FEATURE SELECTION CONSIDERING ATTRIBUTE INTER-DEPENDENCIES

Manuel Mejía-Lavalle, Eduardo F. Morales[1]
Instituto de Investigaciones Eléctricas, Reforma 113, 62490 Cuernavaca, Morelos, México
[1] INAOE, L.E.Erro 1, 72840 StMa. Tonantzintla, Puebla, México
mlavallle@iie.org.mx, emorales@inaoep.mx

Abstract. With the increasing size of databases, feature selection has become a relevant and challenging problem for the area of knowledge discovery in databases. An effective feature selection strategy can significantly reduce the data mining processing time, improve the predicted accuracy, and help to understand the induced models, as they tend to be smaller and make more sense to the user. Many feature selection algorithms assumed that the attributes are independent between each other given the class, which can produce models with redundant attributes and/or exclude sets of attributes that are relevant when considered together. In this paper, an effective best first search algorithm, called buBF, for feature selection is described. buBF uses a novel heuristic function based on *n-way* entropy to capture inter-dependencies among variables. It is shown that buBF produces more accurate models than other state-of-the-art feature selection algorithms when compared on several synthetic and real datasets.

Keywords: Data mining, Feature selection, *n-way* entropy.

1 Introduction

Data mining is mainly applied to large amounts of stored data to look for the implicit knowledge hidden within this information. To take advantage of the enormous amount of information currently available in many databases, algorithms and tools specialized in the automatic discovery of hidden knowledge within this information have been developed. This process of non-trivial extraction of relevant information that is implicit in the data is known as Knowledge Discovery in Databases (KDD), in which the data mining phase plays a central role in this process.

It has been noted, however, that when very large databases are going to get mined, the mining algorithms get very slow, requiring too much time to process the information. One way to approach this problem is to reduce the amount of data before applying the mining process. In particular, the pre-processing method of feature selection, applied to the data before mining, has been shown to be promising because it can eliminate the irrelevant or redundant attributes that cause the mining tools to become inefficient and ineffective. At the same time, it can preserve-increase the classification quality of the mining algorithm (accuracy) [1].

Although there are many feature selection algorithms reported in the specialized literature, none of them are perfect: some of them are effective, but very costly in computational time (e.g., wrappers methods), and others are fast, but less effective in the feature selection task (e.g., filter methods).

Specifically, wrapper methods, although effective in eliminating irrelevant and redundant attributes, are very slow because they apply the mining algorithm many times, changing the number of attributes each time of execution as they follow some search and stop criteria [2]. Filter methods are more efficient; they use some form of *correlation measure* between individual attributes and the class [3]; however, because they measure the relevance of each isolated attribute, they cannot detect if redundant attributes exist, or if a combination of two (or more) attributes, apparently irrelevant when analyzed independently, are indeed relevant [4].

In this article we propose a feature selection method that tries to solve these problems in a supervised learning context. Specifically, we use a heuristic search alternative, inspired by the Branch & Bound algorithm, which reduces considerably the search space, thus reducing the processing time. Additionally, we propose a novel evaluation criterion based on an *n-way* entropy measure that, at the same time, selects the relevant attributes and discovers the important inter-dependences among variables of the problem.

To cover these topics, the article is organized as follows: Section 2 surveys related work; Section 3 introduces our feature selection method; Section 4 details the experiments; conclusions and future research directions are offered in Section 5.

2 Related Work

The emergence of Very Large Databases (VLDB) leads to new challenges that the mining algorithms of the 1990´s are incapable to attack efficiently. According to [5], from the point of view of the mining algorithms, the main lines to deal with VLDB (scaling up algorithms) are: a) to use relational representations instead of a single table; b) to design fast algorithms, optimizing searches, reducing complexity, finding approximate solutions, or using parallelism; and c) to divide the data based on the variables involved or the number of examples. In particular, some of these new approaches in turn give origin to Data Reduction that tries to eliminate variables, attributes or instances that do not contribute information to the KDD process. These methods are generally applied before the actual mining is performed.

In fact, the specialized literature mentions the *curse of dimensionality*, referring to the fact that the processing time of many induction methods grows dramatically (sometimes exponentially) with the number of attributes. Searching for improvements on VLDB processing power (necessary with tens of attributes), two main groups of methods have appeared: wrappers and filters [5]. We focus our research on filters methods with, near to, optimum solutions because of their relatively low computational cost.

Narendra [6] and others [7], [8], [9] have proposed a filter method for optimal feature selection. In general, they use the Branch & Bound algorithm, starting the search with all the D features and then applying a backward elimination feature strategy, until they obtain d optimal features ($d < D$). Additionally, they use a monotonic subset feature evaluation criterion: i.e., when augmenting (subtracting) one feature to the feature subset, the criterion value function always increases (decreases). The monotonicity property allows us to prune unnecessary sub-trees (e.g., sub-trees that do not improve the solution because they have values less than the bound obtained for another sub-tree). These approaches have demonstrated to be efficient; however, they have several drawbacks, because they need:

- An a priori definition of the number of features d (equal to the maximum tree deep level to consider); this is a problem because, in most cases, the number of relevant attributes is previously unknown,
- To start evaluating all the features (top-down strategy); this strategy represents high computational cost at the beginning of the subset feature search process,
- To use a monotonic subset evaluation criterion; although a monotonic criterion permits safe sub-trees cut offs, it assumes that the features are independent between each other, given the class attribute.

Trying to tackle these problems, in this paper we propose a bottom-up Best First method that is described in the next Section.

3 Bottom-Up Best First

The proposed method basically has two components: a) the evaluation function of each feature subset (in a supervised learning context), and b) the search strategy.

3.1 Evaluation criterion

With respect to the feature subset evaluation criterion, we proposed a non-monotonic function that, essentially, is calculated in a similar way to the Shannon entropy, only that instead of considering the entropy of one single feature, or attribute, against the class attribute (*2-way* entropy, or traditional entropy), it is calculated considering the entropy of two (or more attributes) against the class (*n-way* entropy). With this approach, we sought to capture the inter-dependences among attributes.

Formally, the traditional entropy H of a variable X after observing values of another variable Y is defined as:

$$H(X \mid Y) = -\Sigma_j P(y_j)\ \Sigma_i P(x_i \mid y_j)\ log_2 (P(x_i \mid y_j)), \tag{1}$$

where $P(x_i \mid y_j)$ is the posterior probabilities of X given the values of Y. We obtain the *n-way* entropy Hn with the same equation but, instead of using the count of only one attribute, we count the number of times that a particular combination of attribute values appears, against the class value, taking into account all the instances of the dataset. In this form, if the *n-way* entropy Hn decreases, using a particular feature subset, means that we have additional information about the class attribute. For instance, if U and V are different attribute subsets, C is the class attribute, and if $Hn(U|C) > Hn(V|C)$, then we conclude that subset V predicts better than subset U.

The idea of calculating in this manner the *n-way* entropy is inspired by the work of Jakulin and Bratko [10]. Although they calculate this in a more costly way using the concept of Interaction Gain I. For instance, they obtain the 3-way interactions using:

$$I(X; Y; C) = H(X|C) + H(Y|C) - H(X,Y|C) - \{ H(X) + H(Y) - H(X,Y) \}, \tag{2}$$

so, we experiment with the *n-way* entropy variant Hn because of its simplicity and its relative low computational cost.

Nevertheless, a defect or problem with the *n-way* entropy Hn is that it decreases quickly when the number of the combined attribute values grows, resulting as ”false” low entropy. In an extreme case, it is possible that we can count as many different combined attribute values as the total number of dataset instances. If we count as many combined attribute values as instances, then the entropy will be zero (perfect). But this does not necessarily reflect, in an effective way, how that combination of attributes is relevant. The specialized literature has already reported how the entropy tends to prefer those attributes that have more different values, then, an attribute randomly generated could be considered better than another attribute observed from the real system.

Although there are some proposals to mitigate the problem (e.g., gain ratio or symmetrical uncertainty), they usually add an extra computational cost; because of that, we directly apply a reward to the *n-way* entropy considering the number of values that a specific attribute (or attributes) can take. Our proposed evaluation criterion, or metric, is defined as:

$$nwM = \lambda\ (Hn) + (1 - \lambda)(tot.\ combined\ attribute\ values\ /\ tot.\ instances) \tag{3}$$

With this metric, a balance between the *n-way* entropy Hn and the combined attribute values is sought, obtaining a metric, now called *nwM*, to detect relevant and inter-dependant features. The λ parameter can take values between zero and one and it is defined by the user according to how much weight he desires to give to each term. We empirically test the proposed metric, and obtain very promising results (see Section 4).

3.2 Search strategy

With respect to the search strategy, we propose to explore a search tree with forward feature selection or bottom-up schema.

The idea consists in using a best first search strategy: always expanding (aggregates a new feature) to the node (attribute subset) whose metric is the best of the brother nodes (node with the smaller *nwM*) and better than the parent node, stopping the search when none of the expanded nodes are better than the parent node. In this case, following the best first search strategy, the search continues selecting the best non-expanding node, according to the metric, and expanding until none of the children nodes are better than the parent node, and so on.

Thus, the proposed search schema explores the most promising attribute combinations according to the non-monotonic metric, generating several possibly good solutions. At the same time, it carries out sub-tree pruning, when the *nwM* metric has indicated, heuristically, that continuing to explore some of those sub-trees, maybe will not improve the evaluation criterion. The search process stops when the memory has been saturated, or when all the nodes have been expanded. The modified algorithm, called now bottom-up Best First (buBF), is shown in Fig. 1 (|| . || is the set size).

```
Given a dataset with D features and N instances, and $\lambda \in [0,1)$,

1. obtain nwM (2-way entropy) for each feature in the dataset;
2. while (available memory) or (unexplored nodes) do begin
3.    select for expansion the feature subset F with the best nwM and
                                          better than his parent node;
4.    for  I := 1 to ( D – || F || ) do begin
5.      obtain nwM ( $F \cup I \mid I \notin F$ );
6.    end;
7. end;
8. show feature subset with the best nwM;
```

Fig. 1. buBF algorithm.

The proposed search seems like a Branch & Bound strategy, in the sense that it prunes sub-trees that maybe will not conduct to better solutions, according to the evaluation criterion. Nevertheless, it is not exactly equal to the feature selection Branch & Bound schema reported in the specialized literature.

The differences basically consist of:

• Instead of removing attributes and evaluating the resulting feature subset (backward elimination), our method adds attributes and evaluates (forward selection). Using forward selection we will be able to process datasets with more features.

• Instead of using a monotonic evaluation criterion, a non-monotonic criterion is employed. Although sub-tree pruning is not safe using a non-monotonic criterion, our heuristic measure captures attributes inter-dependencies.

• Instead of having to define an a priori tree depth, in our case the tree depth search is variable, and depends on the evaluation criterion: this criterion indicates stopping the depth search when none children node is better than the parent node.

• In our case, adding nodes (attributes) is sought to determine not only the relevant attributes, but also their inter-dependences, being that other methods reported in the literature assumes attribute independence [8].

4 Experiments

We conducted several experiments with synthetic and real datasets to empirically evaluate if buBF can do better in selecting features than other well-known feature selection algorithms, in terms of learning accuracy and processing time. We choose synthetic datasets in our experiments because the relevant features of these datasets are known beforehand.

4.1 Experimentation details

The experimentation objective is to observe the buBF behavior related to classification quality and response time using 10 synthetic datasets, each of them with different levels of complexity. To obtain the

10 datasets we use the functions described in [11]. Each of the datasets has nine attributes (1.salary, 2.commission, 3.age, 4.elevel, 5.car, 6.zipcode, 7.hvalue, 8.hyears, and 9.loan) plus the class attribute (with class label Group "A" or "B"); each dataset has 10,000 instances. The values of the features of each instance were generated randomly according to the distributions described in [11]. For each instance, a class label was determined according to the rules that define the functions. For example, function 9 uses four attributes and classifies an instance following the statement and rule shown in Fig. 2.

$$\text{disposable} := (0.67 * (\mathit{salary} + \mathit{commission}) - 5000 * \mathit{elevel} - 0.2 * \mathit{loan} - 10000)$$

```
IF ( disposable > 0 ) THEN class label := Group "A"
    ELSE class label := Group "B"
```

Fig. 2. A function example.

We experiment too with the corrAL (and corrAL-47: see [12] for details) synthetic dataset, that has four relevant attributes (A0, A1, B0, B1), one irrelevant (I) and one redundant (R); the class attribute is defined by the function $Y = (A0 \wedge A1) \vee (B0 \wedge B1)$. Finally, we test our proposed method with a real database with 24 attributes and 2,770 instances; this database contains information of Mexican electric billing costumers, where we expect to obtain patterns of behavior of illicit customers.

In order to compare the results obtained with buBF, we use Weka´s [13] implementation of ReliefF, OneR and ChiSquared feature selection algorithms. These implementations were run using Weka´s default values, except for ReliefF, where we define to 2 the number of neighborhood, for a more efficient response time. Additionally, we experiment with 7 Elvira´s [14] filter-ranking methods: *Mutual Information, Euclidean, Matusita, Kullback-Leibler-1 and 2, Shannon and Bhattacharyya*.

To select the best ranking attributes, we use a threshold defined by the largest gap between two consecutive ranked attributes (e.g., a gap greater than the average gap among all the gaps [12]). In the case of buBF, we set λ to 0.85 for all the experiments. All the experiments were executed in a personal computer with a Pentium 4 processor, 1.5 GHz, and 250 Mbytes in RAM. In the following Section the obtained results are shown.

4.2 Experimental results

Using 10 synthetic datasets, the features selected by each method are shown in Table 1, where "Oracle" represents a perfect feature selection method (it selects exactly the same features that each function uses to generate the class label). We can observe that, in some cases, the methods almost select the same features, but there are other functions in which the methods disagree. For function 8, only OneR cannot determine any feature subset, because ranks all attributes equally.

Next, we used the selected features for each method as input to the decision tree induction algorithm J4.8 included in the Weka tool. J4.8 is the last version of C4.5, which is one of the best-known induction algorithms used in data mining. We use 10-fold cross validation in order to obtain the average test accuracy for each feature subset. The results are shown in Table 2 (in this case, using all the attributes results in the same accuracy than using only the oracle attributes).

To summarize the obtained results in Table 2, we count the times when buBF wins, losses or ties against the other methods. This information is reported in Table 3, where it can be observed that buBF has a good performance, because there was only loss one time versus ReliefF, and one time versus ChiSquared, but it still maintained good accuracy.

Table 1. Features selected by different methods (10 synthetic datasets).

Function number	Method											
	Oracle	Mut.Infor	Euclidean	Matusita	Kullback Leibler-1	Kullback Leibler-2	Shannon	Bhattach	ReliefF	OneR	ChiSquar	buBF
1	3	3	3	3	3	9-7-2-8	9-1	3	3	3	3	3
2	1-3	1	2-1	1	1-2	1	9-3-7-1	1	3-1	1	1-2	3-1
3	3-4	4-3	4	4-3	4-3	4-3	3-9-1	4-3	4-3	4-3	4-3	3-4
4	1-3-4	1	2-1	1	1	1	1-9	1	1-4-2	1-2	1-2	4-3-1
5	1-3-9	9-1	9-4	9	9	9-1	1-3	9	9-3-1	9	9	5-2-3-9
6	1-2-3	1-3-2	2	1-3	1-3	1	3	1-3-2	3-1-2	3-1-2	1-3-2	1-2-3
7	1-2-9	9	2-9	9	9-1-2	9	1-9	9-1	9-1-2	9	9-1-2	9-1-2
8	1-2-4	2-1	2-4-1	2-1	2-1-4	2-1	9-3	2-1	1-2-4	-	1-2-4	4-2-1
9	1-2-4-9	9	2-4-9	9-1	9	9	9	9-1	9-1-2	9	9-1-2-4-3	2-1-9
10	1-2-4-7-8-9	4	4	4	4	4	9-1-3	4	8	4	4-8-7-6	6-8-4

Table 2. J4.8´s accuracies (%) using the features selected by each method (10 synthetic datasets).

Function number	Method											
	Oracle/ All	buBF	ReliefF	ChiSquar	Bhattach	Mut.Infor	Kullback Leibler-1	Matusita	OneR	Kullback Leibler-2	Euclidean	Shannon
1	100	100	100	100	100	100	100	100	100	67	100	67
2	100	100	100	73	73	73	73	73	73	73	73	100
3	100	100	100	100	100	100	100	100	100	100	68	59
4	100	100	90	84	84	84	84	84	84	84	84	84
5	100	91	100	74	74	82	74	74	74	82	74	60
6	99	99	99	99	99	99	87	87	99	68	64	69
7	98	98	98	98	94	86	98	86	86	86	88	94
8	100	100	100	100	99	99	100	99	-	99	100	98
9	97	94	94	97	92	85	85	92	85	85	88	85
10	99	99	80	99	97	97	99	97	98	97	97	80
Avg.	**99.3**	**98.1**	**96.1**	**92.4**	**91.2**	**90.5**	**89.8**	**89.2**	**84.9**	**84.1**	**83.6**	**79.6**

Table 3. buBF accuracy results summary vs. other methods (10 synthetic datasets).

buBF vs.	Method											
	Oracle/ All	OneR	ReliefF	ChiSquar	Bhattach	Mut.Infor	Kullback Leibler-1	Matusita	Shannon	Kullback Leibler-2	Euclidean	Average
Win	0	7	2	3	7	7	5	8	9	9	8	**5.9**
Loss	2	0	1	1	0	0	0	0	0	0	0	**0.4**
Tie	8	3	7	6	3	3	5	2	1	1	2	**3.7**

With respect to the processing time, this is shown in Table 4. We observe that, although buBF is computationally more expensive than OneR and ChiSquared, these algorithms cannot detect some attribute inter-dependencies; on the other hand, buBF is faster than ReliefF, but with similar, or better, feature selection performance.

To have a better idea of the buBF performance, we can compare the results presented previously against the results produced by an exhaustive wrapper approach. In this case, we can calculate that, if the average time required to obtain a tree using J4.8 is 1.1 seconds, and if we multiply this by all the possible attribute combinations, then we will obtain that 12.5 days, theoretically, would be required to conclude such a process.

Table 4. Averaged processing time for each method (10 synthetic datasets).

Exhaustive wrapper	**ReliefF**	**OneR**	**ChiSquared and Elvira**	**buBF**
1,085,049 secs. (12.5 days)	573 secs. (9.55 mins.)	8 secs.	1 sec.	71 secs. (1.18 mins.)

In order to observe how the selected features (Table 1) respond with another classifier, we use these features as input to the Naïve Bayes Classifier (NBC) included in the Weka tool. Results are shown in Table 5. Again, buBF obtains satisfactory accuracy results.

Table 5. NBC´s averaged accuracies (%) for 10-fold-cross validation using the features selected by each method (10 synthetic datasets).

Function number	**Method**											
	Oracle	**buBF**	**Matusita**	**Kullback Leibler-1**	**Bhattach**	**Mut.Infor**	**ChiSquar**	**ReliefF**	**Euclidean**	**Kullback Leibler-2**	**OneR**	**Shannon**
1	89	89	89	89	89	89	89	89	89	67	89	67
2	69	69	69	64	69	69	64	69	64	69	69	68
3	65	65	65	65	65	65	65	65	66	65	65	58
4	76	76	76	76	76	76	70	69	70	76	70	76
5	68	68	68	68	68	68	68	68	68	68	68	60
6	71	71	72	72	71	71	71	71	59	60	71	58
7	89	89	86	89	88	86	89	89	86	86	86	88
8	99	99	98	99	98	98	99	99	99	98	50	98
9	89	88	88	85	88	85	88	88	86	85	85	85
10	98	98	98	98	98	98	97	80	98	98	98	80
Avg.	**81.3**	**81.2**	**81**	**81**	**81**	**80.5**	**80**	**78.7**	**78.5**	**77.2**	**75.1**	**73.8**

When we test with the corrAL and corrAL-47 datasets [12], our method was the only that can remove the redundant attribute (Table 6; results for FCBF method was taken from [12]). This suggests that our method, although requires more processing time, is a good approach to capture inter-dependencies among attributes. On the other hand, buBF processing time is competitive when we try to use wrapper feature selection methods.

Finally, testing over the electric billing database, buBF obtains the best accuracy ties with Kullback-Leibler-2, but with less attributes (Table 7).

We point out that we do not carry out comparisons against Branch & Bound methods because, in general, these require a previous definition of the number of attributes to select, which is not necessary with buBF.

Table 6. Features selected by different methods (corrAL and corrAL-47 datasets).

Method	**Features selected**	
	corrAL	**corrAL-47**
buBF	B1, B0, A1, A0	A0, A1, B0, B1
ReliefF	R, A0, A1, B0, B1	R,$B1_1$,A0,$A0_0$,B1,$B1_0$,B0,$B0_0$,$B0_2$,A1,$A1_0$
$FCBF_{(log)}$	R, A0	R, A0, A1, B0, B1
$FCBF_{(0)}$	R, A0, A1, B0, B1	R, A0, A1, B0, B1

Table 7. J4.8´s accuracies (%) for 10-fold-cross validation using the features selected by each method (electric billing database).

Method	**Total features selected**	**Accuracy (%)**	**Pre-processing time**
buBF	5	97.50	1.5 mins.
Kullback-Leibler 2	9	97.50	6 secs.
All attributes	24	97.25	0
ChiSquared	20	97.18	9 secs.
OneR	9	95.95	41 secs.
ReliefF	4	93.89	14.3 mins.
Euclidean distance	4	93.89	5 secs.
Shannon entropy	18	93.71	4 secs.
Bhattacharyya	3	90.21	6 secs.
Matusita distance	3	90.21	5 secs.
Kullback-Leibler 1	4	90.10	6 secs.
Mutual Information	4	90.10	4 secs.

5 Conclusions and Future Work

We have presented a new algorithm for feature selection that tries to overcome some drawbacks found in Branch & Bound feature selection algorithms. Thus, the proposed method follows a forward attribute selection (instead of backward, like other methods do) finding reductions in processing time, because it is less costly to obtain the evaluation criterion for few attributes than for all the features.

Additionally, we propose a new subset evaluation criterion, that considers a balanced *n-way* entropy with respect to the combined attribute values; this metric is not very expensive and, due to the fact that is non-monotonic, heuristically allows pruning the search tree, with additional processing time savings. Furthermore, the *n-way* entropy considers the inter-dependences among features, obtaining not only isolated relevant features, and doing unnecessary a previously definition of the tree depth.

From the experimental results, the proposed method buBF represents a promising alternative, compared to other methods, because of its acceptable processing time and good performance in the feature selection task.

Some future research issues arise with respect to buBF improvement. For example: further experimentations with more real databases; comparing against other similar methods (e.g., Liu´s ABB [15]); using another metric variations to eliminate the λ parameter (e.g., DKM) and using more efficient search methods (e.g. multi-restart hill-climbing); improving the tree pruning strategy and test the method with data sets with more instances and attributes.

References

1. Guyon, I., Elisseeff, A., An introduction to variable and feature selection, Journal of machine learning research, 3, 2003, pp. 1157-1182.
2. Kohavi, R., John, G., Wrappers for feature subset selection, Artificial Intelligence Journal, Special issue on relevance, 1997, pp. 273-324.
3. Piramuthu, S., Evaluating feature selection methods for learning in data mining applications, Proc. 31st annual Hawaii Int. conf. on system sciences, 1998, pp. 294-301.
4. Molina, L., Belanche, L., Nebot, A., Feature selection algorithms, a survey and experimental eval, IEEE Int.conf.data mining, Maebashi City Japan, 2002, pp. 306-313.
5. Mitra, S., et.al., Data mining in soft computing framework: a survey, IEEE Trans. on neural networks, vol. 13, no. 1, January, 2002, pp. 3-14.
6. Narendra, P., Fukunaga, K., A branch and bound algorithm feature subset selection, IEEE Trans. computers, vol. 26, no. 9, sept 1977, pp. 917-922.
7. Yu, B., Yuan, B., A more efficient branch and bound algorithm for feature selection, Pattern Recognition, vol. 26, 1993, pp. 883-889.
8. Frank, A., Geiger, D., Yakhini, Z., A distance-B&B feature selection algorithm, Procc. Uncertainty in artificial intelligence, México, august. 2003, pp. 241-248.
9. Somol, P., Pudil, P., Kittler, J., Fast Branch & bound algorithms for optimal feature selection, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 26, no. 7, july 2004, pp. 900-912.
10. Jakulin, A., Bratko, I., Testing the significance of attribute interactions, Procc. Int. conf. on machine learning, Canada 2004, pp. 409-416.
11. Agrawal, R., Imielinski, T, Swami, A., Database mining: a performance perspective, IEEE Trans. Knowledge data engrg. Vol. 5, no. 6, 1993, pp. 914-925.
12. Yu, L., Liu, H., Efficient feature selection via analysis of relevance and redundancy, Journal of Machine Learning Research 5, 2004, pp. 1205-1224.
13. www. cs.waikato.ac.nz/ml/weka, 2004.
14. www. ia.uned.es/~elvira/ , 2004.
15. Liu, H. Motoda, and M. Dash. A monotonic measure for optimal feature selection. In Proceedings of European Conference on Machine Learning,, 1998, pp. 101-106.