Exclude: A New Heuristics Based Algorithm for Excluding Irrelevant Features in Inductive Learning

Saleh M. Abu-Soud
Department of Software Engineering
Princess Sumaya University for Technology
P.O. Box (1438) Amman 11941
JORDAN
abu-soud@psut.edu.jo

Abstract - In this paper, a new algorithm, called Exclude, for excluding irrelevant features has been suggested. This algorithm is stand-alone which means that it can be applied by any inductive learning algorithm. It increases the efficiency of the inductive algorithm which is applied by, reduces number of rules, and simplifies the resulted rules (fewer conditions in their LHS). All this happens with maintaining the accuracy at acceptable levels. In Exclude algorithm, a new heuristic function has been suggested and tested with hundreds of experiments on many datasets with several known inductive algorithms. These experiments are categorized into three categories: the first set of experiments test the suggested approach on induction without feature subset selection, while the second set tests this approach on several decision tree and non-decision tree inductive learning algorithms as ILA, ID3, and AQ. The third set compares the results of this approach with the results of some other feature selection methods as Wrapper, PSORSFS and Relief-F. The results obtained are encouraging and showed that the suggested approach is powerful and comparable with other methods.

Keywords- Feature selection, Heuristics, inductive learning, Irrelevant Features.

I. Introduction

SUPERVISED machine learning algorithms usually deal with complex tasks which contain a large number of features (attributes), where many of these features may be un-useful for the learning process. These features are called irrelevant features. The existence of such features does not contribute in enhancing the quality of the output of the inductive algorithms. On the contrary; their existence definitely will affect negatively the efficiency of the algorithm and the quality of its output, where in this case, the algorithm produces extra rules and these rules may be even more complex than usual.

The process of focusing on the most important features and excluding the irrelevant ones is of utmost importance to improve performance on some dependent measures such as learning speed, number and simplicity (the generality) of rules produced by the learner, and of course, the classification power of these rules which represents the accuracy of the rules in classifying unseen examples.

There are a lot of feature selection algorithms in the literature. These algorithms are either stand-alone1 or tailored in an inductive learning algorithm. Each algorithm tackles this problem in different way. Many algorithms fall into what is called the filter approach, which selects features using a preprocessing step. The main drawback of this approach is that it ignores the effect of the selected features on the performance of the induction algorithm used [1]. One of the algorithms that is located in this category is the FOCUS algorithm [2, 3] which originally applied on noise-free binary domains. It checks all features and selects the minimal subset that is sufficient to determine the class value for all examples in the training set. The Relief algorithm [4, 5] is another example of this category. It is a randomized algorithm that runs also on binary classification problems, which attempts to find out all relevant attributes. It does that by assigning a relevance weight to each attribute to target class value. Relief-F [6] is a general case of Relief, which work on multiple classes.

One of the most interesting stand-alone approaches is the wrapper approach [1, 7]. In this approach, the feature selection algorithm performs a forward search [best-first or Hill-climbing] in the space of possible parameters for a good subset using the induction algorithm itself as part of the evaluation function. This is done by running an inductive algorithm on the dataset and using the estimated accuracy of the resulting classifier as its metric. OBLIVION algorithm [8] is an example of this approach which combines the wrapper idea with the nearest-neighbor method, which assigns to new examples the class of the nearest case stored in cash memory during the learning process.

Another approach for feature selection is the usage of heuristic search algorithms like hill-climbing, greedy search, and best-fist search [14, 15]. All these algorithms share a common characteristic in that they search a space of attributes that are generated by an operator, and select the best attribute that classifies more classes or that improves accuracy over the previously selected one. All these methods in this category work either by selection, i.e. starting with an attribute and then adding other attributes or by elimination that is to start with a

[1]The induction algorithm calls it as a subprogram

set of attributes and then removing one or more attributes until we reach the best state.

PSORSFS is another interesting algorithm which is a Rough set-based Feature Selection algorithm [20].In this approach, each feature subset can be seen as a point or position in such a space. If there are N total features, then there will be 2N kinds of subset, different from each other in the length and features contained in each subset. The optimal position is the subset with least length and highest classification quality. Now we put a particle swarm into this feature space, each particle takes one position. The particles fly in this space, their goal is to fly to the best position. Over time, they change their position, communicate with each other, and search around the local best and global best position. Eventually, they should converge on good, possibly optimal, positions. It is this exploration ability of particle swarms that should better equip it to perform feature selection and discover optimal subsets.

Some feature selection methods are embedded or tailored into certain induction algorithms. ID3 [9], C4.5 [10], and CART [11] for example, use an evaluation function in each stage of these algorithms to choose the attribute that is most likely to be best in discrimination of classes. Relief algorithm [5] applies a complex evaluation function to select features and uses ID3 to induce the decision tree using these selected features. Other embedded methods like [12, 13] use separate-and-conquer methods in the learning process by using an evaluation function to select the features that are used to distinguish a class from others.

The main problem addressed in this paper is to exclude the most irrelevant features from the dataset under consideration in the supervised learning process. A proposed algorithm called Exclude has been built to solve this problem. This model falls under the filter approach i.e. it excludes the most possible irrelevant attributes as a preprocessing step before running the inductive learning algorithm on the optimized dataset. It overcomes the main drawback of the filter approach that is it takes into consideration the accuracy when excluding irrelevant attributes. To study the proposed algorithm from different points of view, three sets of experiments had been conducted. The first set of experiments show that the proposed system resulted in reducing number of rules and increasing their simplicity which in turn enhances the efficiency of the learning process, with keeping the accuracy as in the original data as possible. In the second set of experiments, this system had been applied on some known inductive algorithms as: ID3 [9], ILA [16], and AQ [17], while in the third set of experiments, the proposed system had been compared with other well-known methods as: Relief-F [6] and Wrapper [1]. The experiments showed that the proposed system enhances the efficiency of the induction algorithm and the quality of the resulted rules significantly and the results obtained are comparable to other systems if not better.

II. Feature subset selection

2.1 The problem

The task of inductive learning is divided into two main phases: selecting the attributes to be used in the induction process, and the way of combining these attributes to best classify the class. It happens that the problem under consideration has a lot of attributes and many of these attributes are irrelevant to the domain of the problem which if considered, will increase the complexity of the induction process. So, it is better to exclude these attributes from the induction process at all.

Excluding irrelevant attributes yields in: 1) increasing the efficiency of the induction process, 2) resulting in less number of rules that are needed to classify the desired class, 3) simplifying the resulting rules. This means to produce more general rules; that is, less number of conditions in the LHS of the rules. But, all these gained advantages must not be at the expense of the induction accuracy. One should not forget that the best induction algorithm is the one that produces less number of most general rules with high accuracy. The accuracy before excluding the irrelevant attributes is the upper bound of the accuracy after excluding them, and it should be close to it as possible.

So the following four factors will be considered in our model:

i. *Number of rules generated*: number of generated rules less than that if these attributes are not excluded.
ii. *Simplicity of generated rules*: also here the generated rules after excluding the irrelevant attributes should be simpler than those generated before the excluding process.
iii. *Speed of inductive algorithm*: if the above two factors are met, then the induction process should be faster.
iv. *Accuracy of generated rules*: this factor implies that the accuracy of the resulted rules should not be lower than the accuracy of the original dataset (before excluding the irrelevant features).

2.2 Definitions

In order to well describe our model, some definitions are needed and described here.

Definition (1) duplicated examples

Two examples in a dataset are said to be duplicated if they have identical values of their attributes and the class attribute, that is: two examples Ψ and δ are duplicated with respect to a class Ω iff attributes (Ψ) = attributes(δ) and class attribute(Ψ) = class attribute(δ). For example, the following two examples are duplicated; knowing that α and β are the attributes, while Ω is the class attribute:

Ψ: α , β , Ω, and δ: α , β , Ω. In this case, all duplicated examples should be eliminated keeping only one of them. Number of duplicates is denoted by the variable C_d. C_d in this case is 2.

Definition (2) contradicted examples

Two examples are said to be contradicted if their attributes are identical and have different class attribute, i.e. the two examples Ψ and δ contradicted each other iff attributes(Ψ) = attributes(δ) and class attribute(Ψ)$\neq$class attribute(δ).

Examples Ψ and δ contradict each other in the following case, knowing that α and β are the attributes, while μ is the class attribute:
Ψ: α , β , Ω, and δ: α , β , μ. In this case, both examples should be removed from the dataset. Number of contradicts is denoted by C_c and its value in this case is 2.

To well illustrate these definitions, consider the case in Example (1) below.

Example (1): suppose we have the dataset as in Table I.
As depicted in Table I, this dataset consists of three attributes: *size, color, and shape*, and one class attribute *decision*. Let's assume that the attribute *shape* is an irrelevant attribute (later we will know how to know whether the attribute is irrelevant or not) and needs to be excluded from the dataset. The resulted dataset is depicted in Table II.

Table I. Object Classification Training Set[18].

Example No.	size	color	shape	**decision**
1	medium	blue	brick	**yes**
2	small	red	wedge	**no**
3	small	red	sphere	**yes**
4	large	red	wedge	**no**
5	large	green	pillar	**yes**
6	large	red	pillar	**no**
7	large	green	sphere	**yes**

Table II. The resulted dataset after excluding the attribute shape.

Example No.	size	color	**decision**
1	medium	blue	**yes**
2	small	red	**no**
3	small	red	**yes**
4	large	red	**no**
5	large	green	**yes**
6	large	red	**no**
7	large	green	**yes**

It is noted from Table II that examples 2 and 3 are contradictions, so they both should be removed. It is noted also that examples 4 and 6 and examples 5 and 7 are duplicates, so we should keep one example of each group. The resulted dataset is as shown in Table III. Note that in this example C_c=2 and C_d=4.

Definition (3) missed out classes

For a dataset δ with attributes $\alpha_1, \alpha_2, \ldots \alpha_n$ and a class attribute with values $\beta_1, \beta_2, \ldots \beta_m$, if some α_i where i=1 to n-1 are excluded from δ and this results in disappearing of some β_j, where$1 \leq j \leq m$, then this case is called *missed out classes*.

Table III. The resulted dataset after manipulating duplicates and contradicts.

Example No.	size	color	**decision**
1	medium	blue	**yes**
4	large	red	**no**
5	large	green	**yes**

This means that if some attributes when excluded from a dataset, results in losing some class values completely from the dataset, which in turn results in disappearing some class categories entirely from the dataset. To illustrate this case, consider the season dataset [19] shown on Table IV. Note that the class attribute is season with 4 values: summer, autumn, winter, and spring.

Table IV. Season dataset [19].

weather	trees	temperature	**season**
rainy	yellow	average	**autumn**
rainy	leafless	low	**winter**
snowy	leafless	low	**winter**
sunny	leafless	low	**winter**
rainy	leafless	average	**autumn**
rainy	green	high	**summer**
rainy	green	average	**spring**
sunny	green	average	**spring**
sunny	green	high	**summer**
sunny	yellow	average	**autumn**
snowy	green	low	**winter**

Let's exclude the temperature attribute. The resulted dataset after manipulating duplications and contradictions is shown in Table V.

As depicted in Table V, the season class attribute has lost two of its values i.e. summer and spring, while the dataset in Table III has not this situation.

It is worthy to mention that attributes are of three types with respect to irrelevancy:

1. **Strongly irrelevant attribute**: it is the attribute that if excluded from a dataset, enhancements may be achieved in all or some of the following factors: performance, number of resulted rules, average number of conditions in the resulted rules, but the induction power (accuracy) of the resulted rules on the original dataset remains 100%.
2. **Weakly irrelevant attribute**: it is the attribute that if excluded from a dataset, enhancements may be achieved in all or some of the following factors: performance, number of resulted rules, average number of conditions in the resulted rules, but the induction power (accuracy) of the resulted rules on the original dataset is less than 100% to an acceptable level. The degree of its weakness is measured by the farness of the resulted accuracy from

100%.The required "*farness*" (in the model is denoted by *PredefinedRatio*) can be specified by the user.

Table V. Processed Season dataset after excluding temperature attribute

weather	trees	**season**
rainy	yellow	**autumn**
sunny	yellow	**autumn**
snowy	leafless	**winter**
sunny	leafless	**winter**
snowy	green	**winter**

3. **Relevant attribute**: an attribute is relevant if it is not strongly or weakly irrelevant attribute. That is if it is excluded from the dataset, results in a significant reduction in the induction power (accuracy) regardless of other factors. This type of attributes is skipped out by the model and not excluded at all.

2.3 The heuristic functions

The most guaranteed way to decide whether an attribute is relevant or not, is to remove it from the dataset under consideration and perform the learning process on the resulted dataset and observe the difference. But this process is costly and is not practical. So it is necessary to discover a way of deciding on the relevancy of an attribute prior and without performing the learning process, i.e. to find good heuristics that discover the most likely irrelevant attributes and guide the exclusion process. The generation process of these heuristics needs thorough empirical studies on as many datasets as possible whose main aim is to find general heuristic rules that will be applied on datasets to guide the irrelevant feature excluding process.

We will begin with a discussion of the main concepts that the suggested heuristics are based on. We have studied tens of datasets intensively and noticed some observations. The first observation came out when we studied the relationship between an attribute under consideration say $attr_x$ in one side, and other attributes in the same dataset plus the decision class of the dataset in the other side (let us call them $rest_x$), we noticed that for different values of $attr_x$, if we have as much similar values of $rest_x$ as possible then $attr_x$ is an important (relevant) attribute if deleted will degrade the classification power of the resulted rules. To illustrate this observation, consider the following subset of a dataset with attributes attr1, attr2, attr3, and the decision class decision:

attr1	attr2	attr3	decision
a	x	y	d1
b	x	y	d1
c	x	y	d1
d	x	z	d2
e	w	y	d3

We can easily get the following three rules: ***if attr2 =x and attr3 =y then d1*** , ***if attr1 =d then d2, and if attr1 =e then d3*** that cover all the examples in the dataset with 100% accuracy. Now suppose that we want to exclude the attribute attr1, (i.e. attr1 is $attr_x$ and attr2, attr3, and decision are $rest_x$) the resulted dataset will look like the following:

attr2	attr3	decision
x	y	d1
x	y	d1
x	y	d1
x	z	d2
w	y	d3

We have three duplicated rows (i.e. C_d=3), and according to definition 1, these must be eliminated and reduced to one as follows:

attr2	attr3	decision
x	y	d1
x	z	d2
w	y	d3

From the resulted dataset, one can get the following three rules: ***if attr2 =x and attr3 =y then d1*** , ***if attr3 =z then d2, and if attr2 =w then d3*** that cover all the examples in the dataset with also the same accuracy on the original dataset. This means that attribute attr1 is not important (irrelevant) and if excluded, it will not increase number of rules and will not degrade the classification power. This means that attr1 is not important and if deleted will not make significant changes in the resulted rules, i.e. attr1 is irrelevant. It is worthy to say that it is to our benefit to have as much duplicates as possible because the dataset will be smaller with almost same or less rules without degrading the classification power significantly.

Let's repeat the process but on att2. The following dataset will be resulted:

attr1	attr3	decision
a	y	d1
b	y	d1
c	y	d1
d	z	d2
e	y	d3

The following 5 rules will be generated that cover all the examples in the dataset: ***if attr1 =a then d1, if attr1 =b then d1, if attr1 =c then d1, if attr1 =d then d2, and if attr1 =e then d3.*** It is noted here that excluding attr2 has resulted in no duplicates in the new dataset (i.e. C_d=0) and has increased number of generated rules (from 3 to 5).

Another observation that has been noticed is discussed in the following example. Consider the following subset of a dataset:

attr1	attr2	attr3	decision
a	x	y	d1
b	x	y	d2
c	f	y	d3
d	z	w	d4

The following is the minimum set of rules that cover this dataset subset with 100% accuracy:
If attr1 = a then d1, ***If attr1 = b then d2***, ***If attr1 = c then d3***, and ***If attr1 = d then d4***.

Let's delete attr1 from the dataset. The following dataset will be resulted:

attr2	attr3	decision
x	y	d1
x	y	d2
f	y	d3
z	w	d4

The resulted dataset contains three contradicted examples (i.e. C_c = 3). According to definition 2, these examples should be removed completely from the dataset. The resulted dataset becomes as follows:

attr2	attr3	decision
f	y	d3
z	w	d4

With two rules ***if attr2 = f then d3*** and ***if attr2 = z then d4.*** If we apply these two rules on the original dataset it will cover only two examples (i.e. two out of four, that is the accuracy is 50%). Beside this low accuracy, it is noted that we have lost two decision values d1 and d2. This means that attr1 is important (relevant) and it is not wise to be deleted. Actually, the relevancy of an attribute is stronger if we have as many contradictions as possible, because this will cause more loss of useful examples that result in generating rules with poor classification power.

So we prefer to exclude the attribute that causes more duplication and fewer contradictions as possible. To combine these two factors together, we consider the ratio of contradictions to duplications; i.e. $(C_c / C_d) \in C_d \neq 0$ which will be denoted by the variable C_cC_dRatio. Note that if C_d= 0, this means that deleting an attribute will not cause any duplications which means that this attribute is relevant and should not be excluded in any way, so it will be skipped from further processes of the algorithm. The value of C_cC_dRatio ranges from 0 to (NumberOfExamples-2)/2 (-2 because there must be at least 2 duplicated examples in the dataset). C_cC_dRatio points, in some way, to the degree of irrelevancy. So, the smaller value of C_cC_dRatio is better. Actually this ratio can be used as a stopping criterion of the algorithm by comparing it to a predefined value that can be input to the algorithm by the user (*PredefinedRatio* variable will be used in the algorithm to store this predefined value).

It is noted also, as seen in the last example above that in some cases deleting an attribute causes a loss in some decision values completely from the dataset. This attribute is strongly relevant and should not be excluded from the dataset and should be skipped from further processes of the algorithm. The Boolean variable *MissDecisionClassValue* is used to denote this case, it will be assigned either by true or false if a loss happens or no loss exists respectively.

As the main aim of inductive learning algorithms is to maximize their classification power on unseen test examples (which is called the accuracy of the algorithm), we categorize attributes to the above three types according to accuracy and these types will guide the suggested model to exclude or not exclude attributes. The ideas in the previous discussion in addition to these three types are formulated in Definition (4) as the heuristic functions on which the suggested model will be based on.

Definition (4) The Heuristic Functions

1. An attribute is **strongly relevant** iff after it is excluded, number of duplicates in the resulted dataset = 0, i.e C_d = 0 or its exclusion causes a *missed out class value* case (denoted by *MissDecisionClassValue)*. In this case, it should not be excluded from the dataset.
2. An attribute is **strongly irrelevant** iff after it is excluded, number of contradicts in the resulted dataset = 0, i.e C_c = 0. In this case, it should be excluded immediately from the dataset.
3. An attribute is **weakly irrelevant** iff after it is excluded number of contradicts is greater than duplicates, i.e the ratio $C_c / C_d > 0$. This ratio will be denoted hereafter by C_cC_dRatio. It ia always positive. The degree of its relevancy is measured by the farness of the resulted ratio from 0, i.e. as this ratio is closer to 0 it is considered irrelevant and is considered relevant if it is far from 0. The required "*farness*" (in the model is denoted by *PredefinedRatio*) can be specified by the user. So the decision of exclusion of the attribute under consideration based on the *PredefinedRatio* that will be entered by the user.

III. Suggested Model

3.1 Discussion

The proposed model suggests that the process of determining which attributes are irrelevant and should be removed from the dataset must be done prior to the learning process. So it emphasizes on studying the dataset under consideration and processes it in a way to exclude the most irrelevant attributes that enhances the above mentioned four factors, i.e. speed, accuracy, number and simplicity of generated rules. The model works on noise free datasets with discrete values.

3.2 The *Exclude* Algorithm

The algorithm starts with excluding the attributes temporarily one by one and for each excluded attribute, all duplicates and contradicts are counted and eliminated from the resulted dataset as depicted in the definitions. This process is repeated for all attributes. At the end of this iteration, the attribute with the minimum contradicts(C_c)/duplicates(C_d) ratio (C_cC_dRatio) is deleted permanently from the dataset and the above process is repeated on the resulted dataset and so on until one attribute remains in the dataset or until the predefined contradicts/duplicates ratio (*PredefinedRatio*) is reached. It conducts the best first search method.

Please note that the C_cC_dRatio is computed as follows: if, for example, for a dataset $C_c = 3$ and $C_d = 50$ then $(C_c / C_d) = 0.06$. This means that this case is 0.06 far from 0 (the perfect case). But in the literature, it is used to use 100% to denote completeness and perfection. So, to be compatible with what is used in the literature, we multiply this ratio by 100 and subtract the result from 100. So, the resulted ratio in the example became 100-0.06*100, which is 94%. Both values have the same meaning, but the later value is more meaningful and more readable. This means that the C_cC_dRatio with higher values will be preferred and considered instead of smaller ones.

The steps of the algorithm are listed as shown in Algorithm 1.

```
Algorithm 1: Exclude
Input: Dataset D that contains n attributes, where n ≥ 2 and one decision class
Output: Dataset D_out which is D after excluding all irrelevant attributes
Process:
 1. $i = 1$
 2. $n$ = number of attributes in D
 3. Input PredefinedRatio [0 .. 100]
 4. $C_cC_dRatio = \infty$
 5. $D_{temp} = D$
 6. Do While ($i < n$) and ($C_cC_dRatio \geq PredefinedRatio$)
      6.1. $MaxC_cC_dRatio = -\infty$
      6.2. MissDecisionClassValue = false
      6.3. repeat
             6.4.1.  Remove attribute($i$) from $D_{temp}$ along with its data
             6.4.2.  $C_d$ = number of duplicates in $D_{temp}$
             6.4.3.  If ($C_d = 0$) or (MissDecisionClassValue = true) then go to step 6.4.10.
             6.4.4.  Eliminate duplicates from $D_{temp}$
             6.4.5.  $C_c$ = number of contradicts in $D_{temp}$
             6.4.6.  Eliminate contradicts from $D_{temp}$
             6.4.7.  If (MissDecisionClassValue = true) then go to step 6.4.10.
             6.4.8.  $C_cC_dRatio = 100-((C_c / C_d)*100$ with upper limit = 100)
             6.4.9.  If ($C_cC_dRatio > MaxC_cC_dRatio$) then ($MaxC_cC_dRatio = C_cC_dRatio$) and ($Max_i = i$)
             6.4.10. Restore attribute($i$) into $D_{temp}$ along with its data
             6.4.11. Increase $i$ by 1
           Until ($i > n$)
      6.4. Remove attribute ($Max_i$) from $D_{temp}$ permanently along with its data
      6.5. Eliminate Duplicates and Contradicts from $D_{temp}$
      6.6. $C_cC_dRatio = \infty$
      6.7. $i = 1$
      6.8. $n = n - 1$
    end while
 7. $D_{out} = D_{temp}$
 8. END
```

IV. An Illustrative Example

To best understand the suggested algorithm, let's go through a simple illustrative example using the object classification dataset mentioned above and shown in Table I. ILA algorithm [16] will be used to check the feasibility of the model.

Applying ILA on this dataset generates 5 rules with 1.2 average conditions on their LHS and 0.0649 second as their execution time. The list of resulted rules and other details are shown in Fig. 1.

Applying the *Exclude* algorithm, it will start with temporarily excluding the attributes one by one and calculating all related parameters for each excluded attribute and consider the attribute with maximum C_cC_dRatio to be deleted permanently. Let's illustrate the result of excluding the first attribute i.e. Size. The resulted dataset is shown in Table VI. It is noted from the table that there are two duplicated examples 2 and 4 i.e. C_d=2. According to the algorithm, one of them must remain and the rest should be deleted. The result is shown in Table VII.

It is noted that no contradictions are resulted i.e. C_c=0. This results in 100 *in* C_cC_dRatio. We repeat this process for the rest of attributes. Table VIII shows the results of this process for all attributes.

As noted in Table VIII, the attribute Size has the maximum C_cC_dRatio, so it will be permanently deleted from the dataset. If we apply ILA on the dataset without Size attribute, we will get 5 rules with 1.2 as the average conditions on their LHS and .0213 second as their execution time. The accuracy of applying these 5 rules on the original dataset is 100%; i.e. they classified all examples successfully which means that this attribute is strongly irrelevant. Let's continue to see the behavior of the algorithm on the rest of attributes. Now the dataset in Table VII becomes the dataset under consideration. Applying the algorithm on it will result in what appeared in Table IX.

It is noted from Table IX that Color is the attribute with the maximum C_cC_dRatio (actually it is the only attribute that can be considered). Allying ILA on the dataset in Table VII after excluding the Color attribute, results in getting 3 rules with 1 as the average conditions on their LHS and .0026 second as their execution time. The accuracy of applying these 5 rules on the original dataset is 71.429% which means that the Color attribute is not fully irrelevant (actually it is 28.571% relevant) even though we got less number of rules with less average conditions and execution time, but the accuracy at the end remains a significant factor in inductive learning algorithms. So we can assume a predefined ratio to stop the process to certain level of accuracy. The final results are as appeared in Fig. 2.

V. Experiments and Results

To test *Exclude* from different perspectives, three categories of experiments have been conducted on the suggested algorithm. The first set of experiments aims to test internal behavior of *Exclude* on three datasets with different number of attributes and examples; namely: Object Classification, Monk1, and Cars. The second set of experiments tests the feasibility of using *Exclude* with different inductive learning algorithms; namely: ID3 [9], and AQ [17] in addition to ILA [16]. The third set aims to test *Exclude* against similar feature selection approaches. Two algorithms are used for this purpose; one from the same category of the suggested algorithm i.e. the filter approach that is the Relief-F in which the whole feature selection process is done prior to the run of the algorithm. The second algorithm is the wrapper approach [1] which lies into another category of approaches in which the feature subset selection is done using the inductive learning algorithm as a black box as part of the evaluation function.

All experiments in this section have been conducted on an IBM compatible machine with 64-bit Windows 7 professional OS, 3.2 GHz Intel Core i5-4460 processor, and 8.00 GB RAM. Ten fold cross validation of the training data was used to provide an estimate of the accuracy of feature sets with respect to a particular feature selection algorithm.

```
Dataset File Name: object classification
 Number of Attributes: 3
 Number of Classes: 2
 Number of Samples: 7
 Evaluation Method: Random Sampling
 Percentage of unseen is: 0
 Number of experiments is: 1
 Number of training samples is: 7
 Number of unseen samples   is: 0
=====================================
  Number of rules: 5
 Average Number of conditions: 1.2
 Rules:
 If color = green => yes
 If shape = sphere => yes
 If size = medium => yes
 If shape = wedge => no
 If size = large and color  = red => no
  =====================================
 Average Number of rules: 5
 Average Number of conditions: 1.2
  Total time Consumed is: 00:00:00.0649561
```

Fig. 1 Rules resulted from applying ILA on Object Classification Training Set

```
Dataset File Name: Object Classification Training set
              without size and color attributes
 Number of Attributes: 1
 Number of Classes: 2
 Number of Samples: 3
 Evaluation Method       :Random Sampling
 Percentage of unseen is       : 0
 Number of experiments is      : 1
 Number of training samples is : 3
 Number of unseen samples   is : 7
============================================
=====
Number of rules: 3
 Average Number of conditions: 1
 Rules:
 If shape = brick => yes
 If shape = sphere => yes
 If shape = wedge => no
============================================
=====
 Average Number of rules: 3
 Average Number of conditions: 1
 Average Accuracy: 71.4285714285714%
 Total time Consumed is: 00:00:00.0028892
```

Fig. 2 The final results of applying the *Exclude* Algorithm on Classification Training set.

Table VI. Object Classification Training Set after excluding Size

Example no.	Color	Shape	Decision
1	blue	brick	yes
2	***red***	***wedge***	***no***
3	red	sphere	yes
4	***red***	***wedge***	***no***
5	green	pillar	yes
6	red	pillar	no
7	green	sphere	yes

Table VII. Object Classification Training Set after excluding Size and manipulating duplicates

Example no.	Color	Shape	Decision
1	blue	brick	yes
2	*red*	*wedge*	*no*
3	red	sphere	yes
5	green	pillar	yes
6	red	pillar	no
7	green	sphere	yes

Table VIII. The results of the first iteration on the original Object Classification Training set.

attribute	**Duplicates (C_d)**	**Contradicts (C_c)**	***C_cC_dRatio***	***Notes***
Size	1	0	100	
Color	0	2	∞	Ignored: 0 duplicates
Shape	2	2	0	

Table IX. The results of the second iteration on Object Classification Training set without Size attribute.

attribute	**Duplicates (C_d)**	**Contradicts (C_c)**	***C_cC_dRatio***	***Notes***
Color	1	2	0	
Shape	2	2	0	Ignored: Missed out class value

5.1 Evaluating the internal behavior of Exclude

Exclude algorithm has been tested on several datasets that are ranged from small to large, namely; object classification, monk1, and cars. These data sets and all datasets used in all experiments are described in Table X which are obtained from the University of California Irvine Repository of Machine Learning Databases and Domain Theories via anonymous ftp to *charlotte.ics.uci.edu : pub/machine-learning-databases*. A well-known inductive learning algorithm called ILA [16] is used in the experiments to generate rules and computes the classification accuracy of these rules. The original dataset that contains all the attributes will be used as the unseen test dataset to compute the accuracy of the resulted classifiers after excluding attributes.

Let us start with object classification dataset. This dataset has been used in the illustrative example in section 4 in order to show the steps of the algorithm how it works, and is used here to complete the vision. Table XI shows the results of applying *Exclude* on it.

As noted from Table XI, the attribute size is strongly irrelevant; its deletion reduced the data set from 7 examples to 6, and number of attributes from 3 to 2. Even that number of rules remains 5 and average conditions remains 1.2, the execution time of the resulted rules reduced from 0.0649 to 0.0428 seconds and their accuracy remains 100% on the original dataset. If we continue to iteration 2, in which the attribute color is excluded, number of rules reduced to 3, the execution time to 0.0026, average number of rules to 1, and their accuracy to 71.429%. This attribute is weakly irrelevant. If one is interested in accuracy, he can stop at iteration 1 by comparing *C_cC_dRatio* with a predefined value equal to 100%.

Table X. Description of the Domains

Domain / Characteristic	Object Classification	Monk1	Cars	Balance	Vote
Number of attributes	3+1	6+1	6+1	4+1	16+1
Number of examples	7	124	1728	625	300
Average Values per attribute	3	2.83	3.5	5	2
Number of Class Values	2	2	4	3	2
Distribution of Examples Among Class Values	57.14% are yes 42.86% are no	50% are 0 50% are 1	22.22% are acc 4.00% are good 70.02 are unacc 3.76% are vgood	46.08% are L 07.84% are B 46.08% are R	61.33% are democrat 38.67% are republican

For the cars dataset, it is noted from the results obtained as depicted in Table XII that no attribute is strongly irrelevant based on accuracy, but there are two weakly irrelevant attributes; doors and lug_boot. The attribute doors is the highest possible irrelevant attribute with accuracy 90.22% but with enhancements on number of rules that has been reduced from 541 to 77, the execution time from 2.443 to 0.186 and average number of conditions from 5.88 to 4.334. The attribute lug_boot in turn is the next highest possible irrelevant attribute to exclude after doors with enhancements in the above factors but with accuracy 82.18%. No attribute after this can be excluded because miss of class values happens. Actually again the user can stop at the desired accuracy level he wants by entering the required predefined value. It is worth to mention the fact that the attribute lug_boot is irrelevant after excluding doors. This means that lug_boot may be relevant if one tried to exclude it from the original dataset directly. Actually lug_boot if excluded directly (before doors) from the original dataset causes missed class value and consequently cannot be excluded.

Table XI. The results of applying *Exclude* on object classification dataset

	Original dataset	Iteration 1	Iteration 2
# examples	7	6	3
# attributes	3	2	1
# rules	5	5	3
Attributes excluded	0	size	color
C_cC_dRatio	-	100	0
Execution time (s)	0.0649	0.0428	0.0026
Average no. of conditions	1.2	1.2	1
% Accuracy on original dataset	100%	100%	71.429%

When we come to monk1 dataset, interested results obtained as shown in Table XIII. It is noted from the results that 3 attributes are strongly irrelevant and can be excluded with keeping the accuracy of the generated rules without these attributes at 100% on the original dataset with enhancements on all other factors.

Table XII. The results of applying *Exclude* on cars dataset

	Original dataset	Iteration 1	Iteration 2	Iteration 3
# examples	1728	388	118	Missed class
# attributes	6	5	4	
# rules	541	77	29	
Attributes excluded	--	doors	lug_boot	
C_cC_dRatio	--	92.5	76.1	
Execution time (s)	2.443	0.186	0.024	
Average no. of conditions	5.88	4.334	3.310	
% Accuracy on original dataset	100%	90.22%	82.18%	

Table XIII. The results of applying *Exclude* on monk1 dataset

	Original dataset	Iteration 1	Iteration 2	Iteration 3	Iteration 4
# examples	124	105	60	35	Missed class
# attributes	6	5	4	3	
# rules	32	31	26	25	
Attributes excluded	0	Attr3	Attr4	Attr6	
C_cC_dRatio	-	100	100	100	
Execution time (s)	1.717	1.140	0.164	0.100	
Average no. of conditions	3.281	3.29	3.3	2.88	
% Accuracy on original dataset	100%	100%	100%	100%	

It is worthy to mention that there is a kind of harmony between the suggested heuristic function used in the model and the actual classification accuracy obtained after applying the resulted rules, after excluding the irrelevant attributes, on the original dataset. This can easily be noted in Tables 11 to 13 when comparing C_cC_dRatio values with % Accuracy on original dataset. This means that the suggested heuristic function is strong and results in accurate results. Actually, C_cC_dRatio is used to estimate the classification accuracy that will be obtained if the corresponding attribute is excluded, while the "% Accuracy on original dataset" is the actual classification accuracy of applying the resulted rules on the examples in the original dataset.

5.2 Running Exclude with different induction Algorithms

Exclude, as discussed earlier, is a stand-alone algorithm that must be run as a preprocessing step prior to the usage of the induction algorithm. So we claim that it can be used with any induction algorithm. In this section, we used two well-known induction algorithms for this purpose; i.e. ID3 [9], and AQ [17] in addition to ILA [16].

Three datasets are used in these experiments, namely Balance, tic-tac-toe, and monk1. The description of these datasets is depicted in Table X.

In order to check the effect of *Exclude*, the three induction algorithms used in the experiments are run on the original datasets first and then on the same datasets after applying *Exclude* on them. Table XIV shows the result of applying the three algorithms on the three original datasets. It consists of three columns, the first column shows the results of applying ILA, ID3, and AQ on Balance dataset, while the second column is for Vote dataset, and the third is for Monk1 dataset. Three pieces of information are needed here for each application of the induction algorithm on the datasets; namely: number of rules since it is known that as less number of rules that can infer more examples are preferable, average number of condition on the left hand side of the resulted rules which indicates the simplicity of the rules, and the execution time spent by the algorithm to produce the rules. It is worth to say that the comparison between the three algorithms is not the aim of this paper, since a thorough comparison among them was done in [16].

Table XIV. The results of applying ID3, AQ, and ILA on the original datasets: monk1, Vote, and Balance.

	Monk1			**Vote**			**Balance**		
	ILA	**ID3**	**AQ**	**ILA**	**ID3**	**AQ**	**ILA**	**ID3**	**AQ**
Number of Rules	32	54	49	42	68	53	303	401	312
Average number of conditions	3.28	4.62	3.39	3.45	3.75	3.49	3.41	3.85	3.53
Execution Time (s)	1.72	2.52	2.19	4.17	5.23	4.52	0.87	1.84	1.39

The results of running *Exclude* on these three datasets are shown in Table XV. It is clear from the results that *Exclude* performed well on Vote and Monk1 datasets. It reduced number of examples from 300 to 30 on Vote, i.e. 90% reduction and from 124 to 35, i.e. around 72% reduction, while it excluded 11 attributes out of 16, which means that 5 attributes remained as weakly relevant. Weakly relevant because the predefined ratio for this dataset was 97% (because the next value needed to exclude another attribute was 93% which will reduce the accuracy significantly). For Monk1 dataset, *Exclude* was perfect. It found 3 strongly relevant attributes out of 6 and reduced number of examples from 124 to 35, all this with 100% accuracy. Balance dataset does not contain irrelevant attributes, so nothing has been changed with it, i.e. the output dataset from *Exclude* is the same as the original. I chose to keep it here to show that not all datasets contain irrelevant attributes with acceptable predefined ratio (the best irrelevant attribute in this dataset is with predefined ratio less than 60%).

Table XV.The results of applying *Exclude* onmonk1, Vote, and Balance.

	Monk1	**Vote**	**Balance**
Number of Examples(original/after Exclude)	124/35	300/30	625/625
Attributes (excluded/remained)	3/3	11/5	0/4
PredefinedRatio	100%	97%	100%

Applying the same experiment above, but on the datasets resulted from *Exclude*, show that *Exclude* performs well regardless of the inductive algorithm used. As it is seen on Table XVI, number of rules has been reduced by around an average of 22% for the three algorithms on Monk1 and 65% on Vote. It is apparent also from the results that a significant reduction of the average number of conditions by around 12% for Monk1 and 30% on Vote has been obtained, and around an average of 80% reduction in execution time on Monk1 and 96% on Vote. The most important factor to measure the induction power of a classifier is its ability to classify more unseen examples. Here, the original datasets are used as unseen datasets. *Exclude* again proofs the classification power of its resulted rules. For Monk1 dataset, since *Exclude* succeeded in excluding 3 strongly irrelevant features, the resulted classifier has classified all the examples in the original dataset for all the three algorithms, and classified more than 95% of the examples for Vote dataset for the three algorithms even that the excluded attributes are weakly irrelevant. No enhancements are obtained for Balance dataset since it has no irrelevant attributes.

Table XVI. The results of applying ID3, AQ, and ILA on the datasets: Monk1, Vote, and Balance after running *Exclude*.

	Monk1			**Vote**			**Balance**		
	ILA	**ID3**	**AQ**	**ILA**	**ID3**	**AQ**	**ILA**	**ID3**	**AQ**
Number of Rules	27	41	36	13	28	18	303	401	312
Average number of conditions	2.88	4.05	2.90	2.30	2.76	2.44	3.41	3.85	3.53
Execution Time (s)	0.10	0.87	0.43	0.013	0.27	0.06	0.87	1.84	1.39
% Accuracy on original dataset	100	100	100	97.33	91.44	92.68	100	100	100

5.3 Comparing Exclude with some other Feature Selection Approaches

To complete the whole vision, *Exclude* will be compared with three known feature selection algorithms; the first one is Relief-F algorithm which is located in the same category of *Exclude* i.e. the filter approach, the second algorithm is Wrapper method, while the third one is PSORSFS algorithm which is based on rough set theory [20].

To exploit the strength aspects of *Exclude* and to fully evaluate its performance compared with other similar algorithm in the domain, two sets of experiments have been conducted. In the first set, two datasets were used: Monk1 and Vote. The results of this set of experiments are show in Table XVII. The three algorithms are compared with three factors: reduction size, which indicates the number of attributes excluded or removed by the algorithm from the original dataset, Accuracy which indicates the classification power of the resulted classifier of the reduced datasets with respect to the original dataset, while the third factor is the execution time of the algorithm needed to produce the reduced dataset.

It is clear from the obtained results that all algorithms had found three relevant attributes in Monk1 dataset, but Exclude overcomes them in accuracy and speed. For Vote dataset, Exclude excluded 11 attributes out of 16, i.e. it ended with a dataset with only 5 attributes, while PSORSFS and Wrapper resulted in a dataset with 8 and 13 attributes respectively. Even this reduction for Exclude, its resulted classifier classified correctly 93.33% out of the examples in the original dataset. It is noted that Exclude performed faster than others, this is because the fact that its resulted datasets are almost the smallest among others.

Table XVII. Comparison among PSORSFS, Wrapper, and *Exclude* algorithms on Monk1 and Vote datasets.

Algorithm	PSORSFS			Wrapper			Exclude		
dataset	**Reduction size**	**Accuracy %[*]**	**Execution Time (s)**	**Reduction size**	**Accuracy %[*]**	**Execution Time (s)**	**Reduction size**	**Accuracy %[*]**	**Execution Time (s)**
Monk1	3	93.5	2.658	3	94.05	0.254	3	100	0.100
Vote	8	95.3	5.619	3	88.95	0.429	11	93.33	0.013

[*]*Rules are generated by ILA for all methods*

In the second set of experiments, we used seven training sets. The characteristics of these training sets are summarized in Table XVIII. These training sets are automatically generated realistic data sets, using the Synthetic Classification Data Set Generator (SCDS) version 2. Using synthesized data sets is another important way to assess inductive learning algorithms.

In this set of experiments, ILA has been used also to generate all rules. Table XIX summarizes all the results obtained through this experiment. The first column of the table shows number of rules in the original sets. This piece of information is needed to compare it with number of rules in the resulted datasets.

It is noted from Table XIX that *Exclude* performs good in most cases with respect to reduction size, accuracy on the original dataset, and number of rules. It is worthy to mention that the results are domain dependent i.e. depends on the values in the dataset, but these results give an indication of the power of the algorithm. It is noticed that *Exclude* works better on datasets with many attributes and decision values, as in the case of dataset 4 that contains 19 attributes and 4 decision values, and dataset 7 that contains 25 attributes and 6 decision values.

VI. Summary and Conclusions

A new simple algorithm; called Exclude, has been suggested and developed. This algorithm is a result of an empirical study and intensive investigation on many datasets. It is based on an iterative process of excluding the most possible irrelevant attribute and keeping the exclusion process of the most irrelevant attribute with respect to the most recent excluded one and so on until one attribute remains in the data set or a predefined threshold value has been reached.

The results showed that Exclude can produce datasets with less number of attributes and examples, which in turn implies to simpler classifiers with respect to number of rules and number of conditions in their left hand side part of the produced rules.

Exclude has been tested by three powerful inductive learning algorithms ILA, ID3, and AQ. The experiments showed that it can be used regardless of the inductive learning algorithm used.

It has been also tested on many datasets and compared with some other well-known feature selection methods as Relief-F, PSORSFS, and Wrapper algorithms. It got comparable results among these methods if not better in most datasets used in the experiments.

Table XVIII. Characteristics of the seven Training Sets.

	Number Of Examples	Number Of Attributes	Average Values Per Attribute	Number Of Class Values	Distribution Of Examples Among Class Values
Data Set 1	100	3+1	20	3	1. 44.0% are X 2. 28.0% are Y 3. 28.0% are Z
Data Set 2	300	5+1	5	2	1. 50.0% are X 2. 50.0% are Y
Data Set 3	500	5+1	3	4	1. 47.8% are X 2. 29.4% are Y 3. 16.4% are Z 4. 6.4% are W
Data Set 4	500	19+1	6	4	1. 46.8% are X 2. 29.6% are Y 3. 19.0% are Z 4. 4.6% are W
Data Set 5	1000	12+1	6	3	1. 83.8% are X 2. 13.0% are Y 3. 3.2% are Z
Data Set 6	5000	15+1	10	2	1. 51.2% are X 2. 48.8% are Y
Data Set 7	10000	25+1	100	6	1. 33.7% are X 2. 25.8% are Y 3. 19.5% are Z 4. 11.1% are X 5. 7.9% are Y 6. 2.0% are Z

Table XIX. Comparison among PSORSFS, Wrapper, and *Exclude* algorithms on the seven datasets.
** Rules are generated by ILA for all methods*

Algorithm / dataset	**# rules for original dataset***	**Relief-F**			**Wrapper**			***Exclude***		
		Reduction size	**Accuracy%**	**# rules***	**Reduction size**	**Accuracy %**	**# rules***	**Reduction size**	**Accuracy %**	**# rules***
Data Set 1	17	0	100	17	1	92.63	14	***1***	***93.04***	***13***
Data Set 2	51	2	91.84	41	1	88.49	45	***2***	***92.04***	***39***
Data Set 3	64	1	83.86	57	2	86.4	53	***3***	***91.32***	***46***
Data Set 4	111	7	87.55	83	6	87.06	86	***10***	***94.26***	***74***
Data Set 5	203	5	92.17	162	5	***92.89***	167	5	92.86	***160***
Data Set 6	627	7	90.83	438	***8***	***92.89***	***427***	6	89.17	430
Data Set 7	892	12	92.6	621	11	91.86	658	***17***	***97.41***	***366***

References

[1] R. Kohavi and G. John. Wrappers for feature subset selection, Artificial Intelligence 97, 1997, pp. 273-324.

[2] H. Almuallim and T. Dietterich. Learning with many irrelevant features, in: Proceedings AAAI-91, Anaheim, CA (MIT Press, Cambrige, MA, 1991) 547-552.

[3] H. Almuallim and T. Dietterich. Learning Boolean concepts in the presence of many irrelevant features, Artificial Intelligence 69, 1994, 279-306.

[4] K. Kira and L.A. Rendell. The feature selection problem: Traditional methods and a new algorithm, in: Proceedings AAAI-92, San Jose, CA (MIT Press, Cambridge, MA, 1992) 129-134.

[5] K. Kim and L.A. Rendell. A practical approach to feature selection, in: Proceedings 9th InternationalConference on Machine Learning, Aberdeen, Scotland (Morgan Kaufmann, Los Altos, CA, 1992).

[6] I. Kononenko. Estimating attributes: analysis and extensions of Relief, in: F. Bergadano and L. DeRaedt, eds., Proceedings European Conference on Machine Learning (1994).

[7] G.H. John, R. Kohavi and K. Pfleger. Irrelevant features and the subset selection problem, in: Proceedings 11th International Conference on Machine Learning, New Brunswick, NJ (Morgan Kaufmann, SanMateo, CA, 1994) 121-129.

[8] P. Langley and S. Sage. Oblivious decision trees and abstract cases, in: Working Notes of the AAAI-94 Workshop on Case-Based Reasoning, Seattle, WA (AAAI Press, 1994) 113-117.

[9] J.R. Quinlan. Learning efficient classification procedures and their application to chess end games, in: R.S. Michalski, J.G. Carbonell and T.M. Mitchell, eds., Machine Learning: An Artificial Intelligence Approach (Morgan Kaufmann, San Mateo, CA, 1983).
[10] J.R. Quinlan. C4.5: Programs for Machine Learning (Morgan Kaufmann, San Mateo, CA, 1993).
[11] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone. Classification and Regression Trees (Wadsworth, Belmont, CA 1984).
[12] P. Clark and T. Niblett. The CN2 induction algorithm, Machine Learning 3 (1989) 261-284.
[13] R.S. Michalski. Pattern recognition as rule-guided inductive inference, IEEE Trans. Pattern Analysis Machine Intelligence. 2 (1980) 349-361.
[14] P.A. Devijver and J. Kittler. Pattern Recognition: A Statistical Approach (Prentice-Hall, Englewood Cliffs, NJ, 1982).
[15] M. R. Tolun, H. Sever, M. Uludag, and S. Abu-Soud. ILA-2: An inductive learning algorithm for knowledge discovery. Cybernetics & Systems, 1999, 30(7), 609-628.
[16] M. Tolun, and S. Abu-Soud. ILA: An Inductive Learning Algorithm for Rule Extraction, Expert Systems with Applications, 14(3), (1998) 361-370.
[17] R.S. Michalski,I. Mozetic, J.Hong, and N. Lavrac. The Multipurpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains, Proc. of the Fifth National Conference on Artificial Intelligence, Philadelphia, PA: Morgan Kaufmann, 1986, 1041-1045.
[18] Thornton, C.J. Techniques in Computational Learning-An Introduction, London: Chapman & Hall, 1992.
[19] Pham, D.T. & Aksoy, M.S. "RULES: A Simple Rule Extraction System", Expert Systems with Applications, 8(1), 1995, 59-65.
[20] X. Wang, J. Yang, W. Xia, and R. Jensen. Feature Selection based on Rough Sets and Particle Swarm Optimization, Pattern Recognition Letters, Vol. 28, Issue 4, 2007, 459-471.

Dr. Saleh M. Abu-Soud is an associate professor at the Department of Software Engineering in Princess Sumaya University for Technology (PSUT). He got his PhD in Computer Science in 1992 (METU), M. Sc. in Computer Science in 1988 (METU), and B.Sc. in Computer Science in 1985 (Yarmouk University). He was working in Jordan University in the period between 1992 and 1995, then he joined PSUT till now, in which he served as the head of the department of Computer Science in the period from 2005 to 2007. He left to work in NYIT for four years in the period from 2007 to 2011, in which he was a professor of Computer Science and the director of accreditation and quality assurance department in the period from 2007 to 2010. His research interest is in the area of Artificial Intelligence. He is the owner of ILA inductive learning algorithm. He is interested mainly in many research topics as Machine Learning, Biometric Keystroke Dynamics, and Speech Synthesis with inductive learning. He has many research papers and 2 books. He supervised dozens of master students and many PhD students, more details can be seen on (***http://scholar.google.com/citations?user=YjZiOScAAAAJ&hl=en***) and ***https://www.researchgate. net/profile/Saleh_Abu-Soud***. He is a member of many international projects.