]$ to select the number w of bits of $\mathbf{g}$ to be switched from 1 to 0, i.e.,

$$p_i = \frac{1}{s} \tag{4}$$

and, similarly, a uniform probability distribution in the interval $[1, v_{\max}]$, to select the number v of bits of $\mathbf{g}$ to be switched from 0 to 1, i.e.,

$$p_i = \frac{1}{1 - v_{\max}} \tag{5}$$

We call this this basic approach to system state perturbation *SAIS with Uniform Selection* (SAIS-US). In Sect. 4 we will show some modifications to SAIS-US, aimed to accelerate its convergence.

Table 1 lists the main properties of the data sets used in each experiment. The initialization of the parameters of SAIS and the techniques used in the four experiments are shown in Table 2, while an overall view of the results obtained is presented in Table 12. We shall now present a detailed analysis of the four experiments.

Table 3 SAIS-US on *Synth* data set: Inputs selected in the 10 runs of the external 10-fold validation procedure. For each fold we show the inputs selected, ranked according to their aged relevance index.

Rank	Run 1	Run 2	Run 3	Run 4	Run 5	Run 6	Run 7	Run 8	Run 9	Run 10
1	1	3	4	2	4	3	3	4	4	1
2	2	2	610	1	3	2	4	1	1	2
3	241	–	–	–	–	266	–	–	–	–
4	–	–	–	–	–	975	–	–	–	–

Table 4 *Synth* data set: Ranking of the first 10 inputs obtained after the external 10-fold validation procedure using the Hard Voted Relevance (HVR) and Soft Voted Relevance (SVR).

Rank	HVR	SVR
1	1, 2, 4	4
2	–	3
3	–	2
4	3	1
5	241	343

3.1 Experiment A

The first experiment has been conducted on a synthetic data set (Synth) of 50 instances and in a space of 1000 features, composed by two balanced classes (see Table 1). The first 2 inputs are able to linearly separate the two classes. The third and fourth inputs are the opposites respectively of the first and second inputs (this models problems with redundant inputs), while the remaining 996 were randomly generated with a uniform distribution.

As shown in Table 2, we used a Linear SVM as the learning machine and an external 10-fold validation technique. The best results have been obtained using a penalization parameter $\lambda = 2 \cdot 10^{-2}$ for the generalized energy: we found one misclassified pattern (2.0%) and a number of selected inputs s ranging between 2 and 4 as shown in Table 3.

With the exception of Run 3, in all runs the selected inputs are one of the following pairs of inputs $(1, 2)$, $(1, 4)$, $(2, 3)$, $(3, 4)$, and sometimes include also some random inputs (Run 1, Run 6). We note that in Run 3 SAIS-US selects only two inputs, 4 and 610, the latter being one of the randomly generated inputs.

The evaluation of the Hard Voted Relevance (HVR) and Soft Voted Relevance (SVR) helps us to filter the most relevant inputs. Table 4 shows the first 5 selected inputs and the value of these two indexes.

The two graphs on the top of Fig. 1 show the trend of the classification error ε and of the number of selected inputs versus the iteration number of the algorithm in a run of SAIS-US. Each iteration corresponds to a different value of temperature T (i.e. Step 5 and Step 6 of Algorithm 1). These graphs illustrate the ability of SAIS-US to minimize both the Classification Error ε and the number of relevant inputs s. Fig. 1 also shows that the same happens for the other experiments reported in this paper.

3.2 Experiment B

The first real data set we considered is the *Leukemia* data set[1] first presented by Golub et al. [16] (see Table 1). The Leukemia problem consists in characterizing two forms of acute

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

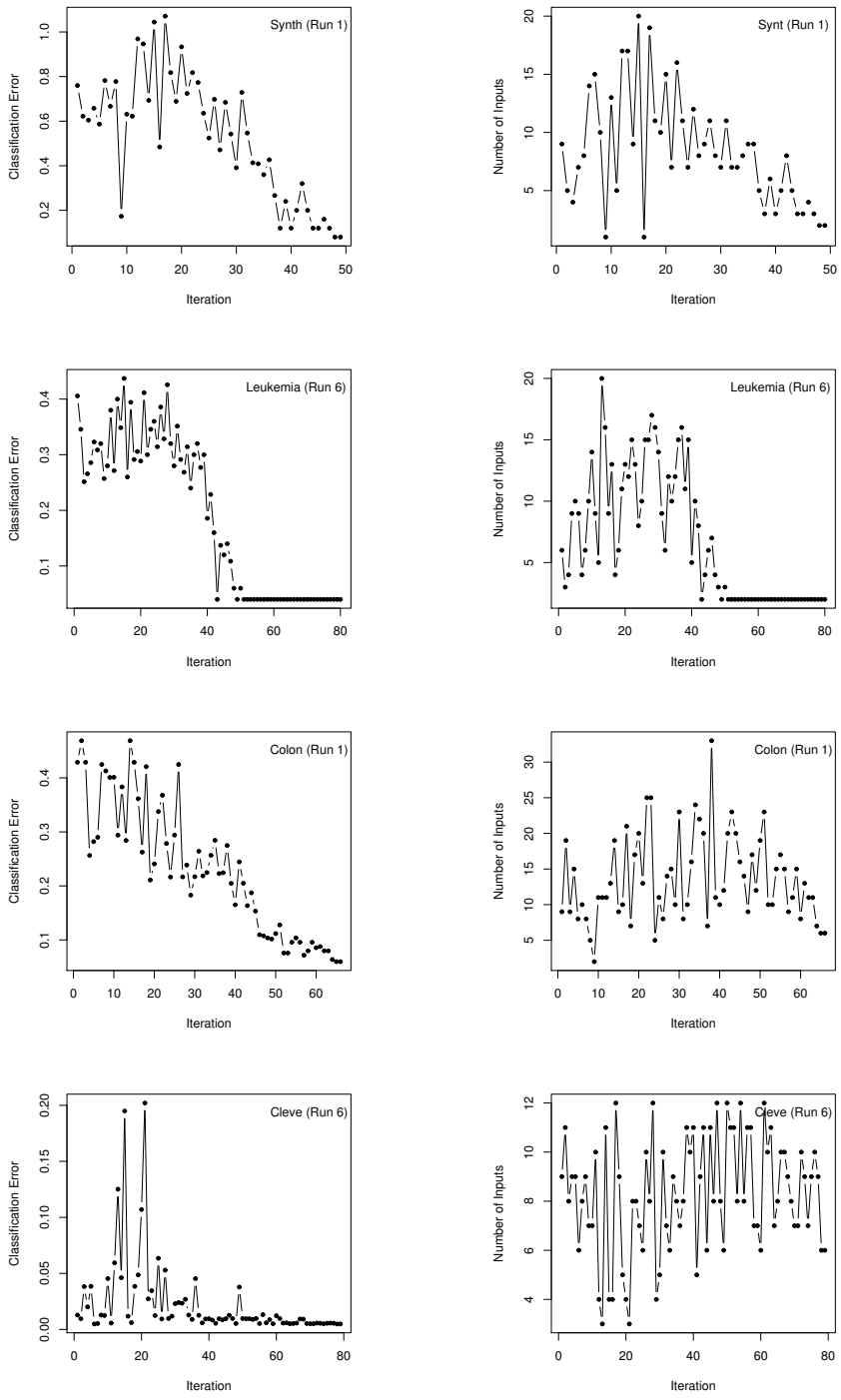

Fig. 1 Classification error ε and number of selected inputs s versus the number of iterations for: Run 1 of Experiment A on Synth data set; Run 6 of Experiment B on the Leukemia data set; Run 1 of Experiment C on the Colon data set; Run 6 of Experiment D on the Cleve data set. Each iteration corresponds to a different value of temperature T in Algorithm 1 (the plotted values are just before Step 6).

Table 5 SAIS-US on *Leukemia* data set: Inputs selected in 10 runs of the external LOO validation procedure. For each fold we show the inputs selected, according to the aged relevance index. The genes selected in Golub et al. [16] are underlined.

Rank	Run 1	Run 2	Run 3	Run 4	Run 5
Gene 1	X95735_at	M27891_at	X95735_at	X95735_at	J04615_at
Gene 2	U51166_at	X59812_at	X69819_at	M21812_at	L38928_at

Rank	Run 6	Run 7	Run 8	Run 9	Run 10
Gene 1	X95735_at	M55150_at	X95735_at	M16038_at	X95735_at
Gene 2	L76703_at	HG2171-HT2241_at	X78136_at	L16991_at	X87344_cds10_r_at

Table 6 *Leukemia* data set: Ranking of the first 10 inputs obtained after the external the LOO validation procedure using the Soft Voted Relevance. Genes which have also been found in [16] are underlined, and names are indicated.

Rank	**Soft Voted Relevance**	**Name**
1	X95735 _at	Zyxin
2	M27891_at	Cystatin
3	M55150_at	Fumarylacetoacetate
4	J04615_at	SNRPN
5	M63138_at	Cathepsin
6	X66365_at	CDK6
7	X71490_at	ATP6E
8	U72936_s_at	X-linked helicase II
9	S81003_at	L-UBC
10	M96326_rna1_at	Azurocidin

leukemia, Acute Lymphoblastic Leukemia (ALL) and Acute Myeloid 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. [16] selected a set of 50 most relevant genes.

Given the scarcity of instances in the sample, for the estimation of the Classification Error ε we applied the SAIS-US algorithm using a leave-one-out (LOO) validation technique and the assumptions and techniques shown in Table 2.

In the 38 runs of the LOO validation we obtained a total of two misclassifications, thus the LOO estimate of the classification error is 5.3%. For each run of LOO we started SAIS-US with a string $\mathbf{g}$ with 20 bits set to 1. The algorithm always ended with a set of two genes. Table 5 shows the genes selected by 10 runs of the LOO validation technique.

Table 6 shows the ranking obtained using Soft Voted Relevance. In the first three positions we found the genes X95735_at, M23197_at and M55150_at, that are also in the set selected by Golub et al. [16]. Hard Voted Relevance put in the first position the gene X95735_at, and in the second place the other 9 genes listed in Table 5. The most rele-

Table 7 SAIS-US on *Colon* data set: Inputs selected in runs 1, 2, 3, 4, 5 of the external 10-fold validation procedure. For each fold we show the inputs selected, ranked according to their aged relevance index. The gene U14971 (underlined in the table) has been also highlighted in [3].

Rank	Run 1	Run 2	Run 3	Run 4	Run 5
1	H20709	T63484	R84411	J00231	T51023
2	J03077	H20709	X12369	H20709	D31885
3	T47377	H43887	U14971	R72300	M64110
4	U37012	H88360	D63874	T57780	Z18538
5	M28214	T64878	R74349	T61602	U14973
6	R39465	H81068	X12466	X15183	M69043
7	–	–	H75955	T61446	T94350
8	–	–	M26383	T53412	J03210
9	–	–	U09413	U29656	–
10	–	–	R43914	Y00097	–
11	–	–	T95046	R56401	–
12	–	–	–	T48612	–
13	–	–	–	R73606	–
14	–	–	–	R43976	–
15	–	–	–	M87434	–

Table 8 SAIS-US on *Colon* data set: Inputs selected in runs 6, 7, 8, 9, 10 of the external 10-fold validation procedure. For each fold we show the input selected, ranked according to their aged relevance index. The genes T63484 and T51560 (underlined in the table) have been also highlighted in [3].

Rank	Run 6	Run 7	Run 8	Run 9	Run 10
1	U37012	H20709	M22382	J02854	J02854
2	H64489	J03077	L09604	T51560	T51023
3	H46994	U37012	R44052	R87126	M11799
4	J00231	T47377	R06601	R07007	D14520
5	X70326	Z14978	M36634	X55362	T57686
6	H02465	K03474	U27337	T49941	T61661
7	U19796	R52000	H46994	X14830	H65355
8	–	M94630	T57882	H82741	H73943
9	–	–	M55683	D43949	M98343
10	–	–	T74556	T55871	D13641
11	–	–	L20688	T74257	H14607
12	–	–	D13630	–	R56207
13	–	–	T63484	–	–
14	–	–	U04241	–	–
15	–	–	M57710	–	–
16	–	–	R56443	–	–
17	–	–	D30655	–	–
18	–	–	R98842	–	–
19	–	–	T66960	–	–
20	–	–	R62459	–	–
21	–	–	L11369	–	–

vant gene given by both indexes (i.e., X95735_at, Zyxin) has been highlighted by Guyon et al. [17] as well. Both computational and biological evidence of involvement of this protein in processes related to leukemia has also been discussed in [45].

Table 9 *Colon* data set: Ranking of the first 10 inputs obtained after the external the 10-fold validation procedure using the Hard Voted Relevance and Soft Voted Relevance. The gene T63484 (underlined) has been also highlighted in [3].

Rank	**Hard-Voted Relevance**	**Soft-Voted Relevance**
1	H20709	H20709
2	U37012	T51023
3	T63484	J02854
4	J03077	U37012
5	T51023	J03077
6	J00231	T47377
7	H46994	J00231
8	T47377	T92451
9	J02854	H64489
10	R39465	T63484

3.3 Experiment C

The second real data set on which we performed input selection is the *Colon* data set by Alon et al. [3]. 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 same patients who provided the tumor instances, so that positive instances are well paired to the corresponding negative instances. The actual data set used in the experiments[2] contains only the 2000 genes most clearly expressed in the experiments, those with the lowest minimal intensity across the 62 tissue instances (see Table 1).

As shown in Table 2, we used a penalization parameter $\lambda = 10^{-3}$, a linear SVM as the learning machine, and an external 10-fold cross-validation technique. For each run of the 10-fold validation, we obtained a different set of inputs ranging from 6 to 21 genes (Tables 7,8), that are listed ranked according to their aged relevance degree. In the 10 runs of the 10-fold validation we obtained a total of twelve misclassifications, and, therefore, the 10-fold estimate of the classification error is $\varepsilon = 19.4\%$. Higher values of λ lead to the selection of a smaller input space at the cost of higher classification errors.

Table 9 reports the genes' ranking using Hard-Voted Relevance and Soft-Voted Relevance. We can note that genes U14971, T63484 and T51560 reported in Fig.s 7,8 have been also highlighted in [3], but only gene T63484 appears in the lists of Table 9.

3.4 Experiment D

The fourth 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. The Cleve data set contains 303 instances (patients) with six categorical and eight numerical inputs (see Table 1). The data set contains 5 missing values in numerical attributes[3] that can be handled by Decision Trees. We focus on the problem of finding a set of inputs able to distinguish between health and disease.

As we have mixed inputs (numerical and categorical variables) we used a RPART classification tree [10, 35] as a classifier and external 10-fold validation technique (Table 2).

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

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

Table 10 SAIS-US on *Cleve* data set: Inputs selected in the 10 runs of the external 10-fold validation procedure, using as penalization parameter $\lambda = 10^{-4}$ in the generalized energy. For each fold we show the inputs selected, ranked according to their aged relevance index.

Rank	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Run 6**	**Run 7**	**Run 8**	**Run 9**	**Run 10**
1	chol	chol	chol	chol	chol	maxbps	chol	chol	chol	chol
2	maxbps	age	age	bps	oldpeak	oldpeak	maxbps	bps	age	oldpeak
3	age	oldpeak	oldpeak	maxbps	sex	bps	age	thal	maxbps	age
4	induced	bps	thal	chest	age	chol	oldpeak	oldpeak	bps	sex
5	–	ecg	fbs	–	–	ecg	induced	–	sex	–

Table 11 *Cleve* data set: Ranking of the first four inputs selected with SAIS-US using the external 10-fold validation procedure.

Rank	**HVR**	**HVR**
1	sex	ncol
2	induced	chest
3	ncol	induced
4	ecg	thal

Table 12 Summary of the results obtained in the four experiments.

Experiment	**A**	**B**	**C**	**D**
Data set	**Synth**	**Leukemia**	**Colon**	**Cleve**
λ	10^{-2}	10^{-2}	$2 \cdot 10^{-3}$	10^{-4}
Classification error	2.0%	5.3%	19.4%	23.8%
Inputs selected	[2–4]	2	[6–21]	[4–5]

RPART can work with mixed data sets and is is able to handle missing attribute values. In Table 10, we report the results obtained by setting the penalization coefficient $\lambda = 10^{-4}$. The inputs selected are ranked according to their aged relevance. In Table 11, we show the inputs ranked using hard voted relevance and soft voted relevance after the external 10-fold validation procedure.

4 Speeding up SAIS

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

Let us consider, e.g., the method by Guyon et al. [17] which performs input 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 inputs it must run an SVM for each dimension of the input space, starting from thousands down to one. In SAIS, instead, when we work with numerical inputs only and use SVMs, we have many more (hundreds of thousand) SVMs to train, but each learning procedure is fast, as it performs classification in a space of small dimension (only some tens of inputs). This is an advantage over feature elimination procedures.

Table 13 Time duration in hours of input selection methods on Leukemia database obtained on a Pentium IV 1900 MHz personal computer. Running times for SAIS variants are averaged on 10 runs and reported along with their standard deviations. As RFE is deterministic, its standard deviation is 0.

Method	**time**	**standard deviation**
RFE [17]	2.47	0
SAIS US	1.62	0.68
SAIS SPS ($\varphi = 0$)	1.25	0.39
SAIS SPS ($\varphi = 0.25$)	1.52	0.44
SAIS SPS ($\varphi = 0.50$)	1.49	0.75
SAIS SRS ($\alpha = 2$)	1.04	0.24
SAIS SRS ($\alpha = 3$)	0.60	0.18
SAIS SRS ($\alpha = 4$)	0.65	0.21
SAIS STS ($c = 2$)	0.68	0.41
SAIS STS ($c = 5$)	0.52	0.13
SAIS STS ($c = 8$)	0.47	0.13

In Table 13 we present a comparison of convergence times obtained using RFE [17], 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. [16] on a Pentium IV 1900 MHz personal computer and averaged on 10 runs of each algorithm.

SAIS implements a combinatorial search on the all possible input subspaces on the basis of the penalized classification error, which is more robust than direct evaluation of sensitivity, with respect to both noise and sample variability. However, in case of numerical inputs and linear discriminant classifiers, it is also possible to embed a direct input sensitivity evaluation step in the algorithm.

The heuristic we propose to this end applies 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 sensitivity of the discriminant function to the input x_i, can be evaluated as:

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

or, better, as:

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

Inputs sensitivity is often exploited by "embedded" wrapper methods for input selection as notion of input relevance [46,17,30]. As noted 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 inputs is taken into account.

This kind of advice from input sensitivity can be useful in the perturbation step of SAIS (Step 5c). For example, we can make use of normalized input sensitivity for selecting the w bits of $\mathbf{g}$ in the interval $[1, s]$ to be flipped from 1 to 0 in Step 5c of SAIS, instead of using Uniform Selection as indicated in Sect. 3. We have implemented the following selection techniques borrowed from Genetic Algorithms literature (see, e.g., [32]):

1. Sensitivity Proportioned Selection (SPS). A bit of the string 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. Sensitivity Ranked Selection (SRS). A vector of ranks of inputs $\rho = (\rho_1, \rho_2, ..., \rho_d)$ is obtained by sorting on the basis of their normalized relevance. Then each bit of string $\mathbf{g}$ 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. Sensitivity Tournament Selection (STS). To select a bit of string $\mathbf{g}$ 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 highest normalized sensitivity is selected and switched to 0.

In SPS if $\varphi = 1$ we have the basic perturbation with Uniform Selection used in Sect. 3, 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 Table 13, the best speed-up has been obtained for the pure roulette-wheel algorithm (i.e., $\varphi = 0$).

We can note that using SPS, the values of normalized input sensitivity after few iterations tend to become very similar for all inputs, independently on the value assigned to φ. In the 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 selection and tournament selection.

Using STS, stagnation is avoided at the cost of a reordering overhead, as it happens in Genetic Algorithms. From Table 13, we notice a significant speed-up with respect to SPS, especially with $\alpha = 3$ or 4. Note that when $\alpha = 1$, STS becomes the basic perturbation with Uniform Selection used in Sect. 3.

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 allows to obtain the fastest runs of SAIS, as shown in Table 13, especially with a large number of competitors (c).

We point out that when we apply these 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 speed-up of SAIS, otherwise the move will be rejected at Step 5f. However, the results obtained with these heuristics are comparable with those obtained with SAIS US and illustrated in Table 5.

5 Conclusions

We have presented a global search method for gene selection, based on Simulated Annealing. The method implements heuristics for a comprehensive, yet efficient search of the configuration space. Its properties are verified by means of extensive experiments, which yield results in good agreement with published outcomes of alternative methods, as well as with biological evidence.

These same experiments have shown that, even though the core of the method is a random search procedure, the specific implementation and the ad-hoc heuristics used to guide and to speed up the method itself give rise to a relatively efficient algorithm. Once we have shown that efficiency is not an issue, then it becomes more attractive to use a global search method rather than other methods based on local search heuristics with comparable (or even higher) computational cost.

Even more efficient implementations can also be devised, based on the parallel computing architectures provided by commodity hardware, like multi-core CPUs with simultaneous multi-threading, or highly parallel GPUs with general-purpose programming frameworks, which are currently ubiquitous and used for increasingly complex scientific computing applications.

Acknowledgements We thank Chih-Chung Chang for help about internals of LIBSVM in R. A discussion with Giorgio Valentini helped us to clarify an important issue of this paper. Work funded by the the Italian Ministry of Education, University and Research (code 2004062740).

References

1. D. K. Agrafiotis, and W. Cedeo, Feature selection for structure-activity correlation using binary particle swarms, *J. Med. Chem.*, vol. 45, pp. 1098–1107.
2. 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.
3. 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, *Proceedings of the National Academy of Sciences*, USA, vol.96 no. 12 pp. 6745–6750, 1999.
4. C. Ambroise and G.J. McLachlan, Selection bias in gene extraction on the basis of microarray gene-expression data, *Proceedings of the National Academy of Sciences*, USA, vol. 99 pp. 6562–6566, 2002.
5. Andonie, R., Fabry-Asztalos, L., Abdul-Wahid, Collar, C., S., Salim, N., An Integrated Soft Computing Approach for Predicting Biological Activity of Potential HIV-1 Protease Inhibitors, *Proceedings of the IEEE International Conference on Neural Networks*, pp. 7495–7502, 2006.
6. A.S. Bangalore, R.E. Shaffer, G.W. Small and M.A. Arnold, Genetic Algorithm-Based Method for Selecting Wavelength and Model Size for Use with Partial Least-Squares Regression: Application to Near-Infrared Spectroscopy, *Anal. Chem.*, vol. 68, pp. 4200–4212, 1996.
7. E. Barkai, Aging in Subdiffusion Generated by a Deterministic Dynamical System, *Phys. Rev. Lett.* vol. 90, 104101, 2003.
8. R.. Bellman, *Adaptive Control Processes: A Guided Tour*, Princeton University Pressv, 1961.
9. 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.
10. L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth & Brooks, Pacific Grove, CA., 1984.
11. 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.
12. C. Cortes and V. Vapnik, Support vector networks, *Machine Learning*, vol. 20 pp. 273 – 297, 1995.
13. R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.
14. 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.
15. 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.
16. 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.
17. 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.

18. I. Guyon, A. Elisseeff, *An introduction to variable and feature selection*, *Journal of Machine Learning Research*, vol. 3 pp. 1157–1182, 2003.
19. S. Izrailev, and D. K. Agrafiotis, Variable selection for QSAR by artificial ant colony systems, *SAR and QSAR in Environ. Res.*, vol 13, pp. 417–423, 2002.
20. D. Jouan-Rimbaud, D.-L. Massart, R. Leardi, O.E.de Noord, Genetic Algorithms as a Tool for Wavelength Selection in Multivariate Calibration, *Anal. Chem.*, vol. 67, pp. 4295-4301, 1995.
21. Justin C.W. Debuse, and Victor J. Rayward-Smith, Feature Subset Selection within a Simulated Annealing Data Mining Algorithm, *Journal of Intelligent Information Systems*, vol. 9, pp. 57–81, 1997.
22. J.H. Kalivas, N. Roberts and J.M. Sutter, Global Optimization by Simulated Annealing with Wavelength Selection for Ultraviolet-Visible Spectrophotometry, *Analytical Chemistry*, vol. 61, pp. 2024–2030, 1989.
23. 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.
24. S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, vol. 220, pp.661–680, 1983.
25. R. Kohavi and G. John, Wrappers for Feature Subset Selection, *Artificial Intelligence*, vol. 97, nos. 1–2, pp. 273–324, 1997.
26. 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.
27. I. Kononenko, *Estimating Attributes: Analysis and Extensions of RELIEF*, Proc. Seventh European Conf. Machine Learning, pp. 171–182, 1994.
28. H. Kubinyi, Variable Selection in QSAR Studies. I. An Evolutionary Algorithm, *Quant. Struct.-Act. Relat.*, vol. 13, pp. 285–294, 1994.
29. R. Leardi, A.L. Gonzles, Genetic Algorithms Applied to Feature Selection in PLS Regression: How and When to Use Them, *Chemometrics and Intelligent Laboratory Systems*, vol. 41, pp. 195–207, 1998.
30. F. Masulli and S. Rovetta, Random Voronoi ensembles for gene selection, *Neurocomputing*, vol. 55, no.s 3-4, pp. 721-726, 2003.
31. 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.
32. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin Heidelberg New York, 3. edition, 1998.
33. 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.
34. W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, *Numerical Recipes in C*, (2nd Edition), Cambridge University Press, 1992.
35. B. D. Ripley, *Pattern recognition and neural networks*. Cambridge university press, Cambridge, 1996.
36. 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
37. F. Romeo and A. Sangiovanni-Vincentelli. *Probabilistic Hill-Climbing Algorithms: Properties and Applications*. Computer Science Press, Chapell Hill, NC, 1985.
38. W. Siedlecki and J. Sklansky, A note on genetic algorithms for large-scale feature selection, *Pattern Recognition Letters*, vol. 10, pp. 335–347, 1989.
39. N. Slonim and N. Tishby, Agglomerative information bottleneck, in *Advances in Neural Information Processing Systems*, pages 617-623, 2000.
40. 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.
41. J.M. Sutter and J.H. Kalivas, Comparison of Forward Selection, Backward Elimination, and Generalized Simulated Annealing for Variable Selection, *Microchemical Journal*, vol. 47, pp. 60–66, 1993.
42. J.M. Sutter, S.L. Dixon and P.C. Jurs, A utomated Descriptor Selection for Quantitative Structure - Activity Relationships using Generalized Simulated Annealing, *Journal of Chemical Information and Computer Sciences*, vol. 35, pp. 77–84, 1995.
43. A. Tanenbaum, *Modern Operating Systems*, (2nd Edition), Prentice Hall, 2001.
44. V. N. Vapnik, *The nature of statistical learning theory*, Springer-Verlag New York, Inc., New York, NY, USA, 1995.
45. Wang Y, Tetko IV, Hall MA, Frank E, Facius A, Mayer KF, Mewes HW, Gene selection from microarray data for cancer classification–a machine learning approach *Comput Biol Chem.* 2005 Feb;29(1):37-46.
46. 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.