Feature Selection for Classification

M. Dash *and* H. Liu
Department of Information Systems & Computer Science
National University of Singapore, Singapore 119260
{manoranj,liuh}@iscs.nus.sg

Abstract

Feature selection has been the focus of interest for quite some time. Much work has been done. With the creation of huge databases and the consequent requirements for good machine learning techniques, new problems arise and novel approaches to feature selection are in demand. This survey is a comprehensive overview of many existing methods from the 1970's to the present. It identifies four steps of a typical feature selection method, and categorizes the different existing methods in terms of generation procedures and evaluation functions, and reveals hitherto unattempted combinations of generation procedures and evaluation functions. Representative methods are chosen from each category for detailed explanation and discussion via example. Benchmark datasets with different characteristics are used for comparative study. The strengths and weaknesses of different methods are explained. Guidelines for applying feature selection methods are given based on data types and domain characteristics. This survey identifies the future research areas in feature selection, introduces newcomers to this field, and paves the way for practitioners who search for suitable methods for solving domain specific real-world applications.

Keywords: Feature Selection, Classification, Framework.

1 Introduction

The majority of real-world classification problems require supervised learning where the underlying class probabilities and class-conditional probabilities are unknown, and each instance is associated with a class label. In real-world situations relevant features are often unknown *a priori*. Therefore, many candidate features are introduced to better represent the domain. Unfortunately many of these are either partially or completely irrelevant/redundant to the target concept. A relevant feature is neither irrelevant nor redundant to the target concept; an irrelevant feature does not affect the target concept in any way, and a redundant feature does not add anything new to the target concept [21]. In many applications, the size of a dataset is so large that learning might not work as well before removing these unwanted features. Reducing the number of irrelevant/redundant features drastically reduces the running time of a learning algorithm and yields more general concept. This helps in getting better insight into the underlying concept of a real-world classification problem [23, 24]. *Feature selection* methods try to pick a subset of features that are relevant to the target concept.

Feature selection is defined by many authors by looking at it from various angles. But as expected many of those are similar in intuition and/or content. The following lists those that are conceptually different and cover a range of definitions.

1. *Idealized*: Find the minimally sized feature subset that is necessary and sufficient to the target concept [22].

2. *Classical*: Select a subset of M features from a set of N features, $M \leq N$, such that the value of a criterion function is optimized over all subsets of size M [34].

3. *Improving Prediction accuracy*: The aim of feature selection is to choose a subset of features for improving prediction accuracy or decreasing the size of the structure without significantly decreasing prediction accuracy of the classifier built using only the selected features [21].

4. *Approximating original class distribution*: The goal of feature selection is to select a small subset such that the resulting class distribution, given only the values for the selected features, is as close as possible to the original class distribution given all feature values [24].

Notice that the third definition emphasizes the prediction accuracy of a classifier built using only the selected features whereas the last definition emphasizes the class distribution given the training dataset. These two are quite different conceptually. Hence, our definition considers both factors.

Definition: Feature selection attempts to select the minimally sized subset of features according to the following criteria. The criteria can be:

1. the classification accuracy does not significantly decrease, and

2. the resulting class distribution, given only the values for the selected features, is as close as possible to the original class distribution, given all features.

Ideally feature selection methods search through the subsets of features and try to find the best one among the competing 2^N candidate subsets according to some evaluation function. But this procedure is exhaustive as it tries to find only the best one. It may be too costly and practically prohibitive even for medium-sized feature set size (N). Other methods based on heuristic or random search methods attempt to reduce computational complexity by compromising performance. These methods need a stopping criterion to prevent an exhaustive search of subsets. In our opinion, there are four basic steps in a typical feature selection method (refer to Figure 1):

1. a *generation procedure* to generate the next candidate subset,
2. an *evaluation function* to evaluate the subset under examination,
3. a *stopping criterion* to decide when to stop, and
4. a *validation procedure* to check whether the subset is valid.

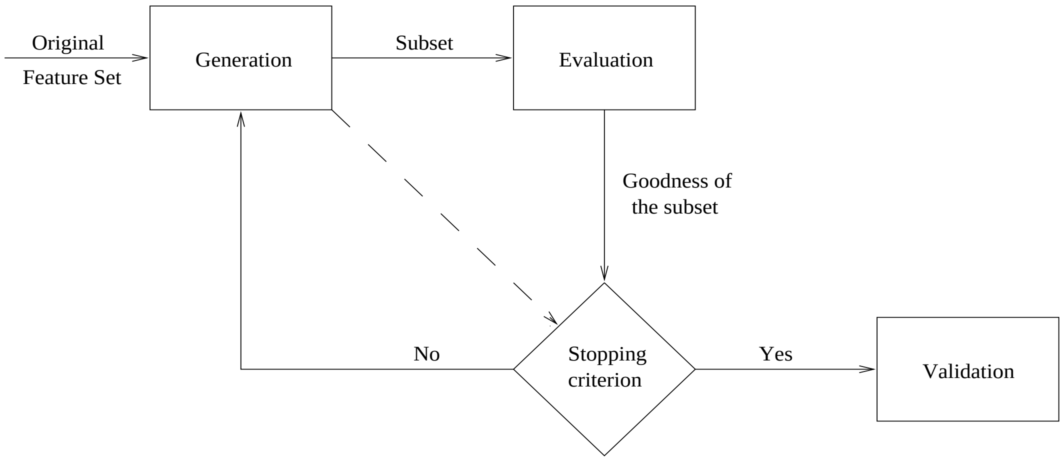

Figure 1: Feature Selection Process with Validation

The *generation procedure* is a *search procedure* ([46], [26], etc). Basically, it generates subsets of features for evaluation. The generation procedure can start (i) with no features, (ii) with all features, or (iii) with a random subset of features. In the first two cases features are iteratively added/removed, whereas in the last case, features are either iteratively added/removed or produced randomly thereafter [26]. An *evaluation function* measures the goodness of a subset produced by some generation procedure, and this value is compared with the previous best. If it is found to be better, then it replaces the previous best subset. Without a suitable *stopping criterion* the feature selection process may run exhaustively or forever through the space of subsets. Generation procedures and evaluation functions can influence the choice for a stopping criterion. Stopping criteria based on a generation procedure include: (i) whether a predefined number of features are selected, and (ii) whether a predefined number of iterations reached. Stopping criteria based on an evaluation function can be: (i) whether addition (or deletion) of any feature does not produce a better subset, and (ii) whether an optimal subset according to some evaluation function is obtained. The loop continues until some stopping criterion is satisfied. The feature selection process halts by outputting a selected subset of features to a validation procedure. There are many variations to this three-step feature selection process, which are discussed in Section 3. The validation procedure is not a part of the feature selection process itself, but a feature selection method (in practice) must be validated. It tries to test the validity of the selected subset by carrying out different tests and comparing the results with previously established results or with the results of competing feature selection methods using artificial datasets, real-world datasets, or both.

There have been quite a few attempts to study feature selection methods based on some framework or structure. Prominent among these are Doak's [13] and Siedlecki and Sklansky's [46] surveys. Siedlecki and Sklansky discussed the evolution of feature selection methods and grouped the methods into past, present, and future categories. Their main focus was branch and bound method [34] and its variants (e.g., [16]). No experimental study was conducted in this paper. Their survey was published in the year 1987, and since then many new and efficient methods have been introduced (e.g., Focus [2], Relief [22], LVF [28]). Doak followed a similar approach to Siedlecki and Sklansky's survey and grouped the different search algorithms and evaluation functions used in feature selection methods independently, and ran experiments using some combinations of evaluation functions and search procedures.

In this paper, a survey is conducted for feature selection methods starting from the early 1970's (e.g., [33]), to the most recent methods (e.g., [28]). In the next section, the major two steps of feature selection (generation procedure and evaluation function) are divided into different groups, and 32 different feature selection methods are categorized based on the type of generation procedure and evaluation function used in them. This framework helps in finding the unexplored combinations of generation procedures and evaluation functions. In Section 3 we briefly discuss the methods under each category, and select a representative method for detailed description using a simple dataset. Section 4 describes an empirical comparison of the representative methods using three artificial datasets suitably chosen to highlight their benefits and limitations. Section 5 consists of discussions on various data set characteristics that influence the choice of a suitable feature selection method, and some guidelines, regarding how to choose a feature selection method for an application at hand, are given based on a number of criteria extracted from these characteristics of data. This paper concludes in Section 6 with further discussions on future research based on the findings of Section 2. Our objective is that this paper will assist in finding better feature selection methods for given applications.

2 Study of Feature Selection Methods

In this section we categorize the two major steps of feature selection: generation procedure and evaluation function. The different types of evaluation functions are compared based on a number of criteria. A framework is presented in which a total of 32 methods are grouped based on the types of generation procedure and evaluation function used in them.

2.1 Generation Procedures

If the original feature set contains N number of features then the total number of competing candidate subsets to be generated is 2^N. This is a huge number even for medium-sized N. There are different approaches of solving this problem, namely: *complete*, *heuristic*, and *random*.

Complete

This generation procedure does a complete search for the optimal subset according to the evaluation function used. An exhaustive search is complete. However, Schlimmer [43] argues that *just because the search must be complete does not mean that it must be exhaustive.* Different heuristic functions are used to reduce the search without jeopardizing the chances of finding the optimal subset. Hence, although the order of the search space is $O(2^N)$, a fewer subsets are evaluated. The optimality of the feature subset, according to the evaluation function used, is guaranteed because the procedure can backtrack. Backtracking is done using various techniques, such as: branch and bound with backtracking, breadth first search, and best first search with backtracking.

Heuristic

In each iteration of this generation procedure all remaining features yet to be selected (rejected) are considered for selection (rejection). There are many variations to this simple process, but generation of subsets is basically incremental (either increasing or decreasing). Order of the search space is $O(N^2)$ or less; some exceptions are Relief [22], DTM [9] that are discussed in detail in the next section. These procedures are very simple to implement and very fast in producing results because the search space is only quadratic in terms of the number of features.

Random

This generation procedure is rather new in its use in feature selection methods compared to the other two categories. Although the search space is $O(2^N)$, but these methods typically search fewer number of subsets than 2^N by setting a maximum number of iterations possible. Optimality of the selected subset depends on the resources available. Each random generation procedure would require values of some parameters. Assignment of suitable values to these parameters is an important task for achieving good results.

2.2 Evaluation Functions

An optimal subset is always relative to certain evaluation function (i.e., an optimal subset chosen using one evaluation function may not be the same as that using another evaluation function). Typically, an evaluation function tries to measure the capability of a feature or a subset to distinguish among different class labels. Langley [26] grouped different feature selection methods into two broad groups (i.e., filter and wrapper) based on their dependence on the inductive algorithm that will finally use the selected subset. *Filter* methods are independent of the inductive algorithm, whereas *wrapper* methods use the inductive algorithm as the evaluation function. Ben-Bassat [4] grouped the evaluation functions existing till 1982 into three categories: information or uncertainty measures, distance measures, and dependence measures, and suggested that the dependence measures can be divided between the first two categories. He has not considered the classification error rate as an evaluation function as no wrapper method existed in 1982. Doak [13] divided the evaluation functions into three categories: data intrinsic measures, classification error rate, and estimated or incremental error rate, where the third category is basically a variation of the second category. The data intrinsic category includes distance, entropy, and dependence measures. Considering these divisions and the latest developments, we divide the evaluation functions into five categories: *distance*, *information (or uncertainty)*, *dependence*, *consistency*, and *classifier error rate*. In the following subsections we briefly discuss each of these types of evaluation functions.

Distance Measures

It is also known as separability, divergence, or discrimination measure. For a two-class problem, a feature X is preferred to another feature Y if X induces a greater difference between the two-class conditional probabilities than Y; if the difference is zero then X and Y are indistinguishable. An example is Euclidean distance measure.

Information Measures

These measures typically determine the information gain from a feature. The information gain from a feature X is defined as the difference between the prior uncertainty and expected posterior uncertainty using X. A feature X is preferred to feature Y if the information gain from feature X is greater than that from feature Y [4]. An example is entropy measure.

Dependence Measures

Dependence measures or correlation measures quantify the ability to predict the value of one variable from the value of another variable. Correlation coefficient is a classical dependence measure and can be used to find the correlation between a feature and a class. If the correlation of feature X with class C is higher than the correlation of feature Y with C, then feature X is preferred to Y. A slight variation of this is to determine the dependence of a feature on other features; this value indicates the degree of redundancy of the feature. All evaluation functions based on dependence measures can be divided between distance and information measures. But, these are still kept as a separate category because, conceptually, they represent a different viewpoint [4].

More about the above three measures can be found in Ben-Bassat's [4] survey.

Consistency Measures

These measures are rather new and have been in much focus recently. These are characteristically different from other measures because of their heavy reliance on the training dataset and use of Min-Features bias in selecting a subset of features [3]. Min-Features bias prefers consistent hypotheses definable over as few features as possible. More about this bias is discussed in Section 3.6. These measures find out the minimally sized subset that satisfies the acceptable inconsistency rate, that is usually set by the user.

Classifier Error Rate Measures

The methods using this type of evaluation function are called "wrapper methods", *i.e.*, the classifier is the evaluation function. As the features are selected using the classifier that later on uses these selected features in predicting the class labels of unseen instances, the accuracy level is very high although computationally quite costly [21].

Table 1 shows a comparison of various evaluation functions irrespective of the type of generation procedure used. The different parameters used for the comparison are:

1. *generality*: how suitable is the selected subset for different classifiers (not just for one classifier),
2. *time complexity*: time taken for selecting the subset of features, and

3. *accuracy*: how accurate is the prediction using the selected subset.

Evaluation Function	*Generality*	*Time Complexity*	*Accuracy*
Distance Measure	Yes	Low	_
Information Measure	Yes	Low	_
Dependence Measure	Yes	Low	_
Consistency Measure	Yes	Moderate	_
Classifier Error Rate	No	High	Very High

Table 1: A Comparison of Evaluation Functions

The '_' in the last column means that nothing can be concluded about the accuracy of the corresponding evaluation function. Except 'Classifier error rate' the accuracy of all other evaluation functions depends on the data set and the classifier (for classification after feature selection) used. The results in Table 1 show a non-surprising trend (i.e., the more time spent, the higher the accuracy). The table also tells us which measure should be used under different circumstances, for example, with time constraints, given several classifiers to choose from, classifier error rate should not be selected as an evaluation function.

2.3 The Framework

In this subsection we suggest a framework on which the feature selection methods are categorized. Generation procedures and evaluation functions are considered as two dimensions, and each method is grouped depending on the type of generation procedure and evaluation function used. To our knowledge, to this date there has not been any attempt to group the methods considering both steps. First we have chosen a total of 32 methods from the literature, and then these are grouped according to the combination of generation procedure and evaluation function used (see Table 2). A distinct achievement of this framework is the finding of a number of combinations of generation procedure and evaluation function (the empty boxes in the table) that do not appear in any existing method yet (to the best of our knowledge).

In the framework a column stands for a type of generation procedure and a row stands for a type of evaluation function. The assignment of evaluation functions within the categories may be equivocal, since several evaluation functions may be placed in different categories when considering them in different perspectives, and one evaluation function may be obtained as a mathematical transformation of another evaluation function [4]. We have tried to resolve this as naturally as possible. In the next two sections we explain each category and methods in it, and choose a method in each category, for a detailed discussion using pseudo code and a mini-dataset (Section 3), and for an empirical comparison (Section 4).

3 Categories and Their Differences

There are a total of 15 (i.e., 5 types of evaluation functions and 3 types of generation procedures) combinations of generation procedures and evaluation functions in Table 2. The blank boxes in the table signify that no method exists for these combinations. These combinations are the

Evaluation Measures	**Generation**		
	Heuristic	*Complete*	*Random*
Distance Measure	*I* - Sec 3.1 Relief Relief-F Sege84	*II* - Sec 3.2 Branch and Bound BFF Bobr88	*III*
Information Measure	*IV* - Sec 3.3 DTM Koll-Saha96	*V* - Sec3.4 MDLM	*VI*
Dependency Measure	*VII* - Sec 3.5 POE+ACC PRESET	*VIII*	*IX*
Consistency Measure	*X*	*XI* - Sec 3.6 Focus Schl93 MIFES_1	*XII* - Sec 3.7 LVF
Classifier Error Rate	*XIII* - Sec 3.8 SBS SFS SBS-SLASH PQSS BDS Moor-Lee94 RC Quei-Gels84	*XIV* - Sec 3.8 Ichi-Skla84a Ichi-Skla84b AMB & B BS	*XV* - Sec 3.8 LVW GA SA RGSS RMHC-PF1

Table 2: A Two Dimensional Categorization of Feature Selection Methods

potential future work. In this section, we discuss each category, by briefly describing the methods under it, and then, choosing a representative method for each category, we explain it in detail with the help of its pseudo code and a hand-run over a prototypical dataset. The methods in the last row represent the "wrapper" methods where the evaluation function is "classifier error rate". A typical wrapper method can use different kinds of classifiers for evaluation; hence no representative method is chosen for the categories under this evaluation function. Instead they are discussed briefly as a whole towards the end of this Section. The prototypical dataset used for hand-run of the representative methods is shown in Table 3 which consists of 16 instances (originally 32 instances) of original CorrAL dataset. This mini-dataset has binary classes, and six boolean features $(A_0, A_1, B_0, B_1, I, C)$ where feature I is *irrelevant*, feature C is *correlated* to the class label 75% of the time, and the other four features are relevant to the boolean target concept: $(A_0 \wedge A_1) \vee (B_0 \wedge B_1)$. In all the pseudo codes, D denotes the training set, S is the original feature set, N is the number of features, T is the selected subset, and M is the number of selected (or required) features.

#	A0	A1	B0	B1	I	C	Class	#	A0	A1	B0	B1	I	C	Class
1	0	0	0	0	0	1	0	9	1	0	0	0	1	1	0
2	0	0	0	1	1	1	0	10	1	0	0	1	1	0	0
3	0	0	1	0	0	1	0	11	1	0	1	0	0	1	0
4	0	0	1	1	0	0	1	12	1	0	1	1	0	0	1
5	0	1	0	0	0	1	0	13	1	1	0	0	0	0	1
6	0	1	0	1	1	1	0	14	1	1	0	1	0	1	1
7	0	1	1	0	1	0	0	15	1	1	1	0	1	0	1
8	0	1	1	1	0	0	1	16	1	1	1	1	0	0	1

Table 3: Sixteen Instances of CorrAL

3.1 Category I: (Generation – Heuristic, Evaluation – Distance)

Brief Description of Various Methods: As seen from Table 2, the most prominent method in this category is Relief [22]. We first discuss Relief and its variant followed by a brief discussion on the other method.

Relief uses a statistical method to select the relevant features. It is a feature weight based algorithm inspired by instance-based learning algorithms ([1], [8]). From the set of training instances, it first chooses a sample of instances; the user must provide the number of instances ($NoSample$) in this sample. Relief randomly picks this sample of instances, and for each instance in it finds $nearHit$ and $nearMiss$ instances based on an Euclidean distance measure. $NearHit$ is the instance having minimum Euclidean distance among all instances of the same class as that of the chosen instance; $nearMiss$ is the instance having minimum Euclidean distance among all instances of different class. It updates the weights of the features that are initialized to zero in the beginning based on an intuitive idea that a feature is more relevant if it distinguishes between an instance and its $nearMiss$, and less relevant if it distinguishes between an instance and its $nearHit$. After exhausting all instances in the sample, it chooses all features having weight greater than or equal to a threshold. This threshold can be automatically evaluated using a function that uses the number of instances in the sample; it can also be determined by inspection (all features with positive weights are selected). Relief works for noisy and correlated features, and requires only linear time in the number of given features and $NoSample$. It works both for nominal and continuous data. One major limitation is that it does not help with redundant features and, hence, generates non-optimal feature set size in presence of redundant features. This can be overcome by a subsequent exhaustive search over the subsets of all features selected by Relief. Relief works only for binary classes. This limitation is overcome by Relief-F [25] that also tackles incomplete data. Insufficient training instances fool Relief. Another limitation is that the user may find difficulty in choosing a proper $NoSample$. More about this is discussed in Section 4.3.

Jakub Segen's [44] method uses an evaluation function that minimizes the sum of a statistical discrepancy measure and the feature complexity measure (in bits). It finds the first feature that best distinguishes the classes, and iteratively looks for additional features which in combination with the previously chosen features improve class discrimination. This process stops once the minimal representation criterion is achieved.

Hand-run of CorrAL Dataset: (Refer to Figure 2) Relief randomly chooses one instance from the sample. Let us assume instance # 5 (i.e., (0,1,0,0,0,1)) is chosen. It finds the $nearHit$ and the

Relief(D, S, $NoSample$, $Threshold$)

1. $T = \phi$
2. Initialize all weights, W_i, to zero.
3. For i = 1 to $NoSample$/* Arbitrarily chosen */
   Randomly choose an instance x n D
   Find its $nearHit$ and $nearMiss$
   For j = 1 to N
   $W_j = W_j - diff(x_j, nearHit_j)^2 + diff(x_j, nearMiss_j)^2$
4. For j = 1 to N
   If $W_j \geq Threshold$
   Append feature f_j to T
5. Return T

Figure 2: Relief [22]

$nearMiss$ of this instance using Euclidean distance measure. Difference between two discrete feature values is one if their values do not match and zero otherwise. For the chosen instance # 5, the instance # 1 is the $nearHit$ (*difference* = 1), and instance # 13 and #14 are the $nearMiss$ (*difference* = 2). Let us choose instance # 13 as the $nearMiss$. Next, it updates each feature weight W_j. This is iterated $NoSample$ times specified by the user. Features having weights greater than or equal to the threshold are selected. Usually, the weights are negative for irrelevant features, and positive for relevant and redundant features. For CorrAL dataset it selects $\{A_0, B_0, B_1, C\}$ (more about this in Section 4.3).

3.2 Category II: (Generation – Complete, Evaluation – Distance)

Brief Description of Various Methods: This combination is found in quite old methods such as branch and bound [34]. Other methods in this category are variations of branch and bound (B & B) method regarding the generation procedure used [50], or the evaluation function used [5]. We first discuss the branch and bound method, followed by a brief discussion of the other two methods.

Narendra and Fukunaga have defined feature selection in a classical way (see definition 2 in Section 1) that requires evaluation functions be monotonic (i.e., a subset of features should not be better than any larger set that contains the subset). This definition has a severe drawback for real-world problems because the appropriate size of the target feature subset is generally unknown. But this definition can be slightly modified to make it applicable for general problems as well, by saying that B & B attempts to satisfy two criteria: (i) the selected subset be as small as possible and (ii) a bound be placed on the value calculated by the evaluation function [13]. As per this modification, B & B starts searching from the original feature set and proceeds by removing features from it. A bound is placed on the value of the evaluation function to make the search fast. As the evaluation function obeys the monotonicity principle, any subset for which the value is less than the bound is removed from the search tree (i.e., all subsets of it are discarded from the search space). Evaluation functions generally used are: Mahalanobis distance [15], the discriminant function, the Fisher criterion, the Bhattacharya distance, and the divergence [34]. Xu, Yan, and Chang [50] proposed a similar algorithm (BFF), where the search procedure is modified to solve the problem of searching an optimal path in a weighted tree with the *informed best first search* strategy in artificial intelligence. This algorithm guarantees the globally best subset without exhaustive enumeration, for any criterion satisfying monotonicity principle.

Bobrowski [5] proves that the homogeneity coefficient ϕ_k^* can be used in measuring the degree of linear dependence among some measurements, and shows that it is applicable to the feature selection problem due to its monotonicity principle (i.e., if $S_1 \subset S_2$ then $\phi_k^*(S_1) \geq \phi_k^*(S_2)$). Hence, this can be suitably converted to a feature selection method by implementing it as an evaluation function for branch and bound with backtracking or a best first generation procedure.

The fact that an evaluation function must be monotonic to be applicable to these methods prevents the use of many common evaluation functions. This problem is partially solved by relaxing the monotonicity criterion and introducing the approximate monotonicity concept [16].

B&B (S, M)

1. if ($card(S)$ <> M) {
   /* subset generation */
   $j = 0$
   For all features f in S {
     $S_j = S - f$ /* remove one feature at a time */
     if (S_j is legitimate) {
       if $IsBetter(S_j, T)$
         $T = S_j$
       /* recursion */
       B&B (S_j, M) }
     j++ }
   return T }}

Figure 3: Branch and Bound [34]

Hand-run of CorrAL Dataset: (Refer to Figure 3) The authors use Mahalanobis distance measure as the evaluation function. The algorithm needs input of the required number of features (M) and it attempts to find out the best subset of $N - M$ features to reject. It starts with the full set of features S^0, removes one feature from S_j^{l-1} in turn to generate subsets S_j^l where l is the current level and j specifies different subsets at the lth level. If $U(S_j^l) > U(S_j^{l-1})$, S_j^l stops growing (its branch is pruned), otherwise, it grows to level $l+1$, in other words, one more feature will be removed. If for the CorrAL dataset M is set to 4 then subset (A_0, A_1, B_0, I) is selected as the best subset of 4 features.

3.3 Category IV: (Generation – Heuristic, Evaluation – Information)

Brief Description of Various Methods: We have found two methods under this category: DTM (Decision Tree Method) [9], and Koller and Sahami's [24] method. DTM shows that the use of feature selection can improve case-based learning. In this method feature selection is used in an application on natural language processing. C4.5 [39] is run over the training set, and the features that appear in the pruned decision tree are selected. In other words, the union of the subsets of features, appearing in the paths to any leaf node in the pruned tree, is the selected subset. The second method, which is very recent, is based on the intuition that any feature, given little or no additional information beyond that subsumed by the remaining features, is either irrelevant or redundant, and should be eliminated. To realize this, Koller and Sahami try to approximate *Markov blanket* where a subset T is a *Markov blanket* for feature f_i if, given T, f_i is conditionally independent both of the class label and of all features not in T (including f_i itself). The implementation of *Markov blanket* is suboptimal in many ways, particularly due to many naive approximations.

DTM(D)

1. $T = \phi$
2. Apply C4.5 to the training set,D
3. Append all features appearing in the pruned decision tree to T
4. Return T

Figure 4: DTM (Decision Tree Method) [9]

Hand-run of CorrAL Dataset: (Refer to Figure 4) C4.5 uses an information-based heuristic, a simple form of which for two class problem (as our example) is $I(p,n) = \frac{-p}{p+n}\frac{log_2 p}{p+n} - \frac{n}{p+n}\frac{log_2 n}{p+n}$, where p is the number of instances with class label *one* and n is the number of instances with class label *zero*. Assume that using an attribute F_1 as the root in the tree will partition the training set into T_0 and T_1 (because each feature takes binary values only). Entropy of feature F_1 is $E(F_1) = \frac{p_0+n_0}{p+n} I(p_0,n_0) + \frac{p_1+n_1}{p+n} I(p_1,n_1)$. Considering the CorrAL dataset let us assume that feature C is evaluated. If it takes value '0' then one instance is positive ($class = 1$) and seven instances are negative ($class = 0$), and for value '1' six instances are positive ($class = 1$) and two are negative ($class = 0$). Hence $E(C) = \frac{1+7}{16} I(1,7) + \frac{6+2}{16} I(6,2)$ is 0.552421. In fact this value is minimum among all features, and C is selected as the root of the decision tree. The original training set of sixteen instances is divided into two nodes each representing eight instances using feature C. For these two nodes again features having least entropy are selected, and so on. This process halts when each partition contains instances of a single class, or until no test offers any improvement. The decision tree constructed thus is pruned basically to avoid over-fitting. DTM chooses all features appearing in the pruned decision tree (i.e., A_0, A_1, B_0, B_1, C).

3.4 Category V: (Generation – Complete, Evaluation – Information)

Brief Description of Various Methods: Under this category we found MDLM (Minimum Description Length Method) [45]. In this method, the authors have attempted to eliminate all *useless* (irrelevant and/or redundant) features. According to the authors, if the features in a subset V can be expressed as a fixed non-class-dependent function F of the features in another subset U, then once the values of the features in subset U are known, the feature subset V becomes useless. For feature selection, U and V together make up the complete feature set and one has the task of separating them. To solve this, the authors use the minimum description length criterion (MDLC) introduced by Rissanen [40]. They formulated an expression that can be interpreted as the number of bits required to transmit the classes of the instances, the optimal parameters, the useful features, and (finally) the useless features. The algorithm exhaustively searches all the possible subsets (2^N)

and outputs the subset satisfying MDLC. This method can find all the useful features and only those if the observations are Gaussian. For non-Gaussian cases, it may not be able to find the useful features (more about this in Section 4.3).

MDLM(D)

1. $MDL = \infty$

2. For all feature subsets L
   1.1 Compute $Length_L = \sum_{i=1}^{i=q} \frac{P_i}{2} \log \frac{|D_L(i)|}{|D_L|} + h_L$
   where $h_L = \frac{1}{2}(N-M)(N+M+3)\log P + \sum_{i=1}^{i=q} M(M+3)\log P_i$,
   N – total number of features,
   M – number of features in the candidate subset,
   P – total number of instances in D,
   P_i – number of instances with class label i,
   q – total number of class labels,
   D_L – covariance matrix formed from all the useful feature vectors,
   $D_L(i)$ – covariance matrix formed from the useful feature vectors of class i,
   $|.|$ – denotes determinant.
   1.2 If $Length_L < MDL$ then
   $T = L$, $MDL = Length_L$

3. Return T

Figure 5: MDLM (Minimum Description Length Method) [45]

Hand-run of CorrAL Dataset: (Refer to Figure 5) As seen in Figure 5, MDLM is basically the evaluation of an equation that represents the description length given the candidate subset of features. The actual implementation suggested by the authors is as follows. Calculate the covariance matrices of the whole feature vectors for all classes and for each class separately. The covariance matrix for useful subsets are obtained as sub-matrices of these. Determine the determinants of the sub-matrices $D_L(i)$ and D_L for the equation as shown in Figure 5 and find out the subset having minimum description length. For the CorrAL dataset the chosen feature subset is $\{C\}$ with a minimum description length of 119.582.

3.5 Category VII: (Generation – Heuristic, Evaluation – Dependence)

Brief Description of Various Methods: We found POE+ACC (Probability of ERROR & Average Correlation Coefficient) method [33] and PRESET [31] under this combination. As many as seven techniques of feature selection are presented in POE+ACC, but we choose the last (i.e., seventh) method because the authors consider this to be the most important among the seven. In this method, the first feature selected is the feature with the smallest probability of error (P_e). The next feature selected is the feature that produces the minimum weighted sum of P_e and average correlation coefficient (ACC), and so on. ACC is the mean of the correlation coefficients of the candidate feature with the features previously selected at that point. This method can rank all the features based on the weighted sum, and a required number of features (M) is used as the stopping criterion. PRESET uses the concept of rough set; it first finds a reduct (i.e., a reduct R

of a set P classifies instances equally well as P does) and removes all features not appearing in the reduct. Then it ranks the features based on their significance. The significance of a feature is a measure expressing how important a feature is regarding classification. This measure is based on dependency of attributes.

POE+ACC(M, w_1, w_2)

1. $T = \phi$
2. Find the feature with minimum P_e and append it to T
3. For i=1 to M-1
   Find the next feature with minimum $w_1(P_e) + w_2(ACC)$
   Append it to T
4. Return T

Figure 6: POE+ACC [33]

<u>*Hand-run of CorrAL Dataset*</u>: (Refer to Figure 6) The first feature chosen is the feature having minimum P_e, and in each iteration thereafter, the feature having minimum $w_1(P_e) + w_2(ACC)$ is selected. In our experiments, we consider w_1 to be 0.1 and w_2 to be 0.9 (authors suggest these values for their case study). To calculate P_e, compute the *a priori* probabilities of different classes. For the mini-dataset, it is $\frac{9}{16}$ for class 0 and $\frac{7}{16}$ for class 1. Next, for each feature calculate the class-conditional probabilities given the class label. We find that for class label 0, the class-conditional probability of feature C having the value 0 is $\frac{2}{9}$ and that of value 1 is $\frac{7}{9}$, and for class label 1, the class-conditional probability of feature C having the value 0 is $\frac{6}{7}$ and that of value 1 is $\frac{1}{7}$. For each feature value (e.g., feature C taking value 0), we find out the class label for which the product of *a priori* class probability and class-conditional probability given the class label is maximum. When feature C takes the value 0, the prediction is class 1, and when feature C takes value 1, the prediction is class 0. For all instances, we count the number of mismatches between the actual and the predicted class values. Feature C has $\frac{3}{16}$ fraction of mismatches (P_e). In fact, among all features C has the least P_e and hence, is selected as the first feature. In the second step, correlations of all remaining features (A_0, A_1, B_0, B_1, I) with feature C are calculated. Using the expression $w_1(P_e) + w_2(ACC)$ we find that feature A_0 has the least value among the five features and is chosen as the second feature. If M (required number) is 4 then subset (C, A_0, B_0, I) is selected.

3.6 Category XI: (Generation - Complete, Evaluation - Consistency)

<u>*Brief Description of Various Methods*</u>: The methods under this combination are developed in the recent years. We discuss Focus [2], Schlimmer's [43] method, and MIFES_1 [35], and select Focus as the representative method for detailed discussion and empirical comparison.

Focus implements the *Min-Features* bias that prefers consistent hypotheses definable over as few features as possible. In the simplest implementation, it does a breadth first search and checks for any inconsistency considering only the candidate subset of features. Strictly speaking Focus is unable to handle noise, but a simple modification that allows a certain percentage of inconsistency will enable it to find the minimally sized subset satisfying the permissible inconsistency.

The other two methods can be seen as variants. Schlimmer's method uses a systematic enumeration scheme as generation procedure and the inconsistency criterion as the evaluation function. It uses an heuristic function that makes the search for the optimal subset faster. This heuristic function is a reliability measure based on the intuition that the probability that an inconsistency will be observed is proportional to the percentage of values that have been observed very infrequently considering the subset of features. All supersets of an unreliable subset are also unreliable. MIFES_1 is very similar to Focus in its selection of features. It represents the set of instances in the form of a matrix[1], each element of which stands for a unique combination of a positive instance (class=1) and a negative instance (class=0). A feature f is said to *cover* an element of the matrix if it assumes opposite values for a positive instance and a negative instance associated with the element. It searches for a cover with $N-1$ features starting from one with all N features, and iterates until no further reduction in the size of the cover is attained.

Focus(D, S)

1. $T = S$

2. For i=0 to N
    For each subset L of size i
      If no inconsistency in the training set D then
        $T = L$
        Return T

Figure 7: Focus [2]

Hand-run of CorrAL Dataset: (Refer to Figure 7) As Focus uses breadth-first generation procedure, it first generates all subsets of size one (i.e., $\{A_0\}, \{A_1\}, \{B_0\}, \{B_1\}, \{I\}, \{C\}$), followed by all subsets of size two, and so on. For each subset generated thus, it checks whether there are at least two instances in the dataset having equal values for all features under examination with different class labels (i.e., inconsistency). If such a case arises, it rejects the subset saying inconsistent, and moves on to test the next subset. This continues until it finds a subset having no inconsistency or the search is complete (i.e., all possible subsets are found inconsistent). For subset $\{A_0\}$, instance # 1 and 4 have the same A_0 value (i.e., 0), but different class labels (i.e., 0 and 1 respectively). Hence it rejects it and moves on to the next subset $\{A_1\}$. It evaluates a total of 41 subsets $(\binom{6}{1} + \binom{6}{2} + \binom{6}{3})$ before selecting the subset (i.e., $\{A_0, A_1, B_0, B_1\}$).

3.7 Category XII: (Generation – Random, Evaluation – Consistency)

Brief Description of Various Methods: This category is rather new and its representative is LVF [28]. LVF randomly searches the space of subsets using a Las Vegas algorithm [6] that makes probabilistic choices to help guide them more quickly to an optimal solution, and uses a consistency measure that is different from that of Focus. For each candidate subset it calculates an inconsistency count based on the intuition that the most frequent class label among those instances matching this subset of features is the most probable class label. An inconsistency threshold is fixed in the beginning (0 by default), and any subset, having an inconsistency rate greater than it, is rejected. This method can find the optimal subset even for datasets with noise, if the rough correct noise

[1] MIFES_1 can only handle binary classes and boolean features.

level is specified in the beginning. An advantage is that the user does not have to wait too long for a good subset because it outputs any subset that is found to be better than the previous best (both by the size of the subset and its inconsistency rate). This algorithm is efficient, as only the subsets whose number of features is smaller than or equal to that of the current best subset are checked for inconsistency. It is simple to implement and guaranteed to find the optimal subset if resources permit. One drawback is that it may take more time to find the optimal subset than algorithms using heuristic generation procedure for certain problems, since it cannot take advantage of prior knowledge.

LVF(D, S, $MaxTries$, δ)

1. $T = S$
2. for j=1 to $MaxTries$ {
   - randomly choose a subset of features, S_j
   - if $card(S_j) \leq card(T)$
     - if $inConCal(S_j, D) \leq \delta$
       - if $card(S_j) < card(T)$
         - $T = S_j$
         - output S_j
       - else
         - append S_j to T
         - output S_j as 'yet another solution' }}
3. return T

Figure 8: LVF [28]

<u>*Hand-run of CorrAL Dataset*</u>: (Refer to Figure 8) LVF [28] chooses a feature subset randomly and calculates its cardinality. The best subset is initialized to the complete feature set. If the randomly chosen subset has cardinality less than or equal to that of the current best subset, it evaluates inconsistency rate of the new subset. If the rate is less than or equal to the threshold value (default value is zero), the new subset is made the current best subset. Let us say the subset $\{A_0, B_0, C\}$ is randomly chosen. Patterns (0,1,0), (1,0,0), and (1,0,1) have mixed class labels with class distribution (1,2), (1,1), and (1,1) for class 0 and 1 respectively. The inconsistency count is the sum of the number of matching instances minus the number of the matching instances with the most frequent class label for each pattern. Hence for the subset $\{A_0, B_0, C\}$ the inconsistency count is 3 ($=(3-2)+(2-1)+(2-1)$) and the inconsistency rate is $\frac{3}{16}$. As we have not specified any threshold inconsistency rate and it is zero by default, so this subset is rejected. But if the randomly chosen subset is $\{A_0, A_1, B_0, B_1\}$ then the inconsistency rate is zero and it becomes the current best subset. No subset of smaller size has inconsistency rate zero leading to the selection of $\{A_0, A_1, B_0, B_1\}$.

3.8 Category XIII-XV: (Evaluation – Classifier Error Rate)

In this subsection, we briefly discuss and give suitable references for the methods under classifier error rate evaluation function that are commonly known as "wrapper" methods.

Heuristic: Under the *heuristic* generation function we find popular methods such as SFS (Sequential Forward Selection) and SBS (Sequential Backward Selection) [12]. SBS-SLASH (Sequential Backward Selection-SLASH [10]), PQSS ((p,q) Sequential Search), BDS (Bi-Directional Search) [13], Schemata Search [32], RC (Relevance in Context [14]), and Queiros and Gelsema's [37] method are variants of SFS, or SBS, or both. SFS starts from the empty set, and in each iteration generates new subsets by adding a feature selected by some evaluation function. SBS starts from the complete feature set, and in each iteration generates new subsets by discarding a feature selected by some evaluation function. SBS-SLASH is based on the observation that when there is a large number of features, some classifiers (e.g., ID3/C4.5) frequently do not use many of them. It starts with the full feature set (like SBS), but after taking a step, eliminates (slashes) any feature not used in what was learned at that step. BDS allows to search from both ends, whereas PQSS provides some degree of backtracking by allowing both addition and deletion for each subset. If PQSS starts from the empty set, then it adds more features than it discards in each step, and if it starts from the complete feature set, then it discards more features and adds less features in each step. Schemata search starts either from the empty set or the complete set, and in each iteration, finds the best subset by either removing or adding only one feature to the subset. It evaluates each subset using LOOCV (leave-one-out cross validation), and in each iteration, selects the subset having least LOOCV error. It continues this way until no single-feature change improves it. RC considers the fact that some features will be relevant only in some parts of the space. It is similar to SBS, but with the crucial difference that it makes local, instance-specific decisions on feature relevance, as opposed to global ones. Queiros and Gelsema's method is similar to SFS, but it suggests that at each iteration, evaluate each feature in various settings by considering different interactions with the set of features previously selected. Two simple settings are (i) always assume independence of features (do not consider the previously selected features), and (ii) never assume independence (consider the previously selected features). They used Bayesian classifier error rate as the evaluation function.

Complete: Under *complete* generation procedure we find four *wrapper* methods. Ichino and Sklansky have devised two methods based on two different classifiers, namely: linear classifier [19] and box classifier [20]. In both the methods, they solve the feature selection problem using zero-one integer programming [17]. AMB&B (Approximate Monotonic Branch and Bound) was introduced to combat the disadvantage of B&B by permitting evaluation functions that are not monotonic [16]. In it the bound of B&B is relaxed to generate subsets that appear below some subset violating the bound; but the selected subset should not violate the bound. BS *Beam Search* [13] is a type of best-first search that uses a bounded queue to limit the scope of the search. The queue is ordered from best to worst with the best subsets at the front of the queue. The generation procedure proceeds by taking the subset at front of the queue, and producing all possible subsets by adding a feature to it, and so on. Each subset is placed in its proper sorted position in the queue. If there is no limit on the size of the queue, BS is an exhaustive search; if the limit on the size of the queue is one, it is equivalent to SFS.

Random: The five methods found under *random* generation procedure are: LVW [27], genetic algorithm [49], simulated annealing [13], RGSS – Random Generation plus Sequential Selection [13], and RMHC-PF1 [47]. LVW generates subsets in a perfectly random fashion (it uses Las Vegas algorithm [6]). GA (genetic algorithm) and SA (simulated annealing), though have some element of randomness in their generation, follow specific procedures for generating subsets continuously. RGSS injects randomness to SFS and SBS. It generates a random subset, and runs SFS and SBS starting from the random subset. RMHC-PF1 (Random Mutation Hill Climbing – Prototype and Feature selection) selects prototypes (instances) and features simultaneously for nearest neighbor classification problem, using a bit vector that records both the prototypes and features. It uses the error rate of an 1-nearest neighbor classifier as the evaluation function. In each iteration it randomly mutates a bit in the vector to produce the next subset. All these methods need proper assignment of values to different parameters. In addition to maximum number of iterations other parameters may be required, such as for LVW the threshold inconsistency rate; for GA initial population size, crossover rate, and mutation rate; for SA the annealing schedule, number of times to loop, initial temperature, and mutation probability.

A Summary

Figure 9 shows a summary of the feature selection methods based on the 3 types of generation procedures. Complete generation procedures are further subdivided into 'Exhaustive' and 'Non-Exhaustive'; under 'Exhaustive' category a method may evaluate 'All' 2^N subsets, or it may do a 'Breadth-First' search to stop searching as soon as an optimal subset is found; under 'Non-Exhaustive' category we find 'Branch& Bound', 'Best First', and 'Beam Search' as different search techniques. Heuristic generation procedures are subdivided into 'Forward Selection', 'Backward Selection', 'Combined Forward/Backward', and 'Instance-Based' categories. Similarly, random generation procedures are grouped into 'Pure' (purely random) or 'Probabilistic' (probability of being generated varies among the subsets). The underlined methods represent their categories in the Table 2, and are implemented for an empirical comparison among the categorical methods in the next section.

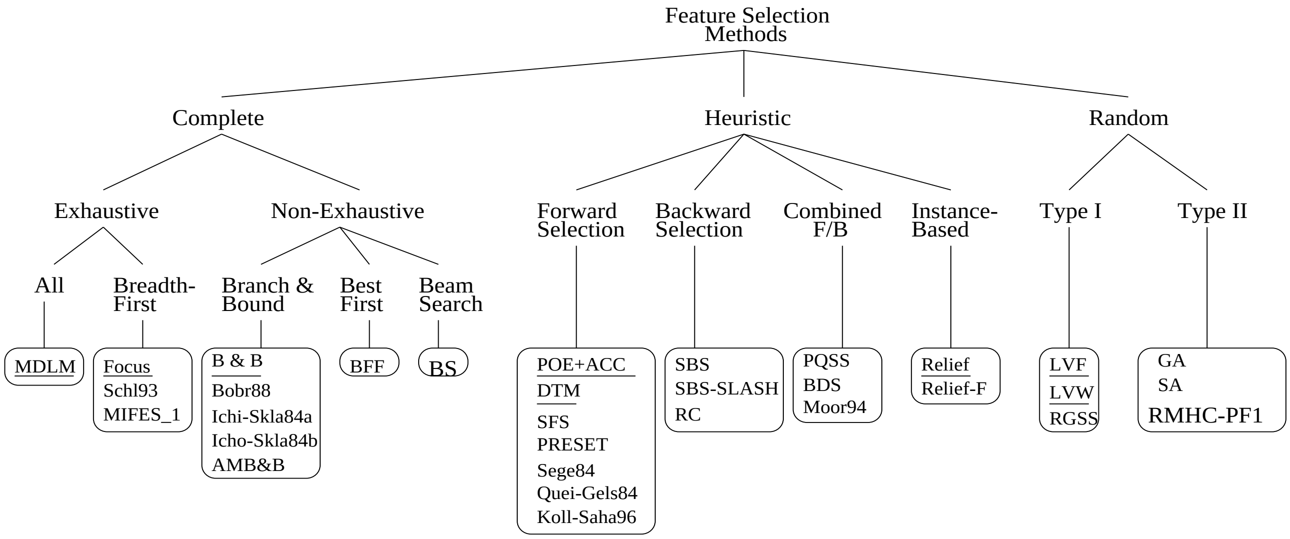

Figure 9: Summary of feature selection methods

4 An Empirical Comparison

In this section, we first discuss the different issues involved in validation procedure commonly used in the feature selection methods. Next, we briefly talk about three artificial datasets chosen for the experiments. At the end, we compare and analyze the results of running the categorical methods over the chosen datasets. As wrapper methods may produce different results for different classifiers used (this would require knowledge of the classifiers themselves), so they are excluded from the empirical comparison, and categories I, II, IV, V, VII, XI, and XII are chosen (other categories are empty).

4.1 Validation

There are two commonly used validation procedures for feature selection methods (a) using artificial datasets and (b) using real-world datasets. Artificial datasets are constructed keeping in mind a certain target concept; so all the actual relevant features for this concept are known. Validation procedures check whether the output (selected subset) is the same as the actual subset. A training dataset with known relevant features as well as irrelevant/redundant features or noise is taken. The feature selection method is run over this dataset, and the result is compared with the known relevant features. In the second procedure, real-world datasets are chosen that may be benchmark datasets. As the actual subset is unknown in this case, the selected subset is tested for its accuracy with the help of any classifier suitable to the task. Typical classifiers used are: naive Bayesian classifier, Cart [7], ID3 [38], FRINGE [36], AQ15 [30], CN2 [11], and C4.5 [39]. The achieved accuracy may be compared with that of some well-known methods, and its efficacy is analyzed.

In this section, we are trying to show an empirical comparison of feature selection methods from seven categories. In our opinion, the second method of validation is not suitable to our task because particular classifiers support specific datasets; a test with such combination of classifier and dataset may wrongly show a high accuracy rate unless *a variety of classifiers are chosen* [2] and *many statistically different datasets are used* [3]. Also, as different feature selection methods have different bias in selecting features, similar to that of different classifiers, it is not fair to use certain combinations of methods and classifiers, and try to generalize from the results that some feature selection method is better than others without considering the classifier. Fortunately many recent papers, that introduced new feature selection methods, have done some extensive tests in comparison. Hence, if a reader wants to probe further, she is referred, in this survey, to the original work. To give some intuition regarding the wrapper methods, we include one method (LVW [27]) in the empirical comparison.

We choose three artificial datasets with combinations of relevant, irrelevant, redundant features and noise to perform comparison. They are simple enough but have characteristics that can be used for testing. For each dataset a training set, commonly used in literature for comparing the results, is chosen for checking whether a feature selection method can find the relevant features.

4.2 Datasets

The three datasets are chosen in such a way that the first dataset has *irrelevant and correlated* features, the second one has *irrelevant and redundant* features, and the third one has *irrelevant*

[2] Schaffer [42] shows that there (obviously) is no single learning algorithm that will work best on all tasks.

[3] Using UC Irvine's data repository alone may not be sufficient as shown by Holte [18].

features and it is *noisy* (misclassification). Our attempt is to find out the strengths and weaknesses of the methods in these settings.

CorrAL [21]

This dataset has 32 instances, binary classes, and six boolean features $(A_0, A_1, B_0, B_1, I, C)$ out of which feature I is *irrelevant*, feature C is *correlated* to the class label 75% of the time, and the other four features are relevant to the boolean target concept: $(A_0 \wedge A_1) \vee (B_0 \wedge B_1)$.

Modified Par3+3

Par3+3 is a modified version of the original Parity3 data that has no redundant feature. The dataset contains binary classes and twelve boolean features of which three features (A_1, A_2, A_3) are relevant, (A_7, A_8, A_9) are *redundant*, and (A_4, A_5, A_6) and (A_7, A_8, A_9) are *irrelevant* (randomly chosen). The training set contains 64 instances and the target concept is the odd parity of the first three features.

Monk3 [48]

It has binary classes and six discrete features (A_1,...,A_6) only the second, fourth and fifth of which are relevant to the target concept: $(A_5 = 3 \vee A_4 = 1) \wedge (A_5! = 4 \vee A_2! = 3)$ with "!=" denoting inequality. The training set contains 122 instances, 5% of which is *noise* (misclassification).

4.3 Results and Analyses

Table 4 shows the results of running the seven categorical methods over the three artificial datasets. In the following paragraphs we analyze these results, and compare the methods based on the datasets used.

Method (RA)	**CorrAL** (A_0, A_1, B_0, B_1)	**Modified Par3+3** $(\{A_1, A_7\},\{A_2, A_8\},\{A_3, A_9\})$	**monk3** (A_2, A_4, A_5)
Relief	A_0, B_0, B_1, C	$A_1, A_2, A_3, A_7, A_8, A_9$	A_2, A_5 always & one or both of A_3, A_4
B & B	A_0, A_1, B_0, I	A_1, A_2, A_3	A_1, A_3, A_4
DTM	A_0, A_1, B_0, B_1, C	A_6, A_4	A_2, A_5
MDLM	C	$A_2, A_3, A_8, A_{10}, A_{12}$	A_2, A_3, A_5
POE+ACC	C, A_0, B_0, I	$A_5, A_1, A_{11}, A_2, A_{12}, A_3$	A_2, A_5, A_1, A_3
Focus	A_0, A_1, B_0, B_1	A_1, A_2, A_3	A_1, A_2, A_4, A_5
LVF	A_0, A_1, B_0, B_1	A_3, A_7, A_8*	A_2, A_4, A_5 at 5% I-T A_1, A_2, A_4, A_5 at 0% I-T
LVW	A_0, A_1, B_0, B_1	A_2, A_3, A_7*	A_2, A_4, A_5

Table 4: Table showing the features selected for the three datasets.
RA – Relevant Attributes
I-T – Inconsistency Threshold
This is one of many found.

Relief [22]

For CorrAL dataset, Relief selects (A_0, B_0, B_1, C) for the range [14,29] of $NoSample$. For all values of $NoSample$, Relief prefers the correlated feature C over the relevant feature A_1. The experiment over modified Par3+3 dataset clearly shows that Relief is unable to detect redundant features (features A_7, A_8, A_9 are same as A_1, A_2, A_3 respectively). But it was partially successful (depends on the value of $NoSample$) for Monk3 dataset that contains 5% noise. Experimentally we found that, for the range [15,55] of $NoSample$, it selects only the relevant features; beyond this range it either chooses some irrelevant features, or does not select some relevant features. Hence, for a dataset with no *a priori* information, the user may find it difficult to choose an appropriate value for $NoSample$. We suggest not to use a value too large or too small.

B&B [34]

For CorrAL dataset it rejects the feature B_1 as the first feature, and in the next iteration it rejects feature C. In case of Par3+3 it works well; when asked to select three features it selects (A_1, A_2, A_3), and when asked to select six it selects $(A_1, A_2, A_3, A_7, A_8, A_9)$. It failed for Monk3 dataset which has noise.

DTM [9]

As explained in Section 3.3, DTM prefers the features that are highly correlated with the class label, and in case of CorrAL dataset feature C that is correlated with the class label 75% of the time, is selected as the root. Par3+3 is a difficult task for C4.5 inductive algorithm, and it fails to find the relevant features. For the Monk3 dataset that contains noise, DTM was partially successful in choosing A_2, A_5 after pruning.

MDLM [45]

MDLM works well when all the observations are Gaussian. But none of the three datasets are perfectly Gaussian, which is the reason why MDLM largely failed. For CorrAL, it selects the subset consisting of only the correlated feature C which indicates that it prefers correlated features. For the other two datasets the results are also not good.

POE+ACC [33]

The results for this method show the rankings of the features. As seen, it was not very successful for any of the dataset. For CorrAL, it chooses feature C as the first feature, as explained in Section 3.5. When we manually input some other first feature (say A_0) then it correctly ranks the other three relevant features in the top four positions (A_0, A_1, B_0, B_1). This interesting finding points out the importance of the first feature on all the remaining features for POE+ACC. For Par3+3 and Monk3 it does not work well.

Focus [2]

Focus works well when the dataset has no noise (as in the case of CorrAL and Par3+3), but it selects an extra feature A_1 for Monk3 that has 5% noise, as the actual subset (A_2, A_4, A_5) does not satisfy the default inconsistency rate zero. Focus is also very fast in producing the results for

CorrAL and Par3+3 because the actual subsets appear in the beginning which supports breadth first search.

LVF [**28**]

LVF works well for all the datasets in our experiment. For CorrAL it correctly selects the actual subset. It is particularly efficient for Par3+3 that has three redundant features, because it can generate at least one result out of the eight possible actual subsets quite early while randomly generating the subsets. For the Monk3 dataset, it needs an approximate noise level to produce the actual subset; otherwise (default case of zero inconsistency rate) it selects an extra feature A_1.

5 Discussions and Guidelines

Feature selection is used in many application areas as a tool to remove irrelevant and/or redundant features. As is well known, there is no single feature selection method that can be applied to all applications. The choice of a feature selection method depends on various data set characteristics: (i) data types, (ii) data size, and (iii) noise. Based on different criteria from these characteristics, we give some guidelines to a potential user of feature selection as to which method to select for a particular application.

5.1 Data Types

Various data types are found, in practice, for *features* and *class labels.*
Features: Feature values can be continuous (C), discrete (D), or nominal including boolean (N). The nominal data type requires special handling because it is not easy to assign real values to it. Hence we partition the methods based on their *ablility to handle different data types (C, D, N).*
Class Label: Some feature selection methods can handle only binary classes, and others can deal with multiple (more than two) classes. Hence, the methods are separated based on their *ablility to handle multiple classes.*

5.2 Data Size

This aspect of feature selection deals with (i) whether a method can perform well for small training set, and (ii) whether it can handle large data size. In this information age, datasets are commonly large in size; hence the second criterion is more practical and of interest to the current as well as the future researchers and practitioners in this area [28]; so we separate the methods based on their *ability to handle large dataset.*

5.3 Noise

Typical noise encountered during feature selection process are (i) misclassification and (ii) conflicting data ([29], [41]). We partition the methods based on their *ablility to handle noise.*

5.4 Some Guidelines

In this subsection we give some guidelines based on the criteria extracted from these characteristics. The criteria are:

- ability to handle different data types (C: (Y/N), D: (Y/N), N: (Y/N)), where C, D, and N denote continuous, discrete, and nominal;
- ability to handle multiple (more than two) classes (Y/N);
- ability to handle large dataset (Y/N);
- ability to handle noise (Y/N); and
- ability to produce optimal subset if data is not noisy (Y/N).

Table 5 lists the capabilities regarding these five characteristics of sixteen feature selection methods that appear in the framework in Section 2. The methods having "classifier error rate" as an evaluation function are not considered because their capabilities depend on the particular classifier used. A method can be implemented in different ways, so the entries in the table are based on the presentation of the methods in the papers. The "-" suggests that the method does not discuss about the particular characteristic. A user may employ her prior knowledge regarding these five characteristics to find an appropriate method for a particular application. For example, if the data contains noise then the methods marked "Y" under the column "Noise" should be considered. If there are more than one method satisfying the criteria mentioned by the user, then the user needs to have a closer look at the methods to find the most suitable among them, or design a new method that suits the application.

6 Conclusion and Further Work

We give a definition of feature selection after discussing many existing definitions. Four steps of a typical feature selection process are recognized: generation procedure, evaluation function, stopping criterion, and validation procedure. We group the generation procedures into three categories: complete, heuristic, and random, and the evaluation functions into five categories: distance, information, dependence, consistency, and classifier error rate measures. Thirty two existing feature selection methods are categorized based on the combinations of generation procedure and evaluation function used in them. Methods under each category are described briefly, and a representative method is chosen for each category that is described in detail using its pseudo code and a hand-run over a dataset (a version of CorrAL). These representative methods are compared using three artificial datasets with different properties, and the results are analysed. We discuss some data set characteristics, and give some guidelines regarding how to choose a particular method if the user has prior knowledge of the problem domain.

This paper is a survey of feature selection methods. Our finding is that a number of combinations of generation procedure and evaluation function have not yet been attempted (to the best of our knowledge). This is quite interesting considering the fact that feature selection has been the focus of interest of various groups of researchers for the last few decades. Our comparison of the categorical methods reveals many interesting facts regarding their advantages and disadvantages in handling different characteristics of the data. The guidelines in Section 5 should be useful to the users of feature selection who would have been otherwise baffled by the sheer number of methods. Properly used, these guidelines can be of immense practical value in choosing the suitable method.

Although feature selection is a matured research area, researchers still try to find better methods to make their classifiers more efficient. In this scenario the framework which shows the unattempted

Category	Method	Ability to handle/produce: Data Types C D N	Multiple Classes	Large Dataset	Noise	Optimal Subset
I	Relief	Y Y Y	N	Y	Y	N
	Relief-F	Y Y Y	Y	Y	Y	N
	Sege84	N Y Y	N	-	-	N
	Quei-Gels84	N Y Y	Y	-	-	N
II	B & B	Y Y N	Y	-	-	Y^{++}
	BFF	Y Y N	Y	-	-	Y^{++}
	Bobr88	Y Y N	Y	-	-	Y^{++}
IV	DTM	Y Y Y	Y	Y	-	N
	Koll-Saha96	N Y Y	Y	Y	-	N
V	MDLM	Y Y N	Y	-	-	N
VII	POE+ACC	Y Y Y	Y	-	-	N
	PRESET	Y Y Y	Y	Y	-	N
XI	Focus	N Y Y	Y	N	N	Y
	Schl93	N Y Y	Y	N	N	Y^{++}
	$MIFES_1^+$	N N N	Y	N	N	Y
XII	LVF	N Y Y	Y	Y	Y^*	Y^{**}

Table 5: Feature Selection Methods and their Capabilities.
+ It can handle only boolean features.
++ If certain assumptions are valid.
$*$ User is required to provide the noise level.
$**$ Provided there are enough resources.

combinations of generation procedures and evaluation functions will be helpful. By trying out particular implementation of evaluation function and generation procedure combinations previously used in some categories, we can design new and efficient methods for some unattempted combinations. It is obvious that not all combinations will be efficient. The combination of consistency measure and heuristic generation procedure in the category 'X' may give some good results. One may also test different combinations that previously exist, such as branch and bound generation procedure and consistency measure in category XI.

Acknowledgements

The authors wish to thank the two referees and Prof Hiroshi Motoda of Osaka University for thier valuable comments, constructive suggestions, and thorough reviews on the earlier version of the paper. Thanks also go to Lee Leslie Leon, Sng Woei Tyng, and Wong Siew Kuen for implementing some feature selection methods and analysing their results.

References

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] H. Almuallim and T. G. Dietterich. Learning with many irrelevant features. In *Proceedings of Ninth National Conference on AI*, pages 547–552, 1992.

[3] H. Almuallim and T. G. Dietterich. Learning boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1-2):279–305, November 1994.

[4] M. Ben-Bassat. Pattern recognition and reduction of dimensionality. In P. R. Krishnaiah and L. N. Kanal, editors, *Handbook of Statistics*, pages 773–791. North Holland, 1982.

[5] L. Bobrowski. Feature selection based on some homogeneity coefficient. In *Proceedings of Ninth International Conference on Pattern Recognition*, pages 544–546, 1988.

[6] G. Brassard and P. Bratley. *Fundamentals of Algorithms.* Prentice Hall, New Jersy, 1996.

[7] G. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth International Group, Belmont, California, 1984.

[8] J. P. Callan, T. E. Fawcett, and E. L. Rissland. An adaptive approach to case-based search. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 803–808, 1991.

[9] C. Cardie. Using decision trees to improve case-based learning. In *Proceedings of Tenth International Conference on Machine Learning*, pages 25–32, 1993.

[10] R. Caruana and D. Freitag. Greedy attribute selection. In *Proceedings of Eleventh International Conference on Machine Learning*, pages 28–36, New Brunswick, New Jersy, 1994. Morgan Kaufmann.

[11] P. Clark and T. Niblett. The cn2 induction algorithm. *Machine Learning*, 3:261–283, 1989.

[12] P. A. Devijver and J. Kittler. *Pattern Recognition : A Statistical Approach.* Prentice Hall, 1982.

[13] J. Doak. An evaluation of feature selection methods and their application to computer security. Technical report, Davis CA: University of California, Department of Computer Science, 1992.

[14] P. Domingos. Context-sensitive feature selection for lazy learners. To appear in Artificial Intelligence Review, 1996.

[15] B. S. Duran and P. K. Odell. *Cluster analysis - a survey.* Springer-Verlay, 1974.

[16] Iman Foroutan and J. Sklansky. Feature selection for automatic classification of non-gaussian data. *IEEE Transactions on Systems, Man, and Cybernatics*, SMC-17(2):187–198, 1987.

[17] A. M. Geoffrion. Integer programming by implicit enumeration and balas, method. *SIAM Review*, 9:178–190, April 1967.

[18] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–90, 1993.

[19] M. Ichino and J. Sklansky. Feature selection for linear classifier. In *Proceedings of the Seventh International Confernce on Pattern Recognition*, volume 1, pages 124–127, July-Aug 1984.

[20] M. Ichino and J. Sklansky. Optimum feature selection by zero-one programming. *IEEE Trans. on Systems, Man and Cybernetics*, SMC-14(5):737–746, September/October 1984.

[21] G. H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 121–129, 1994.

[22] K. Kira and L. A. Rendell. The feature selection problem : Traditional methods and a new algorithm. In *Proceedings of Ninth National Conference on AI*, pages 129–134, 1992.

[23] R. Kohavi and D. Sommerfield. Feature subset selection using the wrapper method: Overfitting and dynamic search space topology. In *Proceedings of First International Conference on Knowledge Disclovery and Data Mining*, pages 192–197. Morgan Kaufmann, 1995.

[24] D. Koller and M. Sahami. Toward optimal feature selection. In *Proceedings of International Conference on Machine Learning*, 1996.

[25] I. Kononenko. Estimating attributes : Analysis and extension of RELIEF. In *Proceedings of European Conference on Machine Learning*, pages 171–182, 1994.

[26] P. Langley. Selection of relevant features in machine learning. In *Proceedings of the AAAI Fall Symposium on Relevance*, pages 1–5, 1994.

[27] H. Liu and R. Setiono. Feature selection and classification - a probabilistic wrapper approach. In *Proceedings of Ninth International Conference on Industrial and Engineering Applications of AI and ES*, 1996.

[28] H. Liu and R. Setiono. A probabilistic approach to feature selection - a filter solution. In *Proceedings of International Conference on Machine Learning*, pages 319–327, 1996.

[29] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors. *Machine Learning: An Artificial Intelligence Approach (Vol. II)*, chapter The effect of noise on concept learning. Morgan Kaufmann, San Mateo, CA, 1986.

[30] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The *aq*15 inductive learning system: An overview and experiments. Technical Report UIUCDCS-R-86-1260, University of Illinois, July 1986.

[31] M. Modrzejewski. Feature selection using rough sets theory. In P. B. Brazdil, editor, *Proceedings of the European Conference on Machine Learning*, pages 213–226, 1993.

[32] A. W. Moore and M. S. Lee. Efficient algorithms for minimizing cross validation error. In *Proceedings of Eleventh International Conference on Machine Learning*, pages 190–198, New Brunswick, New Jersy, 1994. Morgan Kaufmann.

[33] A. N. Mucciardi and E. E. Gose. A comparison of seven techniques for choosing subsets of pattern recognition. *IEEE Transactions on Computers*, C-20:1023–1031, September 1971.

[34] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature selection. *IEEE Transactions on Computers*, C-26(9):917–922, September 1977.

[35] A. L. Oliveira and A. S. Vincentelli. Constructive induction using a non-greedy strategy for feature selection. In *Proceedings of Ninth International Conference on Machine Learning*, pages 355–360, Aberdeen, Scotland, 1992. Morgan Kaufmann.

[36] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, 1(1):81–106, 1986.

[37] C. E. Queiros and E. S. Gelsema. On feature selection. In *Proceedings of Seventh International Conference on Pattern Recognition*, volume 1, pages 128–130, July-Aug 1984.

[38] J. R. Quinlan. Induction of decision trees. In *Machine Learning*, pages 81–106. Morgan Kaufmann, 1986.

[39] J. R. Quinlan. *C4.5 : Programs for Machine Learning.* Morgan Kaufmann, San Mateo, California, 1993.

[40] J. Rissanen. Modelling by shortest data description. *Automatica*, 14:465–471, 1978.

[41] C. Schaffer. Overfitting avoidance as bias. *Machine Learning*, 10(2):153–178, 1993.

[42] C. Schaffer. A conservation law for generalization performance. In *Proceedings of Eleventh International Conference on Machine Learning*, pages 259–265, New Brunswick, NJ, 1994. Morgan Kaufmann.

[43] J. C. Schlimmer. Efficiently inducing determinations : A complete and systematic search algorithm that uses optimal pruning. In *Proceedings of Tenth International Conference on Machine Learning*, pages 284–290, 1993.

[44] J. Segen. Feature selection and constructive inference. In *Proceedings of Seventh International Conference on Pattern Recognition*, pages 1344–1346, 1984.

[45] J. Sheinvald, B. Dom, and W. Niblack. A modelling approach to feature selection. In *Proceedings of Tenth International Conference on Pattern Recognition*, volume 1, pages 535–539, June 1990.

[46] W. Siedlecki and J Sklansky. On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence*, 2:197–220, 1988.

[47] D. B. Skalak. Prototype and feature selection by sampling and random mutation hill-climbing algorithms. In *Proceedings of Eleventh International Conference on Machine Learning*, pages 293–301, New Brunswick, 1994. Morgan Kaufmann.

[48] et al. Thrun, S. B. The monk's problem: A performance comparison of different algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, December 1991.

[49] H. Vafaie and I. F. Imam. Feature selection methods: genetic algorithms vs. greedy-like search. In *Proceedings of International Conference on Fuzzy and Intelligent Control Systems*, 1994.

[50] L. Xu, P. Yan, and T. Chang. Best first strategy for feature selection. In *Proceedings of Ninth International Conference on Pattern Recognition*, pages 706–708, 1988.