Feature Selection Algorithms: A Survey and Experimental Evaluation

Luis Carlos Molina, Lluís Belanche, Àngela Nebot
Universitat Politècnica de Catalunya
Departament de Llenguatges i Sistemes Informátics
Jordi Girona 1-3, Campus Nord C6,
08034, Barcelona, Spain.
{lcmolina,belanche,angela}@lsi.upc.es

Abstract

In view of the substantial number of existing feature selection algorithms, the need arises to count on criteria that enables to adequately decide which algorithm to use in certain situations. This work reviews several fundamental algorithms found in the literature and assesses their performance in a controlled scenario. A scoring measure ranks the algorithms by taking into account the amount of relevance, irrelevance and redundance on sample data sets. This measure computes the degree of matching between the output given by the algorithm and the known optimal solution. Sample size effects are also studied.

1. Introduction

The feature selection problem in terms of supervised inductive learning is: given a set of candidate features select a subset defined by one of three approaches: a) the subset with a specified size that optimizes an evaluation measure, b) the subset of smaller size that satisfies a certain restriction on the evaluation measure and c) the subset with the best commitment among its size and the value of its evaluation measure (general case). The generic purpose pursued is the improvement of the inductive learner, either in terms of learning speed, generalization capacity or simplicity of the representation. It is then possible to understand better the results obtained by the inducer, diminish its volume of storage, reduce the noise generated by irrelevant or redundant features and eliminate useless knowledge.

A feature selection algorithm (FSA) is a computational solution that is motivated by a certain definition of *relevance*. However, the relevance of a feature –as seen from the inductive learning perspective– may have several definitions depending on the objective that is looked for. An irrelevant feature is not useful for induction, but not all relevant features are necessarily useful for induction [14].

The FSAs can be classified according to the kind of output they yield: (1) those algorithms giving a (weighed) linear order of features and (2) those algorithms giving a subset of the original features. Both types can be seen in an unified way by noting that in (2) the weighting is binary.

The work presented in this paper is centered in FSAs tackling the feature selection problem of type (2), studied for many years by the statistical [18] as well as the machine learning [38] communities. Research developed within the machine learning area is usually focused on the proposal of new algorithms, theoretical learning results of existing algorithms or empirical studies (evaluations or applications).

In this research, several fundamental algorithms found in the literature are studied to assess their performance in a controlled scenario. To this end, a measure to evaluate FSAs is proposed that takes into account the particularities of relevance, irrelevance and redundance on the sample data set. This measure computes the degree of matching between the output given by a FSA and the known optimal solution. Sample size effects are also studied. The results illustrate the strong dependence on the particular conditions of the FSA used and on the amount of irrelevance and redundance in the data set description, relative to the total number of features. This should prevent the use of a single algorithm even when there is poor knowledge available about the structure of the solution.

The paper is organized as follows: in section 2 we review some different approaches to provide with a formal definition of relevance. In section 3 we set a more precise definition of the feature selection problem and survey the main characteristics of a FSA in an unified framework. Next, we describe some of the most widespread FSA in machine learning in terms of this framework. The methodology and tools used for the empirical evaluation are covered in section 5. Finally the experimental study is described in section 6 as well as the results. The paper ends with the conclusions and the knowledge gained.

2 Relevance of a Feature

The purpose of a FSA is to identify relevant features according to a definition of relevance. However, the notion of relevance in machine learning has not yet been rigorously defined on a common agreement [6]. Let E_i, with $1 \leq i \leq n$, be domains of features $X = \{x_1, \ldots, x_n\}$; an *instance space* is defined as $E = E_1 \times \ldots \times E_n$, where an instance is a point in this space. Consider p a probability distribution on E and T a space of labels (e.g. classes). It is desired to model or identify an objective function $c : E \rightarrow T$ according to its relevant features. A data set S composed by $|S|$ instances can be seen as the result of sampling E under p a total of $|S|$ times and labelling its elements using c.

A primary definition of relevance [9] is the notion of being "relevant with respect to an objective". It is assumed here to be a classification objective.

Definition 1 (Relevance with respect to an objective) *A feature $x_i \in X$ is relevant to an objective c if there exist two examples A, B in the instance space E such that A and B differ only in their assignment to x_i and $c(A) \neq c(B)$.*

In other words, if there exist two instances that can only be classified thanks to x_i. This definition has the inconvenience that the learning algorithm can not necessarily determine if a feature x_i is relevant or not, using only a sample S of E. Moreover, if the problem representation is redundant (e.g., some features are replicated), it will never be the case that two instances differ only in one feature. A proposal oriented to solve this problem [24] includes two notions of relevance, one with respect to a *sample* and another with respect to the *distribution*.

Definition 2 (Strong relevance with respect to *S*) *A feature $x_i \in X$ is strongly relevant to the sample S if there exist two examples $A, B \in S$ that only differ in their assignment to x_i and $c(A) \neq c(B)$.*

That is to say, it is the same Definition 1, but now $A, B \in S$ and the definition is with respect to S.

Definition 3 (Strong relevance with respect to *p*) *A feature $x_i \in X$ is strongly relevant to an objective c in the distribution p if there exist two examples $A, B \in E$ with $p(A) \neq 0$ and $p(B) \neq 0$ that only differ in their assignment to x_i and $c(A) \neq c(B)$.*

This definition is the natural extension of Definition 2 and, contrary to it, the distribution p is assumed to be known.

Definition 4 (Weak relevance with respect to *S*) *A feature $x_i \in X$ is weakly relevant to the sample S if there exists at least a proper $X' \subset X$ ($x_i \in X'$) where x_i is strongly relevant with respect to S.*

A weakly relevant feature can appear when a subset containing at least one strongly relevant feature is removed.

Definition 5 (Weak relevance with respect to *p*) *A feature $x_i \in X$ is weakly relevant to the objective c in the distribution p if there exists at least a proper $X' \subset X$ ($x_i \in X'$) where x_i is strongly relevant with respect to p.*

These definitions are important to decide what features should be conserved and which can be eliminated. The strongly relevant features are, in theory, important to maintain a structure in the domain, and they should be conserved by any feature selection algorithm in order to avoid the addition of ambiguity to the sample. Weakly relevant features could be important or not depending on the other features already selected and on the evaluation measure that has been chosen (accuracy, simplicity, consistency, etc.).

From another point of view, instead of focusing in which features are relevant, it is possible to use relevance as a "complexity measure" with respect to the objective c. In this case, it will depend on the chosen inducer.

Definition 6 (Relevance as a complexity measure) [9] *Given a data sample S and an objective c, define $r(S, c)$ as the smallest number of relevant features to c using Definition 1 only in S, and such that the error in S is the least possible for the inducer.*

In other words, it refers to the smallest number of features required by a specific inducer to reach optimum performance in the task of modelling c using S.

Definition 7 (Incremental usefulness) [13] *Given a data sample S, a learning algorithm L, and a subset of features X', the feature x_i is incrementally useful to L with respect to X' if the accuracy of the hypothesis that L produces using the group of features $\{x_i\} \cup X'$ is better than the accuracy reached using only the subset of features X'.*

This definition is specially natural in FSAs that search in the feature subset space in an incremental way, adding or removing features to a current solution. It is also related to a traditional understanding of relevance in the philosophy literature [21].

Definition 8 (Entropic relevance) [55] *Denoting the (Shannon) entropy by $H(x)$ and the mutual information by $I(x; y) = H(x) - H(x|y)$ (the difference of entropy in x generated by the knowledge of y), the entropic relevance of x to y is defined as $r(x; y) = I(x; y)/H(y)$.*

Let X be the original set of features and let C be the objective seen as a feature, a set $X' \subset X$ is sufficient *if $I(X'; C) = I(X, C)$ (i.e., if it preserves the learning information). For a sufficient set X', it turns out that*

$r(X';C) = r(X,C)$. *The* most favorable *set is that sufficient set $X' \subset X$ for which $H(X')$ is smaller. This implies that $r(C;X')$ is greater. In short, the aim is to have $r(C;X')$ and $r(X';C)$ jointly maximized.*

To make these definitions more clear, we borrow [9] an example that considers concepts expressible as disjunctions of features (e.g., $x_1 \vee x_3 \vee x_7$), assuming that the learning algorithm has access to the following 5 examples:

```
100000000000000000000000000000 +
111111111100000000000000000000 +
000000000011111111110000000000 +
000000000000000000001111111111 +
000000000000000000000000000000 -
```

The relevant features using Definition 1 depend on the actual (unknown) objective, although any consistent disjunction must include the first feature. Using Definitions 2, 3, 4 and 5 it can be concluded that x_1 (both with regard to S and to p) is strongly relevant and the rest of features are weakly relevant. Using Definition 6 it is simply stated that there are three relevant features ($r(S,c) = 3$), because this is the minimum number of features leading to a consistent disjunction. Notice that the features are not specified (e.g., $x_1 \vee x_{11} \vee x_{21}$). Definition 7 depends on the inducer. As an example, given $X' = \{x_1, x_2\}$, a set of already selected features, none of $x_3 \dots x_{10}$ would be incrementally useful, and any of $x_{11} \dots x_{30}$ would. Definition 8 requires the computation of the corresponding mutual entropies. Notice this is the only definition that considers relevance in a quantitative way.

3 Algorithms for Feature Selection

A FSA should be seen as a computational approach to a definition of relevance, although in many cases the previous definitions are followed in a somewhat loose sense.

3.1 Feature Selection Definition

Let X be the original set of features, with cardinality $|X| = n$. The *continuous* feature selection problem refers to the assignment of weights w_i to each feature $x_i \in X$ in such a way that the order corresponding to its theoretical relevance is preserved. The *binary* feature selection problem refers to the assignment of binary weights. This can be carried out directly (like many FSAs in machine learning [2, 13, 22]), or *filtering* the output of the continuous problem solution (see §6.2).

These are quite different problems reflecting different design objectives. In the continuous case, one is interested in keeping all the features but in using them differentially in the learning process. On the contrary, in the binary case one is interested in keeping just a subset of the features and using them equally in the learning process.

The feature selection problem can be seen as a search in a hypothesis space (set of possible solutions). In the case of the binary problem, the number of potential subsets to evaluate is 2^n. In this case, a general definition is [29]:

Definition 9 (Feature Selection) *Let $J(X')$ be an evaluation measure to be optimized (say to maximize) defined as $J : X' \subseteq X \to \mathbb{R}$. The selection of a feature subset can be seen under three considerations:*

- *Set $|X'| = m < n$. Find $X' \subset X$, such that $J(X')$ is maximum.*
- *Set a value J_o, this is, the minimum J that is going to be tolerated. Find the $X' \subseteq X$ with smaller $|X'|$, such that $J(X') \geq J_o$.*
- *Find a compromise among minimizing $|X'|$ and maximizing $J(X')$ (general case).*

Notice that, with these definitions, an optimal subset of features is not necessarily unique.

3.2 Characterization of FSAs

There exist in the literature several considerations to characterize feature selection algorithms [9, 19, 32]. In view of them it is possible to describe this characterization as a search problem in the hypothesis space as follows:

Search Organization. General strategy with which the space of hypothesis is explored. This strategy is in relation to the portion of hypothesis explored with respect to their total number.

Generation of Successors. Mechanism by which possible variants (successor candidates) of the current hypothesis are proposed.

Evaluation Measure. Function by which successor candidates are evaluated, allowing to *compare* different hypothesis to guide the search process.

3.2.1 Search Organization

A search algorithm is responsible for driving the feature selection process using a specific strategy. Each *state* in the search space specifies a *weighting* $w_1, \dots, w_n$ of the possible features of X, with $|X| = n$. In the binary case, $w_i \in \{0,1\}$, whereas in the continuous case $w_i \in [0,1]$. Notice we are stating that relevance should be upper and lower bounded. Also in the binary case a partial order $\prec$ exists in the search space, with $S_1 \prec S_2$ if $S_1 \subset S_2$ (see

Fig. 1), whereas in the continuous case $S_1 \prec S_2$ if, for all i, $w_i(S_1) \leq w_i(S_2)$ holds (see Fig. 2).

In general, a search procedure examines only a part of the search space. When a specific state has to be visited, the algorithm uses the information of the previously visited states and eventually heuristic knowledge about non-visited ones.

Being L a (labeled) list of weighed subsets of features (i.e. states), L maintains the (ordered) current list of solutions. The labels indicate the value of the evaluation measure. We consider three types of search: exponential, sequential and random. Most sequential algorithms are characterized by $|\mathtt{L}| = 1$, whereas exponential and random ones typically use $|\mathtt{L}| \geq 1$.

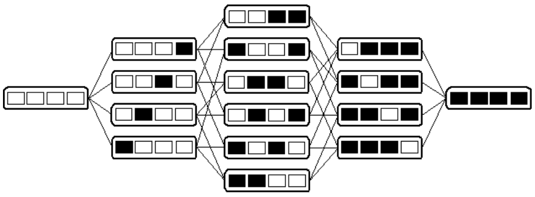


Figure 1. States in the binary search space involving 4 features. A black square represents the inclusion of a feature in the state and a white square represents its exclusion.

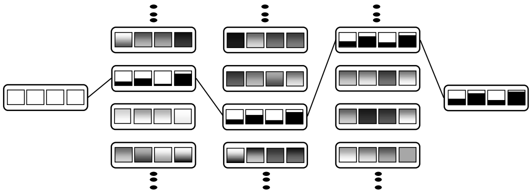


Figure 2. A path of states in the continuous search space involving 4 features. Relevances are represented as a degree of filling.

Exponential Search: It corresponds to algorithms that carry out searches whose cost is $O(2^n)$. The exhaustive search is an optimal search, in the sense that the best solution is guaranteed. An optimal search need not be exhaustive; for example, if an evaluation measure is *monotonic* a Branch and Bound [42] algorithm is optimal. A measure J is monotonic if for any two subsets S_1, S_2 and $S_1 \subseteq S_2$, then $J(S_1) \geq J(S_2)$. Another example would be an A^* search with an admissible heuristic [44].

Sequential Search: This sort of search selects one among all the successors to the current state. This is done in an iterative manner and once the state is selected it is not possible to go back. Although there is no explicit backtracking the number of such steps must be limited by $O(n)$ in order to qualify as a sequential search. The complexity is determined taking into account the number k of evaluated subsets in each state change. The cost of this search is therefore polynomial $O(n^{k+1})$. Consequently, these methods do not guarantee an optimal result, since the optimal solution could be in a region of the search space that is not visited.

Random Search: The idea underlying this type of search is to use its randomness to avoid the algorithm to stay on a local minimum and to allow temporarily moving to other states with worse solutions. These are *anytime* algorithms [32] and can give several optimal subsets as solution.

3.2.2 Generation of Successors

Up to five different operators can be considered to generate a successor for each state: *Forward, Backward, Compound, Weighting, and Random*.

All of the operators act by modifying in some way the weights w_i of the features x_i, with $w_i \in \mathbb{R}$ (in the case of the *weighting* operator), or $w_i \in \{0, 1\}$ (in the case of the rest of operators). In the following descriptions, it is assumed that the evaluation measure J is to be maximized.

Forward: This operator adds features to the current solution X', among those that have not been selected yet. In each step, the feature that makes J be greater is added to the solution. Starting with $X' = \emptyset$, the *forward* step consists of:

$$X' := X' \cup \{x_i \in X \setminus X' \mid J(X' \cup \{x_i\}) \text{ is bigger}\} \quad (1)$$

The stopping criterion can be: $|X'| = n'$ (if n' has been fixed in advance), the value of J has not increased in the last j steps, or it surpasses a prefixed value J_0. The cost of the operator is $O(n)$. The main disadvantage is that it is not possible to have in consideration certain basic interactions among features. For example, if x_1, x_2 are such that $J(\{x_1, x_2\}) \gg J(\{x_1\}), J(\{x_2\})$, neither x_1 and x_2 could be selected, in spite of being very useful.

Backward: This operator removes features from the current solution X', among those that have not been removed yet. In each step, the feature that makes J be greater is removed from the solution. Starting with $X' = X$, the *backward* step consists of:

$$X' := X' \setminus \{x_i \in X' \mid J(X' \setminus \{x_i\}) \text{ is bigger}\} \quad (2)$$

The stopping criterion can be: $|X'| = n'$, the value of J has not increased in the last j steps, or it falls below a prefixed value J_0. This operator remedies some problems although there still will be many *hidden* interactions (in the sense of being unobtainable). The cost is $O(n)$, although in practice it demands more computation than its forward counterpart [27].

Both operators (forward and backward) can be generalized selecting, at each step, subsets of k elements X'' and selecting the one making $J(X' \cup X'')$ or $J(X' \backslash X'')$ bigger, respectively. The cost of the operator is then $O(n^k)$.

Compound: The idea of this tactic is simple: apply f consecutive forward steps *and* b consecutive backward ones. If $f > b$ the net result is a forward operator, otherwise it is a backward one. An interesting approach is to perform the forward *or* the backward steps, depending on the respective values of J. This allows to discover new interactions among features. An interesting "backtracking mechanism" is obtained, although other stopping conditions should be established if $f = b$. For example, for $f = b = 1$, if x_i is added and x_j is removed, this could be undone in the following steps. A possible stopping criterion is $x_i = x_j$. In sequential FSA, the condition $f \neq b$ assures a maximum of n steps, with a total cost $O(n^{f+b+1})$.

Weighting: In the weighting operators, the search space is continuous, and all of the features are present in the solution to a certain degree. A successor state is a state with a different weighting. This is typically done by iteratively sampling the available set of instances.

Random: This group includes those operators that can potentially generate *any other* state in a single step. The rest of operators can also have random components, but they are restricted to some criterion of "advance" in the number of features or in improving the measure J at each step.

3.2.3 Evaluation Measures

There are several approaches to evaluate the goodness $J(X')$ of a feature subset X'. It is clear to observe that the relevance of a feature is solely a function of this measure and not a characteristic of the feature itself. Another important consideration is the fact that the range and scale of J are immaterial. What counts is that the *relative* values assigned to different subsets reflect their greater or lesser relevance to the objective function. Among the reviewed measures the probabilistic and the interclass distances, together with consistency, are measures of class separability. Further, the interclass distance, consistency, entropy and estimations of the probability of error may not require the explicit modeling of probability distributions.

Let $J : X' \subseteq X \to \mathbb{R}$ be an evaluation measure to be maximized, where X' is a (weighed) feature subset.

Probability of error: Provided the ultimate goal is to build a classifier able of correctly labelling instances generated by the same probability distribution, minimizing the (bayesian) probability of error P_e of the classifier seems to be the most natural choice. Therefore, it is also a clear choice for J.

Let $\vec{x} \in \mathbb{R}^n$ represent the unlabeled instances, and $\Omega = \{\omega_1, \ldots, \omega_m\}$ a set of labels (classes), so that $c : \mathbb{R}^n \to \Omega$. Such probability is defined as [18]:

$$P_e = \int [1 - \max_i P(\omega_i|\vec{x})] p(\vec{x}) d\,\vec{x} \qquad (3)$$

where $p(\vec{x}) = \sum_{i=1}^{m} p(\vec{x}|\omega_i) P(\omega_i)$ is the (unconditional) probability distribution of the instances, and $P(\omega_i|\vec{x})$ is the *a posteriori* probability of ω_i being the class of $\vec{x}$.

Since the class-conditional densities are usually unknown, they can either be explicitly modeled (using parametric or non-parametric methods) or implicitly via the design of a classifier that builds the respective decision boundaries between the classes [18]. Some of these classifiers, like the one-nearest-neighbor rule, have a direct relation to the probability of error.

The use of (an estimate $\hat{P}_e$ of) this probability by means of the construction of a classifier, using a sample dataset, is the base of the *wrapper* methods [26]. Provided the classifier has been built using only a subset $X' \subset X$ of the features, we have:

$$\hat{P}_e = 1 - \frac{|S^+_{TE}|}{|S_{TE}|} \qquad (4)$$

so that $J = 1 - \hat{P}_e$, being S_{TE} a test data sample, and S^+_{TE} the subset of S_{TE} where the classifier performed correctly (again using only a partial description X').

The estimation $\hat{P}_e$ may require the use of more elaborate methods than a simple holdout procedure (cross validation, bootstrapping) in order to yield a more reliable value.

Divergence: These measures compute a probabilistic distance or *divergence* among the class-conditional probability densities $p(\vec{x}|\omega_i)$, using the general formula:

$$J = \int f[p(\vec{x}|\omega_1), p(\vec{x}|\omega_2)] d\vec{x} \qquad (5)$$

To qualify as a valid measure, the function f must be such that the value of J satisfies the following conditions: (a) $J \geq 0$, (b) $J = 0$ only when the $p(\vec{x}|\omega_i)$ are equal and (c) J is maximum when they are non-overlapping. If the features used in a solution $X' \subset X$ are good ones, the divergence among the conditional probabilities will be significant. Poor features will result in very similar probabilities. Some classical choices are [18]:

Chernoff

$$f(a, b) = a^s b^{1-s},\ s \in [0, 1] \text{ and then } J_{Che} = -\ln J \qquad (6)$$

Bhattacharyya

$$f(a, b) = \sqrt{ab} \text{ and then } J_{Bha} = -\ln J \qquad (7)$$

Kullback-Liebler

$$f(a, b) = (a - b)(\ln a - \ln b) \text{ and then } J_{KL} = J \qquad (8)$$

Kolmogorov

$$f(a,b) = |a-b| \text{ and then } J_{Kol} = J \tag{9}$$

Matusita

$$f(a,b) = (\sqrt{a}-\sqrt{b})^2 \text{ and then } J_{Mat} = \sqrt{J} \tag{10}$$

Patrick-Fisher

$$f(a,b) = (a-b)^2 \text{ and then } J_{PF} = \sqrt{J} \tag{11}$$

These measures satisfy the previous conditions and can be used in a weighed form, taking into account the prior class probabilities $P(\omega_i)$ so that $f[p(\vec{x}|\omega_1), p(\vec{x}|\omega_2)]$ becomes $f[p(\vec{x}|\omega_1)P(\omega_1), p(\vec{x}|\omega_2)P(\omega_2)]$. They can also be related to P_e in the form of upper-bounds [18, 7].

Dependence: These measures quantify how strongly two features are associated with one another, in the sense that knowing the value of one it is possible to predict the value of the other. In the context of feature selection, a feature is better evaluated the better it predicts the class. The correlation coefficient is a classical measure that still finds application [22]. A somewhat different approach is to estimate the divergence between the class-conditional and the unconditional densities. Any unweighed probabilistic distance measure serves this purpose. Specifically, we have measures of the form $f[p(\vec{x}|\omega_i), p(\vec{x})], i = 1, 2$.

Interclass distance: These measures are based on the assumption that instances of a different class are distant in the instance space. It is enough then to define a metric between classes and use it as measure:

$$D(\omega_i, \omega_j) = \frac{1}{N_i N_j} \sum_{k_1}^{N_i} \sum_{k_2=k_1+1}^{N_j} d(x_{(i,k_1)}, x_{(j,k_2)}) \tag{12}$$

$$J = \sum_{i=1}^{m} P(\omega_i) \sum_{j=i+1}^{m} P(\omega_j) D(\omega_i, \omega_j) \tag{13}$$

being $x_{(i,j)}$ the instance j of class ω_i, and N_i the number of instances of the class ω_i. The most usual distances d belong to the Euclidean family. These measures do not require the modeling of any density function, but their relation to the probability of error can be very loose.

Information or Uncertainty: Similarly to the probabilistic dependence, we may observe $\vec{x}$ and compute the a posteriori probabilities $P(\omega_i|\vec{x})$ to determine how much information on the class of $\vec{x}$ has been gained, with respect to its prior probability. If all the classes become roughly equally probable, then the information gain is minimal and the uncertainty (entropy) is maximum.

Many measures can then be derived that make use of $p(\vec{x})$ and the set $\{P(\omega_i|\vec{x}), i = 1, \ldots, n\}$. For instance, using Shannon's entropy, we have:

$$J_{Sha} = -\int p(\vec{x}) \sum_{i=1}^{m} P(\omega_i|\vec{x}) log_2 P(\omega_i|\vec{x}) d\vec{x} \tag{14}$$

Measures derived from generalizations of Shannon's entropy –as *Renyi's entropy* and the entropy of *degree* α– are discussed in [7].

Entropy can also be used without knowledge of the densities as is done in the induction of decision trees [46], where the *information gain* is typically computed independently for each feature in the induction process. Also, the notion of entropy-based relevance is heuristically used in [55] as $J(X') = r(C; X')$ (see Definition 8).

Consistency: An *inconsistency* in X' and S is defined as two instances in S that are equal when considering only the features in X' and that belong to different classes. The aim is thus to find the *minimum* subset of features leading to zero inconsistencies [4]. The *inconsistency count* of an instance $A \in S$ is defined as [32]:

$$IC_{X'}(A) = X'(A) - \max_k X'_k(A) \tag{15}$$

where $X'(A)$ is the number of instances in S equal to A using only the features in X' and $X'_k(A)$ is the number of instances in S of class k equal to A using only the features in X'. The *inconsistency rate* of a feature subset in a sample S is then:

$$IR(X') = \frac{\sum_{A \in S} IC_{X'}(A)}{|S|} \tag{16}$$

This is a monotonic measure, in the sense

$$X_1 \subset X_2 \Rightarrow IR(X_1) \geq IR(X_2)$$

A possible evaluation measure is then $J(X') = \frac{1}{IR(X')+1}$. This measure is in $[0, 1]$ and can be evaluated in $O(|S|)$ time using a hash table [32].

3.3 General Schemes for Feature Selection

The relationship between a FSA and the inducer chosen to evaluate the usefulness of the feature selection process can take three main forms: *embedded*, *filter* and *wrapper*.

Embedded Scheme: The inducer has its own FSA (either explicit or implicit). The methods to induce logical conjunctions[54, 56] provide an example of this embedding. Other traditional machine learning tools like decision trees or artificial neural networks are included in this scheme[38].

Filter Scheme: If the feature selection process takes place before the induction step, the former can be seen as a filter of non-useful features prior to induction. In a general sense it can be seen as a particular case of the embedded scheme in which feature selection is used as a pre-processing. The filter schemes are independent of the induction algorithm.

Wrapper Scheme: In this scheme the relationship is taken the other way around: it is the FSA that uses the learning algorithm as a subroutine [24]. The general argument in favor of this scheme is to equal the bias of both the FSA and the learning algorithm that will be used later on to assess the goodness of the solution. The main disadvantage is the computational burden that comes from calling the induction algorithm to evaluate each subset of considered features.

3.4 General Algorithm for Feature Selection

An abstract algorithm for feature selection that shows in a unified form the behavior of any FSA is depicted in Fig. 3.

In particular, being L a (weighed) list of weighed subsets of features (i.e. states), L maintains the ordered set of solutions in course. Exponential algorithms are typically characterized by $|L| \geq 1$ (examples would be BRANCH AND BOUND [42] or A^* [44]). The presence in the list is a function of the evaluation measure and defines the expansion order. Heuristic search algorithms also maintain this list (of open nodes), and the weighting is the value of the heuristic. Random search methods as Evolutionary Algorithms [5] are characterized by $|L| \geq 1$ (the list is the population and the weighting is the fitness value of the individuals). Sequential algorithms maintain $|L| = 1$, though there are exceptions (e.g., a bidirectional algorithm [19] would use $|L| = 2$). The second weighting (on the features of each solution subset) allows to include the two types of FSA according to their outcome (see §1).

The initial list L is in general built out of the original set of features and the algorithm maintains the best solution at all times ($Solution$). At each step, a FSA with a given search organization manipulates the list in a specific way and calls its mechanism for the generation of successors which in turn uses J. The result is an updated list and the eventual update of the best solution found so far. Notice that the data sample S is considered global to the algorithm.

3.5 Space of Characteristics of a FSA

All FSA can be represented in a space of characteristics according to the criteria of: search organization (`Org`), generation of successor states (`GS`) and evaluation measures (`J`) (Fig. 4), in accordance with the description in §3.2. This space <`Org`, `GS`, `J`> encompasses the whole spectrum of possibilities for a FSA. New proposals for evaluation measures (not expressible as a combination of the already existent) would extend the vertical axis.

```
Input:
    $S$ – data sample with features $X, |X| = n$
    $J$ – evaluation measure to be maximized
    $GS$ – successor generation operator
Output:
    $Solution$ – (weighed) feature subset

$L :=$ Start_Point($X$);
$Solution :=$ { best of $L$ according to $J$ };
repeat
    $L :=$ Search_Strategy ($L, GS(J), X$);
    $X' :=$ { best of $L$ according to $J$ };
    if $J(X') \geq J(Solution)$ or $(J(X') = J(Solution)$
          and $|X'| < |Solution|)$
    then $Solution := X'$;
until Stop($J, L$)
```

Figure 3. General Algorithm for Feature Selection.

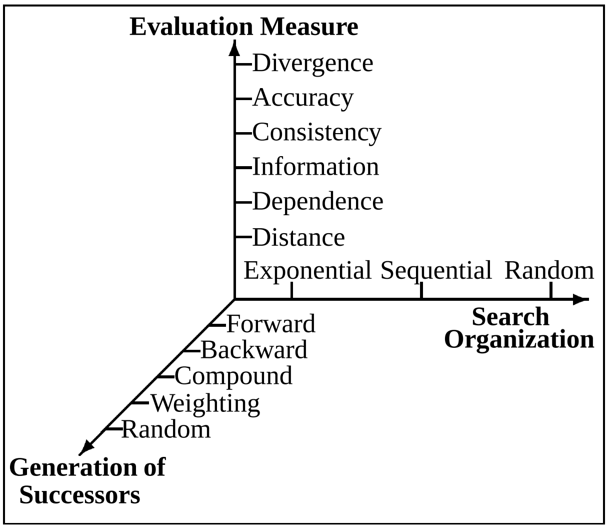


Figure 4. Characterization of a FSA.

Notice that the search organization and the generation of successor states are coordinates whose elements are, in principle, exclusive of each other, in the sense that a certain FSA will use only one combination organization/generation. We then speak of a *hybrid* FSA when it requires more than a point in the same coordinate to be characterized. This is unusual in the literature, although recent works seem to point in this direction [16, 8].

On the contrary, it is very feasible to *combine* several evaluation measures in a single FSA. Moreover, a measure could be regarded as belonging to more than one category (e.g., the divergence, dependence and information measures are very interrelated). In this vain, the vertical axis categorization should be regarded as conceptual (that is to say, what is measured more than how it is measured).

Non-classical algorithms (in the context of feature selection), can also be located in this space. For example, evolutionary approaches with fitness J correspond to <*random*, *random*, *any*> and artificial neural networks to <*sequential*, *weighting*, *accuracy*>. An extensive bibliographical revision of FSAs is shown in Table 1.

Algorithm	SO	GS	J	Ref.
ABB	E	B	Consistency	[33]
B&B	E	B	Consistency	[42]
BDS	S	F/B	Accuracy	[19]
Beam	E	F	any	[1]
BFF	E	F	Distance	[57]
Bobro	E	B	Distance	[10]
BSE	S	F/B	Information/Accuracy	[13]
Cardie	E	F	Information/Accuracy	[12]
CFS	E	F	Dependence	[22]
DTM	S	B	Information	[11]
Focus/-2	E	F	Consistency	[2]
GA	R	R	Accuracy	[53]
K2-AS	S	F	Probability/Accuracy	[50]
Koller	S	B	Information	[27]
LVF	R	R	Consistency	[34]
LVI	R	R	Consistency	[36]
LVW	R	R	Accuracy	[35]
MDLM	E	B	Informacion	[49]
MIFES-1	E	B	Consistency	[43]
Oblivion	E	B	Distance/Accuracy	[30]
POE-ACC	S	F	Dependence	[41]
PQSS	S	C	Accuracy	[19]
Preset	S	W	Dependence	[39]
QBB	R/E	R/B	Consistency	[16]
Race	S	F/B	Probability/Accuracy	[40]
RC	S	B	Accuracy	[20]
Relief	R	W	Distance	[25]
RGSS	R	F/B	Accuracy	[19]
RMHC-PF1	R	R	Distance/Accuracy	[51]
SA	R	R	Accuracy	[19]
SBG	S	B	any	[18]
SBS	S	B	Distance	[37]
SBS-W	S	B	Accuracy	[18]
SBS-SLASH	S	B	Accuracy	[13]
Schlimmer	E	F	Consistency	[47]
Segen	S	F	Distance	[48]
SFG	S	F	any	[18]
SFFS	E	C	any	[45]
SFBS	E	C	any	[45]
Winnow	S	W	Consistency	[31]
W-SBG	S	B	Accuracy	[26]
W-SFG	S	F	Accuracy	[26]

Table 1. Space of Characteristics of some FSA. Key: SO = Search Organization (E = Exponential, S = Sequential, R = Random), GS = Generation of Successors (F = Forward, B = Backward, C = Compound, W = Weighting, R = Random), J = Evaluation Measure.

4 Description of fundamental FSAs

In this section several of the currently most widespread FSA in machine learning are briefly described and commented on. In the following let us assume again that the evaluation measure is to be maximized.

4.1 LVF Algorithm

LVF (Las Vegas Filter) [34] (<*random, random, any*>) repeatedly generates random feature subsets and then computes their evaluation measure. It was originally implemented with consistency of the sample as evaluation measure. The algorithm is described in Fig. 5.

```
Input:
  $max$ − the maximum number of iterations
  $J$ − evaluation measure
  $S(X)$ − a sample $S$ described by $X$, $|X| = n$
Output:
  $L$ − all equivalent solutions found

$L$ := []          // L stores equally good sets
$Best$ := $X$      // Initialize best solution
$J_0$ := $J(S(X))$ // minimum allowed value of J
repeat $max$ times
  $X'$ := Random_SubSet($Best$) // $|X'| \leq |Best|$
  if $J(S(X')) \geq J_0$ then
    if $|X'| < |Best|$ then
      $Best$ := $X'$
      $L$ := [$X'$]          // L is reinitialized
    else if $|X'| = |Best|$ then
      $L$ := append($L, X'$)
    end
  end
end
end
```

Figure 5. LVF **(Las Vegas Filter Algorithm).**

LVW (Las Vegas Wrapper) [35] is a *wrapper* algorithm that uses LVF to generate candidate subsets and accuracy of an inducer as the evaluation measure.

4.2 LVI Algorithm

LVI (Las Vegas Incremental) [36] (<*random, random, consistency*>) is based on the grounds that it is not necessary to use the whole sample S in order to evaluate the measure J. The algorithm departs from a portion S_0 of S; if LVF finds a sufficiently good solution in S_0 then LVI halts. Otherwise the set of samples in $S \setminus S_0$ making S_0 inconsistent is added to S_0, this new portion is handed over to LVF and the process is iterated. Actually the evaluation measure could be any. The algorithm is described in Fig. 6.

```
Input:
  $max$ - the maximum number of iterations
  $J$ - evaluation measure
  $S(X)$ - a sample $S$ described by $X, |X| = n$
  $p$ - initial percentage
Output:
  $X'$ - solution found

$S_0$ := portion($S, p$)   // Initial portion
$S_1$ := $S \setminus S_0$          // Test set
$J_0$ := $J(S(X))$         // Minimum allowed value of J
repeat forever
    $X'$ := LVF ($max, J, S_0(X)$)
    if $J(S(X')) \geq J_0$ then stop
    else
        $C$ := {elements in $S_1$ with low
                contribution to $J$ using $X'$}
        $S_0$ := $S_0 \cup C$
        $S_1$ := $S_1 \setminus C$
    end
end
```

Figure 6. LVI **(Las Vegas Incremental Algorithm).**

Intuitively, the portion can be neither too small nor too big. If it is too small, after the first iteration many inconsistencies will be found and added to the current subsample, which will hence be very similar to S. If it is too big, the computational savings will be modest. The authors suggest $p = 10\%$ or a value proportional to the number of features.

4.3 Relief Algorithm

RELIEF [25] (<*random, weighting, distance*>) chooses randomly an instance A of S and determines its *near hit* and its *near miss* in relation to S. The former is the closest instance to A among all the instances in the same class of A. The latter is the closest instance to A among all the instances in a different class. The underlying idea is that a feature is more relevant to A the more it separates A and its near miss, and the least it separates A and its near hit. The result is a weighed version of the original feature set. The basic algorithm is described in Fig. 7.

An improved version [28] is proposed (RELIEF-F) where the k more similar instances are selected (belonging to the same or different class, respectively) and their averages are computed.

```
Input:
  $p$ - sampling percentage
  $d$ - distance measure
  $S(X)$ - a sample $S$ described by $X, |X| = n$
Output:
  $W$ - array of feature weights

initialize $W$[] to zero
do $p|S|$ times
   $A$ := Random_Instance ($S$)
   $A_{nh}$ := Near-Hit ($A, S$)
   $A_{nm}$ := Near-Miss ($A, S$)
   for each $i \in [1..n]$ do
      $W[i]$ := $W[i] + d_i(A, A_{nm}) - d_i(A, A_{nh})$
   end
end
```

Figure 7. RELIEF **Algorithm.**

4.4 SFG/SBG Algorithms

SFG (SEQUENTIAL FORWARD GENERATION) (<*sequential, forward, any*>) iteratively adds features to a initial subset, in such a way that improves a given measure J taking into account those features already present in the solution. Additionally, an ordered list can also be obtained. SBG (SEQUENTIAL BACKWARD GENERATION) (<*sequential, backward, any*>) is the backward counterpart. The algorithms are jointly described in Fig. 8.

```
Input:
  $S(X)$ - a sample $S$ described by $X, |X| = n$
  $J$ - evaluation measure
Output:
  $X'$ - solution found

$X' := \emptyset$  // forward
$X' := X$  // backward
repeat
 $x'$ := $argmax\{J(S(X' \cup \{x\})) \mid x \in X \setminus X'\}$ // forward
 $x'$ := $argmax\{J(S(X' \setminus \{x\})) \mid x \in X'\}$  // backward
 $X' := X' \cup \{x'\}$  // forward
 $X' := X' \setminus \{x'\}$  // backward
until no improvement in $J$ in last $j$ steps
                          or   $X' = X$  // forward
                          or   $X' = \emptyset$  // backward
```

Figure 8. SBG/SFG **(Sequential Backward /Forward Generation Algorithms).**

The algorithms W-SFG and W-SBG (*w* for *wrapper*) use the accuracy of an external inducer as evaluation measure.

4.5 SFFS Algorithm

SFFS (SEQUENTIAL FLOATING FORWARD SEARCH) [45] (<*exponential, compound, any*>) is an exponential cost algorithm that operates in a sequential flavor. In each selection step SFFS performs a forward step followed by a variable number (possibly null) of backward ones. In essence, a feature is first unconditionally added and then features are removed as long as the generated subsets are the best among their respective size. The algorithm is so-called because it has the characteristic of *floating* around a potentially good solution of the specified size (see Fig. 9). The backward counterpart performs a backward step followed by a variable number (possibly null) of forward ones.

4.6 Focus Algorithm

The basic FOCUS [2] (<*exponential, forward, consistency*>) algorithm starts evaluating each singleton feature set, then each set of two features and so forth. It halts whenever a sufficiently consistent solution is found. The basic algorithm is described in Fig. 10.

The FOCUS-2 [3] algorithm introduces the concept of *conflict* between positive and negative examples to prune the search.

4.7 B&B Algorithm

B&B (BRANCH & BOUND) [42] (<*exponential, backward, any monotonic*>) is an optimal search algorithm. Given a threshold β (specified by the user), the search stops at each node the evaluation of which is lower than β, so that efferent branches are pruned.

ABB (AUTOMATIC BRANCH & BOUND) [33] (<*exponential, backward, any monotonic*>) is a variant of B&B in which the threshold is automatically set. This algorithm is described in Fig. 11.

4.8 QBB Algorithm

QBB (QUICK BRANCH AND BOUND) [16] (<*random/-exponential, random/backward, consistency*>) is a hybrid algorithm composed of LVF and ABB (see §4.1 and §4.7). The basic idea consists of using LVF to find good starting points for ABB. It is expected that ABB can explore the remaining search space efficiently. The algorithm is described in Fig. 12. The authors [16] reported that QBB can be, in general, more efficient than LVF, FOCUS and ABB in terms of average execution time and selected solution.

```
Input:
  $S(X)$ – a sample $S$ described by $X, |X| = n$
  $J$ – evaluation measure
  $d$ – desired size of the solution
  $D$ – maximum deviation allowed with
      respect to $d$
Output:
    solution of size $d \pm D$
```

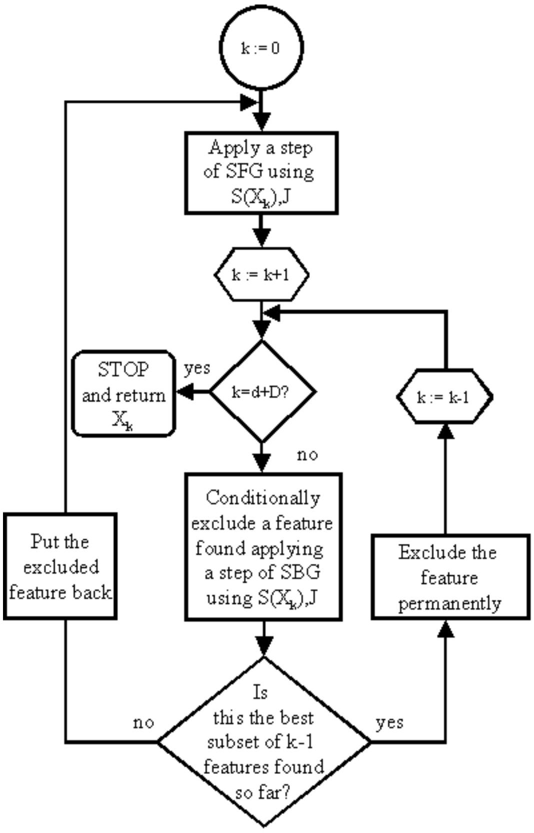

k := 0
Apply a step of SFG using S(X_k),J
k := k+1
k=d+D?
yes
STOP and return X_k
no
Conditionally exclude a feature found applying a step of SBG using S(X_k),J
Put the excluded feature back
Is this the best subset of k-1 features found so far?
no
yes
Exclude the feature permanently
k := k-1

Figure 9. SFFS **Sequential Floating Forward Search Algorithm. The set X_k denotes the current solution of size k.**

5 Empirical Evaluation of FSAs

The first question arising in relation to a feature selection experimental design is: what are the aspects that we would like to evaluate of a FSA solution in a given data set? In this study we decided to evaluate FSA performance with respect to four particularities: relevance, irrelevance, redundance and sample size. To this end, several fundamental FSAs are studied to assess their performance on synthetic data sets with known relevant features. Then sample data sets of different sizes are corrupted with irrelevant and/or redundant features. The experiments are designed to test the *endurance* of different FSAs (e.g., behaviour against the ratio number-of-irrelevant vs. number-of-relevant features).

```
Input:
  $S(X)$ - a sample $S$ described by $X, |X| = n$
  $J$ - evaluation measure (consistency)
  $J_0$ - minimum allowed value of $J$
Output:
  $X'$ - solution found

for $i \in [1..n]$ do
    for each $X' \subset X$ , with $|X'| = i$ do
        if $J(S(X')) \geq J_0$ then stop
    end
end
```

Figure 10. FOCUS **Algorithm.**

```
Input:
  $S(X)$ - a sample $S$ described by $X, |X| = n$
  $J$ - evaluation measure (monotonic)
Output:
  $L$ - all equivalent solutions found

procedure ABB ($S(X)$: sample; var $L'$: list
    of set)
  for each $x$ in $X$ do
    enqueue($Q, X \setminus \{x\}$)  // remove a feature at a time
  end
  while not empty($Q$) do
    $X' :=$ dequeue($Q$)
    // $X'$ is legitimate if it is not a subset of a pruned state
    if legitimate($X'$) and $J(S(X')) \geq J_0$ then
       $L' :=$ append($L', X'$)
       ABB($S(X'), L'$)
    end
  end
end

begin
   $Q := \emptyset$              // Queue of pending states
   $L' := [X]$            // List of solutions
   $J_0 := J(S(X))$   // Minimum allowed value of J
   ABB  ($S(X), L'$)  // Initial call to ABB
   $k :=$ smallest size of a subset in $L'$
   $L :=$ set of elements of $L'$ of size $k$
end
```

Figure 11. ABB **(Automatic Branch and Bound Algorithm).**

5.1 Particularities to be evaluated

Relevance: Different families of problems are generated by varying the number of relevant features N_R. These are features that, by construction, have an influence on the output and whose role can not be assumed by the rest (i.e., there is no redundance).

```
Input:
  $max$ - the maximum number of iterations
  $J$ - monotonic evaluation measure
  $S(X)$ - a sample $S$ described by $X, |X| = n$
Output:
  $L$ - all equivalent solutions found

$L\_ABB$ := []
$L\_LVF$ := LVF ($max, J, S(X)$)
for each $X' \in L\_LVF$ do
   $L\_ABB$ := concat($L\_ABB, ABB(S(X'), J)$)
end
$k$ := smallest size of a subset in $L\_ABB$
$L$ := set of elements of $L\_ABB$ of size $k$
```

Figure 12. QBB **(Quick Branch and Bound Algorithm).**

Irrelevance: Irrelevant features are defined as those features not having any influence on the output, and whose values are generated at random for each example. For a problem with N_R relevant features, different numbers of irrelevant features N_I are added to the corresponding data sets (thus providing with several subproblems for each choice of N_R).

Redundance: In these experiments, a redundance exists whenever a feature can take the role of another (perhaps the simplest way to model redundance). This is obtained by choosing a relevant feature randomly and replicating it in the data set. For a problem with N_R relevant features, different numbers of redundant features $N_{R'}$ are added in a way analogous to the generation of irrelevant features.

Sample Size: It refers to the number of instances $|S|$ of a data sample S. In these experiments, $|S|$ is defined as $|S| = \alpha k N_T c$, where α is a constant, k is a multiplying factor, N_T is the total number of features ($N_R + N_I + N_{R'}$) and c is the number of classes of the problem. Note this means that the sample size will depend linearly on the total number of features.

5.2 Evaluation of Performance

The *score* criterion expresses the degree to which a solution obtained by a FSA matches the correct solution. This criterion behaves as a similarity $s(x, y) : X \times X \to [0, 1]$ in the classical sense [15], satisfying:

1. $s(x, y) = 1 \Longleftrightarrow x = y$
2. $s(x, y) = s(y, x)$

where $s(x,y) > s(x,z)$ indicates that y is more similar to x than z.

Let us denote by X the total set of features, partitioned in $X = X_R \cup X_I \cup X_{R'}$, being $X_R, X_I, X_{R'}$ the subsets of relevant, irrelevant and redundant features of X, respectively and call $X^* \subseteq X$ the ideal solution. Let us denote by $\mathcal{A}$ the feature subset selected by a FSA. The idea is to check how much $\mathcal{A}$ and X^* have in common. Let us define $\mathcal{A}_R = X_R \cap \mathcal{A}$, $\mathcal{A}_I = X_I \cap \mathcal{A}$ and $\mathcal{A}_{R'} = X_{R'} \cap \mathcal{A}$. In general, we have $\mathcal{A}_T = X_T \cap \mathcal{A}$ (hereafter T stands for a subindex in $\{R, I, R'\}$). Since necessarily $\mathcal{A} \subseteq X$, we have $\mathcal{A} = \mathcal{A}_R \cup \mathcal{A}_I \cup \mathcal{A}_{R'}$. The *score* $S_X(\mathcal{A}) : P(X) \to [0,1]$ will fulfill the following conditions:

- $S_X(\mathcal{A}) = 0 \Longleftrightarrow \mathcal{A} = X_I$
- $S_X(\mathcal{A}) = 1 \Longleftrightarrow \mathcal{A} = X^*$
- $S_X(\mathcal{A}) > S_X(\mathcal{A}')$ indicates that $\mathcal{A}$ is more similar to X^* than $\mathcal{A}'$.

The score is defined in terms of the similarity in that for all $\mathcal{A} \subseteq X, S_X(\mathcal{A}) = s(\mathcal{A}, X^*)$. This scoring measure will also be parameterized, so that it can ponder each type of divergence (in relevance, irrelevance and redundance) to the optimal solution. The set of parameters is expressed as $\alpha = \{\alpha_R, \alpha_I, \alpha_{R'}\}$ with $\alpha_T \geq 0$ and $\sum \alpha_T = 1$.

Intuitive Description

The criterion $S_X(\mathcal{A})$ penalizes three situations:

1. There are relevant features lacking in $\mathcal{A}$ (the solution is incomplete).
2. There are more than enough relevant features in $\mathcal{A}$ (the solution is redundant).
3. There are some irrelevant features in $\mathcal{A}$ (the solution is incorrect).

An order of importance and a weight will be assigned (via the α_T parameters), to each of these situations.

Formal Description

The precedent point (3.) is simple to model: if suffices to check whether $|\mathcal{A}_I| > 0$, being $\mathcal{A}$ the solution of the FSA. Relevance and redundance are strongly related given that, in this context, a feature is redundant or not depending on what other relevant features are present in $\mathcal{A}$.

Notice then that the optimal solution X^* is not unique, though all them should be equally valid for the *score*. To this end, the features are broken down in *equivalence classes*, where elements of the same class are redundant to each other (i.e., any optimal solution must comprise only one feature of each equivalence class).

Being $\mathcal{A}$ a feature set, we define a binary relation between two features $x_i, x_j \in \mathcal{A}$ as: $x_i \sim x_j \Longleftrightarrow x_i$ and x_j represent the same information. Clearly $\sim$ is an equivalence relation. Let $\mathcal{A}^{\sim}$ be the quotient set of $\mathcal{A}$ under $\sim$, $\mathcal{A}^{\sim} = \{[x] \mid x \in \mathcal{A}\}$, any optimal solution $\mathcal{A}^*$ will satisfy:

1. $|\mathcal{A}^*| = |X_R|$
2. $\forall\, [x_i] \in \mathcal{A}^{\sim} : \exists x_j \in [x_i] : x_j \in \mathcal{A}^*$

We denote by $\mathcal{A}^*$ *any* of these solutions.

Construction of the *score*

In the present case, the set to be split in equivalence classes is formed by all the relevant features (redundant or not) chosen by a FSA. We define then:

$$\mathcal{A}_R^{\sim} = (\mathcal{A}_R \cup \mathcal{A}_{R'})^{\sim}$$

(equivalence classes in which the relevant features chosen by a FSA are split)

$$X_R^{\sim} = (X_R \cup X_{R'})^{\sim}$$

(equivalence classes in which the original set of features is split)

Let $\mathcal{A}_R^{\sim} \uplus X_R^{\sim} = \{[x_i] \in X_R^{\sim} \mid \exists [x_j] \in \mathcal{A}_R^{\sim} : [x_j] \subseteq [x_i]\}$ and define, for Q quotient set:

$$F(Q) = \sum_{[x] \in Q} (|x| - 1)$$

The idea is to express the *quotient* between the number of redundant features chosen by the FSA and the number it could have chosen, given the relevant features present in its solution. In the precedent notation, this is written (provided the denominator is not null):

$$\frac{F(\mathcal{A}_R^{\sim})}{F(\mathcal{A}_R^{\sim} \uplus X_R^{\sim})}$$

Let us finally build the *score*, formed by three terms: relevance, irrelevance and redundance. Defining:

$$I = 1 - \frac{|\mathcal{A}_I|}{|X_I|}, \qquad R = \frac{|\mathcal{A}_R^{\sim}|}{|X_R|}, \text{ with } \mathcal{A}_R^{\sim} = (\mathcal{A}_R \cup \mathcal{A}_{R'})^{\sim}$$

$$R' = \begin{cases} 0 & \text{if } F(\mathcal{A}_R^{\sim} \uplus X_R^{\sim}) = 0 \\ \frac{1}{|X_{R'}|}\left(1 - \frac{F(\mathcal{A}_R^{\sim})}{F(\mathcal{A}_R^{\sim} \uplus X_R^{\sim})}\right) & \text{otherwise.} \end{cases}$$

for any $\mathcal{A} \subseteq X$ the score is defined as $S_X(\mathcal{A}) = \alpha_R R + \alpha_{R'} R' + \alpha_I I$.

Restrictions on the α_T

We can establish now the desired restrictions on the behavior of the score. From the more to the less severe: there are relevant features lacking, there are irrelevant features, and there is redundancy in the solution. This is reflected in the following conditions on the α_T:

1. Choosing an irrelevant feature is better than missing a relevant one: $\frac{\alpha_R}{|X_R|} > \frac{\alpha_I}{|X_I|}$

2. Choosing a redundant feature is better than choosing an irrelevant one: $\frac{\alpha_I}{|X_I|} > \frac{\alpha_{R'}}{|X_{R'}|}$

We also define $\alpha_T = 0$ if $|X_T| = 0$. Notice that the denominators are important for, as an example, expressing the fact that it is not the same choosing an irrelevant feature when there were only two that when there were three (in the latter case, there is an irrelevant feature that could have been chosen when it was not).

Practical Considerations

In order to translate the previous inequalities into workable conditions, a parameter $\epsilon \in (0,1]$ can be introduced to express the exact relation between the α_T. Let $\underline{\alpha}_T = \frac{\alpha_T}{|A_T|}$. The following two equations have to be satisfied:

$$\beta_R \underline{\alpha}_R = \underline{\alpha}_I, \qquad \beta_I \underline{\alpha}_I = \underline{\alpha}_{R'}$$

for suitable chosen values of β_R and β_I. In this work we take $\beta_R = \epsilon/2$ and $\beta_I = 2\epsilon/3$. This means that, at equal $|A_R|, |A_I|, |A_{R'}|$, α_R is at least twice more important than α_I (because of the $\epsilon/2$) and α_I is at least one and a half times more important than $\alpha_{R'}$. Specifically, the minimum values are attained for $\epsilon = 1$ (i.e., α_R counts twice α_I). For $\epsilon < 1$ the differences widen proportionally to the point that, for $\epsilon \approx 0$, only $\alpha_R R$ will count on the overall score.

6 Experimental Evaluation

In this section we detail the experimental methodology and quantify the various parameters of the experiments. The basic idea consists on generating sample data sets with known particularities (synthetic functions f) and hand them over to the different FSAs to obtained a hypothesis H. The divergence between the defined function and the obtained hypothesis will be evaluated by the *score* criterion. This experimental design is illustrated in Fig. 13.

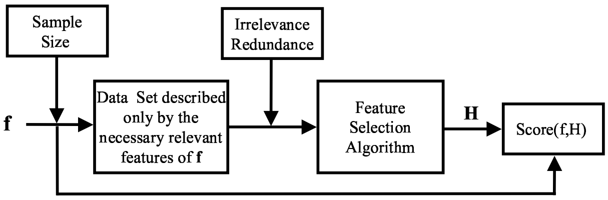

Sample Size
Irrelevance Redundance
f
Data Set described only by the necessary relevant features of **f**
Feature Selection Algorithm
H
Score(f,H)

Figure 13. FlowChart of Experimental Design.

6.1 Description of the FSAs used

The ten FSAs used in the experiments were : E-SFG, QBB, LVF, LVI, C-SBG, RELIEF, SFBG, SFFG, W-SBG, and W-SFG (see Table 2). The algorithms E-SFG, W-SFG are versions of SFG using entropy and the accuracy of a C4.5 inducer, respectively. The algorithms C-SBG, W-SBG are versions of SBG using consistency and the accuracy of a C4.5 inducer, respectively. During the course of the experiments the algorithms FOCUS, B&B, ABB and LVW were put aside due to their unaffordable consumption of resources.

Algorithm	Search Organization	Generation of Succesors	Evaluation Measure
LVF	Random	Random	Consistency
LVI	Random	Random	Consistency
QBB	Random/Expon.	Random/Backward	Consistency
RELIEF	Random	Weighting	Distance
C-SBG	Sequential	Backward	Consistency
E-SFG	Sequential	Forward	Entropy
SFBG	Exponential	Compound	Consistency
SFFG	Exponential	Compound	Consistency
W-SBG	Sequential	Backward	Accuracy(C4.5)
W-SFG	Sequential	Forward	Accuracy(C4.5)

Table 2. FSAs used in the experiments.

6.2 Modifications to the FSA

For purposes of comparison, some modifications were performed to the FSAs, without affecting the nucleus of each algorithm. On the other hand, a filtering criterion was established to binarize the outputs of the algorithms that give a lineal order of features.

Resource: We consider that all the FSAs should have approximately the same opportunities to compete, in what regards the computational resources. This means the exponential algorithms can be finished before its natural stopping condition. In our case, this only happens to the QBB algorithm, which may be forced to give the best solution obtained until that moment. For the case of LVI, it should be pointed out that only 50% (on average) of the data set is sampled, so that double resources are assigned.

Filtering Criterion: Since RELIEF and E-SFG give as output an ordered list of features x_i according to their weight w_i, a filtering criterion is necessary to transform this solution to a subset of features. The procedure used here is simple: since the interest is in determining a good cut point, first those w_i further than two variances from the mean are discarded (that is to say, with very high or very low weights). Then define $s_i = w_i + w_{i-1}$ and $\sigma_j = \sum_{i=2}^{j} s_i$. The objective is to search for the feature x_j such that:

$$1 - \frac{\sigma_j}{\sigma_n}\frac{n-j}{n} \quad \text{is maximum.}$$

The cut point is then set between x_j and x_{j+1}.

6.3 Implementations of Data Families

A total of twelve families of data sets were generated studying three different problems and four instances of each, by varying the number of relevant features N_R. Let $x_1, \ldots, x_n$ be the relevant features of a problem f. The selected problems are:

Parity: This is the classic binary problem of parity n, where the output is $f(x_1, \cdots, x_n) = 1$ if the number of $x_i = 1$ is odd and $f(x_1, \cdots, x_n) = 0$ otherwise.

Disjunction: A disjunctive task, with $f(x_1, \cdots, x_n) = 1$ if $(x_1 \wedge \cdots \wedge x_{n'}) \vee (x_{n'+1} \wedge \cdots \wedge x_n)$, with $n' = n\, div\, 2$ if n is even and $n' = (n\, div\, 2) + 1$ if n is odd.

GMonks: This problem is a generalization of the classic *monks* problems [52]. In its original version, three independent problems were applied on sets of $n = 6$ features that take values of a discrete, finite and unordered set (nominal features). Here we have grouped the three problems in a single one computed on *each* segment of 6 features. Let n be multiple of 6, $k = n\, div\, 6$ and $b = 6(k' - 1) + 1$, for $1 \leq k' \leq k$. Let us denote for "1" the first value of a feature, for "2" the second, etc. The problems are the following:

1. $P1 : (x_b = x_{b+1}) \vee x_{b+4} = 1$
2. $P2$: two or more $x_i = 1$ in $x_b \cdots x_{b+5}$
3. $P3 : (x_{b+4} = 3 \wedge x_{b+3} = 1) \vee (x_{b+4} \neq 3 \wedge x_{b+1} \neq 2)$

For each segment, the boolean condition $P2 \wedge \neg(P1 \wedge P3)$ is checked. If this condition is satisfied for s or more segments with $s = n_s\, div\, 2$ (being n_s the number of segments) the function *Gmonks* is 1; otherwise, it is 0.

6.4 Experimental Setup

The experiments were divided in three groups. The first group refers to the relationship between irrelevance vs. relevance. The second refers to the relationship between redundance vs. relevance. The last group refers to sample size. Each group uses three families of problems (*Parity*, *Disjunction* and *GMonks*) with four different instances for each problem, varying the number of relevant features N_R.

Relevance: The different numbers N_R vary for each problem, as follows: {4, 8, 16, 32} (for *Parity*), {5, 10, 15, 20} (for *Disjunction*) and {6, 12, 18, 24} (for *GMonks*).

Irrelevance: In these experiments, we have N_I running from 0 to 2 times the value of N_R, in intervals of 0.2 (that is, eleven different experiments of irrelevance for each N_R).

Redundance: Similarly to the generation of irrelevant features, we have $N_{R'}$ running from 0 to 2 times the value of N_R, in intervals of 0.2.

Sample Size: Given the formula $|S| = \alpha k N_T c$ (see §5.1), different problems were generated considering $k \in \{0.25, 0.5, 0.75, 1.0, 1.25, 1.75, 2.0\}$, $N_T = N_R + N_I + N_{R'}$, $c = 2$ and $\alpha = 20$. The values of N_I and $N_{R'}$ were fixed as $N_I = N_{R'} = N_R\, div\, 2$.

6.5 Results

Due to space reasons, only a sample of the results are presented, in Fig. 14. In all the plots, each point represents the average of 10 independent runs with different random data samples. The Figs. 14(a) and (b) are examples of irrelevance vs. relevance for four instances of the problems, (c) and (d) are examples of redundance vs. relevance and (e) and (f) are examples of sample size experiments. In all cases, the horizontal axis represents the ratios between these particularities as explained above. The vertical axis represents the average results given by the score criterion.

- In Fig. 14(a) the C-SBG algorithm shows at first a good performance but clearly as the irrelevance ratio increases, it falls dramatically (below the 0.5 level from $N_I = N_R$ on). Note that for $N_R = 4$ performance is always perfect (the plot is on top of the graphic).
- In contrast, in Fig. 14(b) the RELIEF algorithm presents very similar and fairly good results for the four instances of the problem, being almost insensitive to the total number of features.
- In relation to redundance vs. relevance, in Fig. 14(c) the LVF algorithm presents a very good and stable performance for the different problem instances of *Parity*.
- In 14(d) we observe that QBB tends to a poor general performance in the *Disjunction* problem when the total number of features increases.
- The plots in Figs. 14(e) and (f) show additional interesting results because we can appreciate the curse of dimensionality effect [23]. In these figures, LVI and W-SFG present an increasingly poor performance (see the figure from top to bottom) with the number of features provided the number of examples is increasing in a linear way. However, in general, as long as more examples are added performance is better (see the figure from left to right).

A summary of the results is displayed in Fig. 15 for the ten algorithms, allowing for a comparison across all the sample datasets with respect to each studied particularity. Specifically, Figs. 15(a), (c) and (d) show the average score of each algorithm for irrelevance, redundance and sample size, respectively. Moreover, Figs. 15(b), (d) and (f) show the same average weighed by N_R, in such a way that more weight is assigned to more difficult problems.

In each graphic there are two keys: the key to the left shows the algorithms ordered by *total* average performance, from top to bottom. The key to the right shows the algorithms ordered by average performance on the *last* abscissa value, also from top to bottom. In other words, the left list is topped by the algorithm that wins on average, while the right list is topped by the algorithm that ends on the lead. This is also useful to help reading the graphics.

- Fig. 15(a) shows that RELIEF ends up on the lead of the irrelevance vs. relevance problems, while SFFG shows the best average performance. The algorithm W-SFG is also well positioned.
- Fig. 15(c) shows that the algorithms LVF and LVI together with C-SBG are the overall best. In fact, there is a bunch of algorithms that also includes the two *floating* and QBB showing a close performance. Note how RELIEF and the *wrappers* are very poor performers.
- Fig. 15(e) shows that the wrapper algorithms seem to be able to extract the most of the data when there is a shortage of it. Surprisingly, the backward wrapper is just fairly positioned on average. The forward floating algorithm is again quite good on average, together with C-SBG. However, all of the algorithms are quite close and show the same kind of dependency to the data. Note the general poor performance of E-SFG, due to the fact that it is the only algorithm that computes its evaluation measure (entropy in this case) independently for each feature.
- The weighed versions of the graphics do not seem to alter the picture very much. A closer look reveals that the differences between algorithms have widened. Very interesting is the change for RELIEF, that takes the lead both on irrelevance and sample size, but not on redundance.

7 Conclusions

The task of a feature selection algorithm (FSA) is to provide with a computational solution to the feature selection problem motivated by a certain definition of *relevance*. This algorithm should be reliable and efficient. The many FSAs proposed in the literature are based on quite different principles (as the evaluation measure used, the precise way to explore the search space, etc) and loosely follow different definitions of relevance.

In this work a way to evaluate FSAs was proposed in order to understand their general behaviour on the particularities of relevance, irrelevance, redundancy and sample size of synthetic data sets. To achieve this goal, a set of controlled experiments using artificially generated data sets were designed and carried out. The set of optimal solutions is then compared with the output given by the FSAs (the obtained hypotheses). To this end, a *scoring* measure was defined to express the degree of approximation of the FSA solution to the real solution. The final outcome of the experiments can be seen as an illustrative step towards gaining useful knowledge that enables to decide which algorithm to use in certain situations.

In this vein, it is shown the different behaviour of the algorithms to different data particularities and thus the danger in relying in a single algorithm. This points in the direction of using new hybrid algorithms or combinations thereof for a more reliable assessment of feature relevance.

As future activities, this work can be extended in many ways to carry up richer evaluations such as considering features strongly *correlated* with the class or with one another, noise in the data sets, other kinds of data (e.g., continuous data), missing values, and the use of combined evaluation measures.

Acknowledgements This work is supported by the Spanish CICyT Project TAP99-0747 and by the Mexican Petroleum Institute. We also wish to thank the anonymous reviewers for their valuable comments.

References

[1] D. W. Aha and R. L. Bankert. Feature Selection for Case–Based Classification of Cloud Types. In *Working Notes of the AAAI94, Workshop on Case–Based Reasoning*, pages 106–112, Seattle, WA, 1994. AAAI Press.

[2] H. Almuallim and T. G. Dietterich. Learning with Many Irrelevant Features. In *Proc. of the 9th National Conf. on Artificial Intelligence*, volume 2, pages 547–552, Anaheim, CA, 1991. AAAI Press.

[3] H. Almuallim and T. G. Dietterich. Efficient Algorithms for Identifying Relevant Features. In *Proc. of the 9th Canadian Conf. on Artificial Intelligence*, pages 38–45, Vancouver, BC, 1992. Morgan Kaufmann.

[4] H. Almuallim and T. G. Dietterich. Learning Boolean Concepts in the Presence of Many Irrelevant Features. *Artificial Intelligence*, 69(1–2):279–305, 1994.

[5] T. Back, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, Oxford, 1996.

[6] D. Bell and H. Wang. A Formalism for Relevance and its Application in Feature Subset Selection. *Machine Learning*, 41(2):175–195, 2000.

[7] M. Ben-Bassat. Use of Distance Measures, Information Measures and Error Bounds in Fuature Evaluation. In P. R. Krishnaiah and L. N. Kanal, editors, *Handbook of Statistics*, volume 2, pages 773–791, North Holland, 1982.

[8] J. Bins and B. Draper. Feature Selection from Huge Feature Sets. In *Int. Conf. on Computer Vision*, volume 2, pages 159–165, Vancouver, CA, 2001.

[9] A. L. Blum and P. Langley. Selection of Relevant Featu-res and Examples in Machine Learning. In R. Greiner and D. Subramanian, eds., *Artificial Intelligence on Relevance*, volume 97, pages 245–271. Artificial Intelligence, 1997.

[10] L. Bobrowski. Feature Selection Based on Some Homogeneity Coefficient. In *Proc. of 9th Int. Conf. on Pattern Recognition*, pages 544–546. IEEE Press, 1988.

[11] J. Callan, T. Fawcett, and E. Rissland. An Adaptive Approach to Case–Based Search. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence*, pages 803–808. Morgan Kaufmann, 1991.

[12] C. Cardie. Using Decision Trees to Improve Case–Based Learning. In *Proc. of the 10th Int. Conf. on Machine Learning*, pages 25–32, Amherst, MA, 1993. Morgan Kaufmann.

[13] R. A. Caruana and D. Freitag. Greedy Attribute Selection. In *Proc. of the 11th Int. Conf. on Machine Learning*, pages 28–36, New Brunswick, NJ, 1994. Morgan Kaufmann.

[14] R. A. Caruana and D. Freitag. How Useful is Relevance? Technical report, Fall'94 AAAI Symposium on Relevance, New Orleans, 1994.

[15] S. Chandon and L. Pinson. *Analyse Typologique.* Masson, 1981.

[16] M. Dash and H. Liu. Hybrid Search of Feature Subsets. In H. Y. Lee and H. Motoda, editors, *Proc. of the 15th Pacific Rim Int. Conf. on AI*, pages 22–27, Singapore, 1998. Springer Verlag.

[17] M. Dash, H. Liu, and H. Motoda. Consistency Based Feature Selection. In *Pacific–Asia Conf. on Knowledge Discovery and Data Mining*, pages 98–109, 2000.

[18] P. A. Devijver and J. Kittler. *Pattern Recognition – A Statistical Approach.* Prentice Hall, London, GB, 1982.

[19] J. Doak. An Evaluation of Feature Selection Methods and their Application to Computer Security. Technical Report CSE–92–18, Davis, CA: University of California, Department of Computer Science, 1992.

[20] P. Domingos. Context–Sensitive Feature Selection for Lazy Learners. *Artificial Intelligence Review*, 11:227–253, 1997.

[21] P. Gärdenfors. On the Logic of Relevance. *Synthese*, 37:351–367, 1978.

[22] M. A. Hall. *Correlation–based Feature Selection for Machine Learning.* PhD thesis, University of Waikato, 1999.

[23] A. K. Jain and D. Zongker. Feature Selection: Evaluation, Application, and Small Sample Performance. *Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.

[24] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant Features and the Subset Selection Problem. In *Proc. of the 11th Int. Conf. on Machine Learning*, pages 121–129, New Brunswick, NJ, 1994. Morgan Kaufmann.

[25] K. Kira and L. Rendell. A Practical Approach to Feature Selection. In *Proc. of the 9th Int. Conf. on Machine Learning*, pages 249–256, Aberdeen, Scotland, 1992. Morgan Kaufmann.

[26] R. Kohavi. *Wrapper for Performance Enhancement and Oblivious Decision Graphs.* PhD thesis, Stanford University, 1995.

[27] D. Koller and M. Sahami. Toward Optimal Feature Selection. In *Proc. of the 13th Int. Conf. on Machine Learning*, pages 284–292, Bari, IT, 1996. Morgan Kaufmann.

[28] I. Kononenko. Estimating Attributes: Analysis and Extensions of Relief. In *Proc. of the European Conf. on Machine Learning*, pages 171–182, Vienna, 1994. Springer Verlag.

[29] M. Kudo and J. Sklansky. A Comparative Evaluation of medium and large–scale Feature Selectors for Pattern Classifiers. In *Proc. of the 1st Int. Workshop on Statistical Techniques in Pattern Recognition*, pages 91–96, Prague, Czech Republic, 1997.

[30] P. Langley and S. Sage. Oblivious Decision Trees and Abstract Cases. In *Working Notes of the AAAI94 Workshop on Case Based Reasoning*, pages 113–117, Seattle, WA, 1994. AAAI Press.

[31] N. Littlestone. Learning Quickly when Irrelevant Attributes Abound: A New Linear Threshold Algorithm. *Machine Learning*, 2:285–318, 1988.

[32] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining.* Kluwer Academic Publishers, London, GB, 1998.

[33] H. Liu, H. Motoda, and M. Dash. A Monotonic Measure for Optimal Feature Selection. In *Proc. of the European Conf. on Machine Learning*, pages 101–106. Springer Verlag, 1998.

[34] H. Liu and R. Setiono. A Probabilistic Approach to Feature Selection: a Filter Solution. In *Proc. of the 13th Int. Conf. on Machine Learning*, pages 319–327. Morgan Kaufmann, 1996.

[35] H. Liu and R. Setiono. Feature Selection and Classification: a Probabilistic Wrapper Approach. In *Proc. of the 9th Int. Conf. on Industrial and Engineering Applications of AI and ES*, pages 129–135. Morgan Kaufmann, 1996.

[36] H. Liu and R. Setiono. Scalable Feature Selection for Large Sized Databases. In *Proc. of the 4th World Congress on Expert System*, pages 68–75. Morgan Kaufmann, 1998.

[37] T. Marill and D. M. Green. On Effectiveness on Receptors in Recognition Systems. *IEEE Transactions on Information Theory*, 9:11–17, 1963.

[38] T. M. Mitchell. Generalization as Search. *Artificial Intelligence*, 18(2):203–226, 1982.

[39] M. Modrzejewski. Feature Selection Using Rough Sets Theory. In *Proc. of the European Conf. on Machine Learning*, volume 667, pages 213–226. Springer Verlag, 1993.

[40] A. W. Moore and M. S. Lee. Efficient Algorithms for Minimizing Cross Validation Error. In *Proc. of the 11th Int. Conf. on Machine Learning*, pages 190–198, New Brunswick, NJ, 1994. Morgan Kaufmann.

[41] A. Mucciardi and E. Gose. A Comparison of Seven Techniques for Choosing Subsets of Pattern Recognition Properties. *IEEE Transactions on Computers*, C–20(9):1023–1031, 1971.

[42] P. Narendra and K. Fukunaga. A Branch and Bound Algorithm for Feature Subset Selection. *IEEE Transactions on Computer*, C–26(9):917–922, 1977.

[43] A. Oliveira and A. Sangiovanni-Vincentelli. Constructive Induction using a non–greedy Strategy for Feature Selection. In *Proc. of the 9th Int. Workshop on Machine Learning*, pages 354–360, 1992.

[44] J. Pearl. *Heuristics.* Addison-Wesley, 1983.

[45] P. Pudil, J. Novovicová, and J. Kittler. Floating Search Methods in Feature Selection. *Pattern Recognition Letters*, 15(11):1119–1125, 1994.

[46] J. R. Quinlan. *C4.5 Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[47] J. C. Schlimmer. Efficiently Inducing Determinations: A Complete and Efficient Search Algorithm that uses Optimal Pruning. In *Proc. of the 10th Int. Conf. on Machine Learning*, pages 284–290, Amherst, MA, 1993. Morgan Kaufmann.

[48] J. Segen. Feature Selection and Constructive Inference. In *Proc. of 7th Int. Conf. on Pattern Recognition*, pages 1344–1346. IEEE Press, 1984.

[49] J. Sheinvald, B. Dom, and W. Niblack. A Modelling Approach to Feature Selection. In *Proc. of 10th Int. Conf. on Pattern Recognition*, volume 1, pages 535–539. IEEE Press, 1990.

[50] M. Singh and G. M. Provan. Efficient Learning of Selective Bayesian Network Classifiers. In *Proc. of the 13th Int. Conf. on Machine Learning*, pages 453–461. Morgan Kaufmann, 1996.

[51] D. Skalak. Prototype and Feature Selection by Sampling and Random Mutation Hill Climbing Algorithms. In *Proc. of the 11th Int. Conf. on Machine Learning*, pages 293–301, New Brunswick, NJ, 1994. Morgan Kaufmann.

[52] S. B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Dvzeroski, S. E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R. S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. Van de Welde, W. Wenzel, J. Wnek, and J. Zhang. The MONK's Problems: A Performance Comparison of Different Learning Algorithms. Technical Report CS-91-197, Carnegie Mellon University, Pittsburgh, PA, 1991.

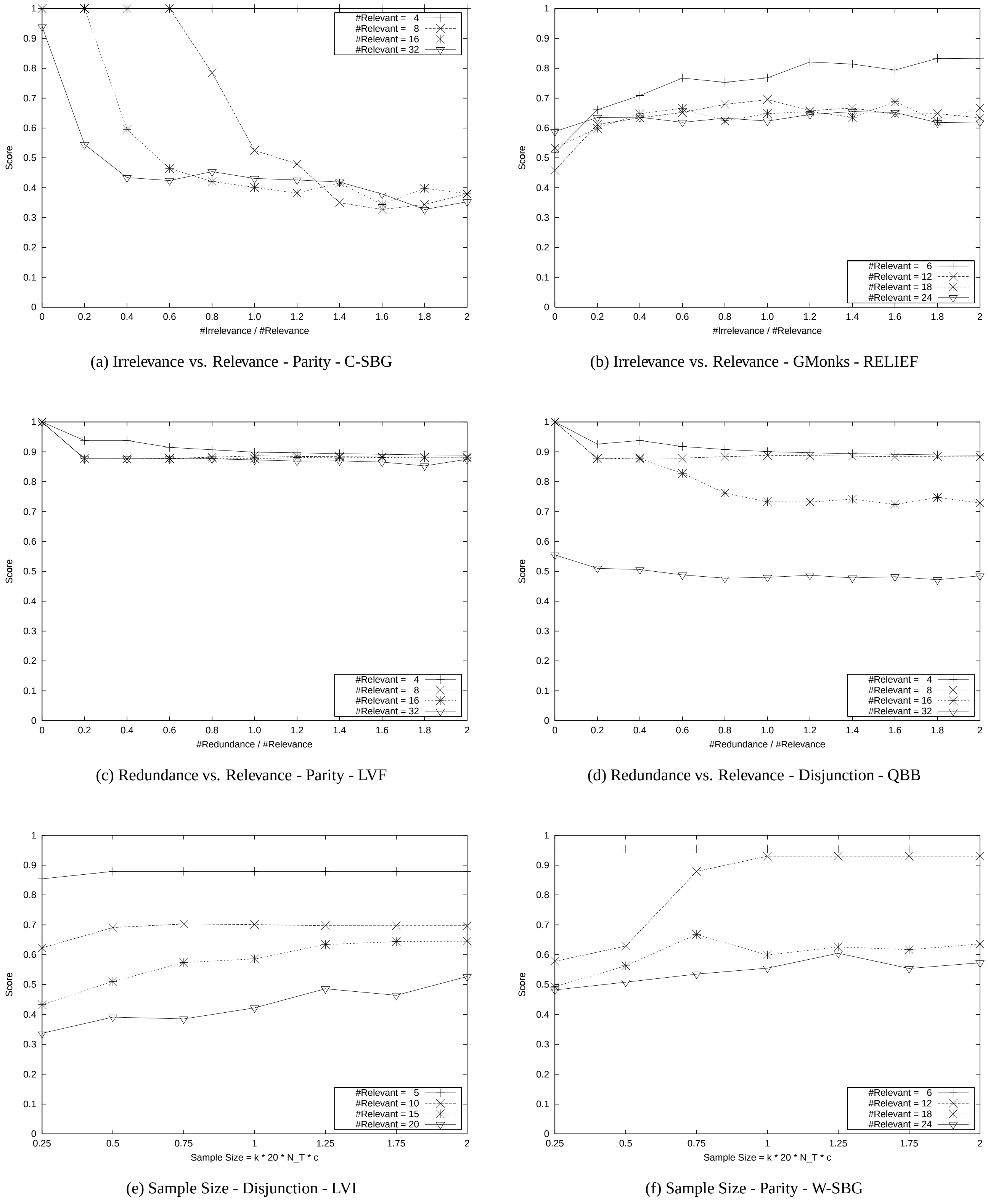
[53] H. Vafaie and I. F. Imam. Feature Selection Methods: Genetic Algorithms vs. Greedy like Search. In *Proc. of Int. Conf. on Fuzzy and Intelligent Control Systems*, 1994.

[54] S. A. Vere. Induction of Concepts in the Predicate Calculus. In *Proc. of the 4th Int. Joint Conf. on Artificial Intelligence*, pages 281–287, Tbilisi, Georgia, 1975. Morgan Kaufmann.

[55] K. Wang, D. Bell, and F. Murtagh. Relevance Approach to Feature Subset Selection. In H. Liu and H. Motoda, editors, *Feature Extraction, Construction and Selection*, pages 85–97. Kluwer Academic Publishers, 1998.

[56] P. H. Winston. Learning Structural Descriptions from Examples. In Winston, P. H., editor, *The Psychology of Computer Vision*, New York, NY, 1975. McGraw Hill.

[57] L. Xu, P. Yan, and T. Chang. Best First Strategy for Feature Selection. In *Proc. of 9th Int. Conf. on Pattern Recognition*, pages 706–708. IEEE Press, 1988.



(a) Irrelevance vs. Relevance - Parity - C-SBG

(b) Irrelevance vs. Relevance - GMonks - RELIEF

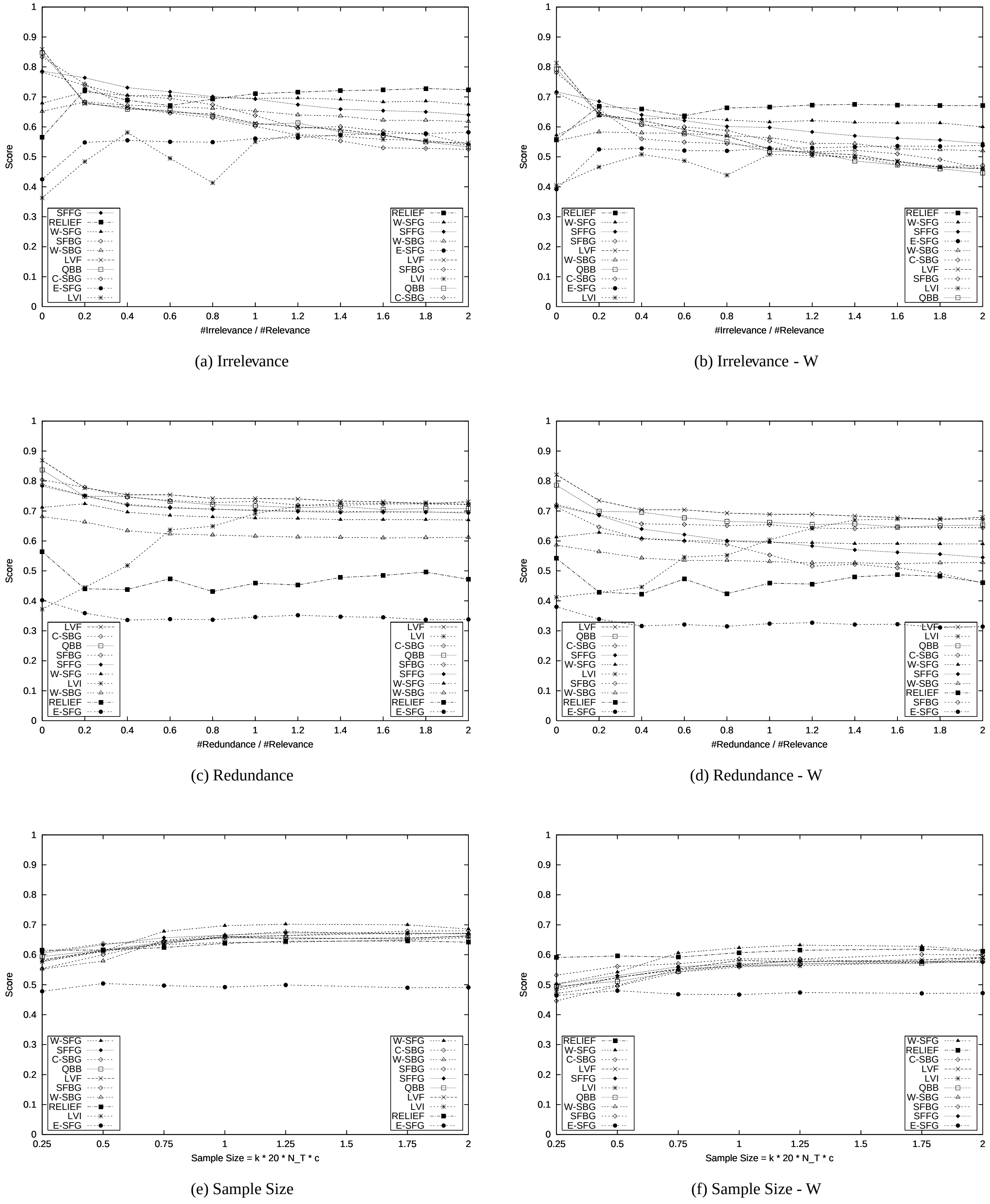
(c) Redundance vs. Relevance - Parity - LVF

(d) Redundance vs. Relevance - Disjunction - QBB

(e) Sample Size - Disjunction - LVI

(f) Sample Size - Parity - W-SBG

Figure 14. Some results of the experiments.



(a) Irrelevance

(b) Irrelevance - W

(c) Redundance

(d) Redundance - W

(e) Sample Size

(f) Sample Size - W

Figure 15. Results ordered by total average performance on the data sets (left inset) and by end performance (right inset). Figs. (b), (d) and (f) are weighed versions of (a), (c) and (e), respectively.