Perceptron for Feature Selection

Manuel Mejía-Lavalle[1], and Enrique Sucar[2]

[1]Instituto de Investigaciones Eléctricas, Reforma 113, 62490 Cuernavaca, Morelos, México
mlavalle@iie.org.mx
[2]INAOE L.E.Erro 1, 72840 StMa. Tonantzintla, Puebla, México
esucar@inaoep.mx

(Paper received on June 21, 2007, accepted on September 1, 2007)

Abstract. An exploratory and effective research around the possible application of the Perceptron paradigm as a method for feature selection is carried out. The main idea is training a Perceptron and then using the learned inter-connection weights as metric of which database's attributes are the most discriminative for the class. We hypothesized that an inter-connection weight close to zero indicates that the associated attribute to this weight can be eliminated because it does not contribute with relevant information in the construction of the class separator hyper-plane of the Perceptron. The experiments that were realized, with 8 real and 11 synthetic databases, show that the proposed algorithm is a good trade-off among feature reduction, accuracy and processing time.

1 Introduction

Feature selection has become a relevant and challenging problem for the area of Knowledge Discovery in databases (KDD). An effective feature selection strategy [1] 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.

Although there is many feature selection algorithms reported in the specialized literature, none of them are perfect: some of them are fast, but not effective in the feature selection task (e.g., *filter* methods), and others are effective, but very costly in computational time (e.g., *wrapper* methods).

Specifically, filter methods are more efficient; they use some form of *correlation* measure between individual attributes and the class [2][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 together [4]. On the other hand wrapper methods, although effective in eliminating irrelevant and redundant attributes and detecting inter-dependencies, 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 [5].

Here, we propose and experiment with a feature selection strategy, in which the trained Perceptron inter-connection weights are utilized like a metric, indicator or measure of attribute importance or relevance. The basic idea is to eliminate the

Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 115-125

attributes whose associated weights are close to zero. Similar idea is used in more complex, time-consuming and memory demanding methods, like the Principal Component Analysis (PCA) technique and the Support Vector Machine (SVM) variants for Feature Selection (SVM-FS) [1][7].

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

2 Feature Selection Perceptron Strategy

The method to explore is the classic Rosenblatt's Perceptron, as strategy for relevant attribute selection. We propose to use a "soft" or "relaxed" Perceptron (similar to [6]), in the sense that it can accept some percentage of misclassified instances; in this case the training stopping criterion occurs when no accuracy improvement is obtained. To measure the improvement/ no-improvement we use the generalization accuracy (Acc) and the Balanced Error Rate (BER) [7]. The Perceptron variation that we used has: a) as many inputs-inter-connection weights as attributes the dataset contain, b) only one neuron with a step activation function, and c) only one output. To obtain the Perceptron output *S*, we use the equation:

$$S = U\{\Sigma_i W_i E_{ij}\} \tag{1}$$

where W_i are the i inter-connection weights; E_{ij} is the input vector (with i elements) that form an instance j; and U is a step function that outputs *1* (one) if $\Sigma_i W_i E_{ij} > \theta$ and *0* (zero) otherwise. θ is the Perceptron's threshold.

To train the Perceptron we apply the equations:

$$W_i(t+1) = W_i(t) + \{\alpha(T-S)E_{ij}\} \tag{2}$$

$$\theta(t+1) = \theta(t) + \{-\alpha(T-S)E_{ij}\} \tag{3}$$

where T is the desired output (or target) and α is the learning rate: this user parameter can take values between 0 and 1.

Although there exist more sophisticated procedures in the area of neural network pruning [8], we choose this idea because of its simplicity (and therefore, efficiency) and direct application to feature selection, because of the direct relation between each feature and its Perceptron inter-connection weight. To execute the overall feature selection process we apply the procedure shown in Fig. 1.

3 Experiments

With the Perceptron for Feature Selection (PFS) we expect:

a) To use less amount of memory, because a Perceptron only requires to store as many inter-connection weights as "n" attributes the database has, as opposed to PCA that builds an "n^2" matrix,
b) To drop the processing time because, as opposed to SVM-FS [7] that involves solving a quadratic optimization problem, the Perceptron converges fast to an approximate solution,
c) To avoid to carry out a combinatorial explosive search or exhaustive exploration; that is to say, without having to evaluate multiple attribute subset combinations, as the wrapper methods do (they evaluate multiple subsets applying the same algorithm that will be used in the data mining phase), or some filter methods, that employ different metric to evaluate diverse attribute subsets, and that use a variety of search strategies like *Branch & Bound, Sequential Greedy, Best-First, Forward Selection, Sequential Backward Elimination, Floating, Random Search*, among others,
d) Implicitly capture the inter-dependences among attributes, as opposed to filter-ranking methods, that evaluate only the importance of one attribute against the class, like *F-score, Symmetrical Uncertainty, Correlation, Entropy, Information Gain*, etc.

Perceptron for Feature Selection Procedure (PFS)

Given a continuous or numeric dataset, with D attributes previously normalized [0,1], and N randomly chosen instances,

1. Let $Acc(t) = 0$ (generalization accuracy), WithoutImprove = ni (number of accepted epochs without accuracy improvement).
2. While $Acc(t+1)$ better than $Acc(t)$ (ni times).
   a. Train a "relaxed" Perceptron (initial weights in zero)
   b. Test after each epoch, and obtain $Acc(t+1)$
   c. If $Acc(t+1)$ better than $Acc(t)$: save weights and do $Acc(t) = Acc(t+1)$
3. Drop attributes with small absolute inter-connection weights.
4. Use the d remain attributes ($d < D$) to create a model as the predictor with some classifier.

Fig. 1. Perceptron for feature selection procedure.

Then, the objective of this Section is to show the exploratory experimentation realized to verify/ invalidate these assumptions (hypothesis).

We conducted several experiments with 8 real and 11 synthetic datasets to empirically evaluate if PFS can do better in selecting features than other well-known feature selection algorithms, in terms of feature reduction, accuracy and processing time. We choose synthetic datasets in our experiments because the relevant features and its inter-dependencies are known beforehand.

3.1 Details

As first experimentation phase, eight real databases were used. The first one is a database with 24 attributes and 2,770 instances; this database contains information of

Mexican electric billing customers, where we expect to obtain patterns of behavior of illicit customers. The next five datasets are taken form the UCI repository [9] (see Table 2 for details).

Additionally, we experiment with two datasets taken from the NIPS 2003 feature selection challenge[1]. These datasets have very high dimensionality but relatively few instances. Specifically, Madelon database has 500 features and 2,000 instances and Gisette dataset has 5,000 features and 6,000 instances.

On the other hand, we test our proposed method with 11 synthetic dataset. Ten of them with different levels of complexity: to obtain these datasets we use the functions described in [10]. 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 [10]. For each instance, a class label was determined according to the rules that define the functions. We experiment also with the corrAL synthetic dataset [11], that has four relevant attributes (A0, A1, B0, B1), one irrelevant (I) and one redundant (R); the class attribute is defined by the function $Y = (A0 \wedge A1) \vee (B0 \wedge B1)$.

In order to compare the results obtained with PFS, we use Weka´s [12] implementation of ReliefF, OneR, CFS and ChiSquared feature selection algorithms. These implementations were run using Weka´s default values, except for ReliefF, where we define 5 as the neighborhood number, for a more efficient response time. Additionally, we experiment with several Elvira´s [13] filter-ranking methods (see Table 1 for details).

To select the best ranking attributes, we use a threshold defined by the largest *gap* between two consecutive ranked attributes, according to [11] (e.g., a *gap* greater than the average *gap* among all the *gaps*). As inductor we utilized Weka´s J4.8 classifier (J4.8 is the last version of C4.5, which is one of the best-known induction algorithms used in data mining) with 10-fold cross validation (although we experimented using more classifiers, we obtained similar results).

In the case of PFS (codified with C language), we set the learning rate α to 0.6, the maximum epochs equal to 500, and the number of epochs without accuracy improvement *ni* to 15, 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 sub-Section the obtained results are shown.

3.2 Experiments with 8 real databases

Testing over the Electric billing database, we use the selected features for each method as input to the decision tree induction algorithm J4.8 included in the Weka tool. We notice that PFS obtains similar accuracy as Kullback-Leibler-2, but with less processing time (Table 1). In Fig. 2 we depict a detail of the interconnection weights obtained by each attribute when we apply PFS to the Electric billing database. Only for this database, attributes 22, 23 and 24 are random attributes and, following [14], we use this information to mark the threshold between relevant and irrelevant attributes.

---

[1] http://www.nipsfsc.ecs.soton.ac.uk/datasets/

Table 1. J4.8's accuracies with features selected by each method (Electric billing database).

Method	Total features selected	Accuracy (%)	Pre-processing time
Kullback-Leibler 2	9	97.50	6 secs.
PFS	11	97.29	3 secs.
All attributes	24	97.25	0 secs.
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.
CFS	1	90.18	9 secs.
Kullback-Leibler 1	4	90.10	6 secs.
Mutual Information	4	90.10	4 secs.

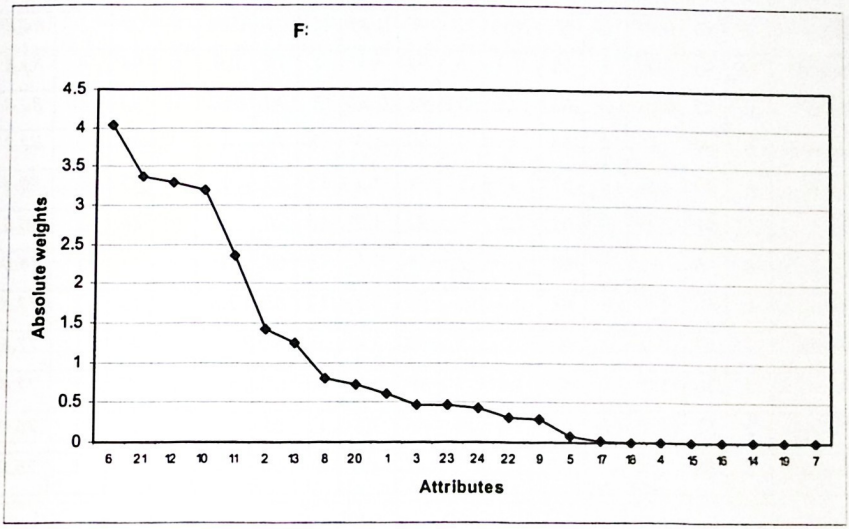

Fig. 2. PFS's weights for each attribute (Electric billing database).

Testing over the five UCI datasets, PFS obtains similar average accuracy as CFS and ReliefF, but with less processing time and good feature reduction (Table 2).

In order to compare our results with real very large databases, we experiment with Madelon and Gisette NIPS 2003 challenge data. In these cases we can not apply Weka or Elvira feature selection tools because they ran out of memory; so, for comparison, we use the results presented by Chen et.al [7]: they apply SVM with a radial basis

function kernel as feature selection method. Table 3 shows results for Madelon and Gisette datasets (N/A means information not mention in [7]).

From Table 3 we can observe that the obtained *BER* using PFS is similar when SVM is applied; on the other hand both, accuracy and *BER*, are poor. The reason for this bad result is because Madelon is a dataset with clusters placed on the summits of a five dimensional hypercube, so, in some sense, is a variation of the XOR problem, a non-linear separable classification problem. Thus, PFS and SVM (still with a kernel function) fail with this database. In the case of Gisette, that contains instances of handwritten digits "4" and "9"; from Table 3 we can see that SVM obtains a superior *BER*, but PFS achieves an acceptable *BER* and accuracy, using few attributes (64 against 913).

Table 2. J4.8's accuracies using the features selected by each method for five UCI datasets.

Method	Vote (16/435)			Horse-c (27/368)			Sick (29/3772)			Sonar (60/208)			Ionosphere (34/351)			Avg. Acc
	TF	Ac	Pt	TF	Ac	Pt	TF	Ac	Pt	TF	Ac	Pt	TF	Ac	Pt	
All atts	16	*96*	0	27	*66*	0	29	*98*	0	60	*74*	0	34	*91*	0	***85.0***
PFS	2	*95*	0.08	3	*68*	0.09	3	*96*	0.4	4	*72*	0.1	5	*93*	0.1	***84.8***
CFS	1	*95*	0.04	2	*66*	0.04	2	*96*	0.25	18	*74*	0.09	8	*90*	3	***84.2***
ReliefF	15	*96*	0.6	3	*66*	0.9	6	*94*	94	4	*70*	0.9	6	*93*	4	***83.8***
SOAP[2]	2	*95*	0.09	3	*66*	0.02	2	*93*	0.12	3	*70*	0.02	31	*90*	0.01	***82.8***
Mutual I	8	*94*	1	4	*68*	1	2	*90*	1.4	18	*73*	1	3	*86*	1	***82.2***
OneR	8	*90*	0.9	3	*67*	1	3	*88*	1.3	12	*72*	1	4	*85*	1	***80.4***
KL-1	5	*91*	1.1	4	*61*	1.2	3	*92*	1.7	16	*70*	1	2	*86*	1	***80.0***
KL-2	4	*88*	1	4	*62*	1.1	2	*89*	1.5	11	*68*	1	3	*83*	1	***78.0***
Matusita	6	*86*	1.9	3	*61*	2.3	2	*91*	3.3	17	*68*	2.5	2	*83*	2	***77.8***
Bhattac	7	*87*	0.9	3	*60*	1	1	*90*	1.4	9	*68*	1	2	*83*	1	***77.6***
Euclidea	8	*86*	1.2	3	*62*	1.4	2	*90*	1.2	10	*67*	1.1	2	*82*	1	***77.4***
ChiSqua	3	*87*	1.4	2	*60*	1.6	3	*88*	1.3	11	*65*	1.2	2	*80*	1	***76.0***
Shannon	3	*86*	1	4	*61*	1.3	2	*87*	1.6	9	*66*	1	2	*80*	1	***76.0***

"(16/435)" means (attributes / instances) for Vote dataset, and so on.
TF=Total features selected Ac=Accuracy (%) Pt=Pre-processing time (secs.)

[2] SOAP's results were taken from [17].

Table 3. Accuracies and *BER* with features selected by each method (Madelon-Gisette).

Database	Method	Features Total (%)	Accuracy (%)	*BER*	Pre-process. time
Madelon	PFS	21 (4. 2%)	58.35	0.4165	48 secs.
	SVM	13 (2. 6%)	N/A	0.4017	N/A
Gisette	PFS	64 (1. 3%)	94. 5	0.0549	3. 3 mins.
	SVM	913 (18. 2%)	N/A	0.0210	N/A

3.3 Experiments with 11 synthetic datasets

The results of applying PFS to 10 synthetic datasets are shown in Table 4. We can observe that the average processing time (column 2) and epochs (column 3) is acceptable. The generalized accuracy obtained for PFS is bad (column 4) but the resulting average accuracy of applies the selected features by PFS to the J4.8 classifier is good (column 5). In columns 6 and 7 we can see that the features selected by PFS are equal or near to the perfect attributes (Oracle column), in almost all cases, except for datasets 3 and 5; the average number of features selected is similar (2.7 vs. 3).

Table 4. PFS with 10 Synthetic Databases.

Database	PFS time (secs)	PFS Epoch	PFS Acc (%)	PFS+J4.8 Acc (%)	PFS Attributes Selected	Oracle
1	3	40	47	100	3-7	3
2	2	24	55	100	1-2-3	1-3
3	2	18	61	68	4	3-4
4	2	17	63	84	1-3	1-3-4
5	3	34	65	82	9	1-3-9
6	4	47	66	99	1-2-3	1-2-3
7	6	59	100	98	9-1-2	1-2-9
8	4	39	100	100	1-2-4	1-2-4
9	4	48	100	97	9-1-2-4	1-2-4-9
10	3	37	99	99	4-8-7-1-2	1-2-4-7-8-9
Avg.	**3.3**	**36.3**	**75.6**	**92.7**	**(2.7)**	**(3)**

Next, we use the selected features obtained by several feature selection methods as input to the decision tree induction algorithm J4.8 (see Table 5).

Table 5. J4.8's accuracies (%) with features selected by each method (10 Synthetic DBs).

Synthetic Database	Method											
	Oracle/All	ReliefF	**PFS**	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	68	100	100	100	100	100	100	100	68	59
4	100	90	84	84	84	84	84	84	84	84	84	84
5	100	100	82	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	97	97	92	85	85	92	85	85	88	85
10	99	80	99	99	97	97	99	97	98	97	97	80
Avg.	99.3	96.1	**92.7**	92.4	91.2	90.5	89.8	89.2	84.9	84.1	83.6	79.6

We use 10-fold cross validation in order to obtain the average test accuracy for each feature subset (in all cases, we obtain similar results using *BER* as quality measure criterion). The column "Oracle/All", from Table 5, represents a perfect feature selection method (it selects exactly the same features that each dataset function uses to generate the class label and, in this case, is equal to the obtained accuracy if we use all the attributes). For dataset 8, only OneR cannot determine any feature subset, because ranks all attributes equally. From Table 5 we can see that the PFS average accuracy is better than several feature selection methods, while worse than only ReliefF.

With respect to the processing time, this is shown in Table 6. We observe that, although PFS is computationally more expensive than ChiSquared and other filter-ranking Elvira's methods, these algorithms cannot detect good relevant attributes or some attribute inter-dependencies; on the other hand, PFS was faster than ReliefF, maintained good generalized accuracy. To have a better idea of the PFS 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 classification 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 6. Average processing time for each method in seconds (10 Synthetic Datasets)

Exhaustive Wrapper	ReliefF	OneR	**PFS**	ChiSquared and Elvira
1,085,049 (12.5 days)	573 (9.55 mins.)	8	3.3	1

When we test with the corrAL synthetic dataset, PFS was the only that can remove the redundant attribute (Table 7); results for FCBF and Focus methods was taken from [11]. Because the corrAL is a small dataset, processing time in all cases is near to zero seconds, and thus omitted.

Table 7. Features selected by different methods (corrAL dataset).

Method	Features selected
PFS	A0, A1,B0, B1
ReliefF	R, A0, A1, B0, B1
OneR	R, A1, A0, B0, B1
ChiSquared	R, A1, A0, B0, B1
Symmetrical Uncertainty	R, A1, A0, B0, B1
$FCFB_{(log)}$	R, A0
$FCFB_{(0)}$	R, A0, A1, B0, B1
CFS	A0,A1,B0,B1,R
Gain Ratio (Weka)	R, A1, A0, B0, B1
Focus	R

4 Discussion and Related Work

A feature selection line of research is using the scale factors produced by, for example, the Principal Component Analysis technique (PCA), the Support Vector Machine (SVM) variants for Feature Selection (SVM-FS) or the Neural Network (NN) paradigms. These approaches all eliminate the attributes whose associated scale factors are close to zero.

Specifically, in the area of NN, there are several methods for post learning pruning inter-connection weights, for example Optimal Brain Damage or Optimal Brain Surgeon [8]. The objective of these approaches is to obtain a simplified NN, conserving good or similar classification power of the complete NN, and therefore, is not directly focused on the feature selection task. Brank et.al. [15] conducted a study to observe how several scale factor feature selection methods interact with several classification algorithms; however, in this research, no information about processing time and feature reduction is presented.

Ruck et.al. [16] apply feature selection using a Multi-layer Perceptron which is much more complex than a Perceptron; they test with gradient descendent and extended Kalman filtering; again, in this work no information about processing time and feature reduction is presented, and they experiment with few and small datasets.

SOAP [17] is a method that operates on numerical attributes and discrete class and has a low computational cost: it counts the number of times the class value changes with respect to an attribute whose values have been sorted into ascending order. SOAP reduces the number of attributes as compared to other methods; nevertheless, the user has to supply the number of attributes that will be used in the final subset. This is a common problem with the *filter-ranking* methods, that output a ordered list of all attributes, according to its relevance.

5 Conclusions and Future Work

We have presented an easy to implement algorithm for feature selection called PFS that is good trade-off among generalization accuracy, processing time and feature reduction. To validate the algorithm we used 8 real databases and 11 synthetic datasets.

The results show that PFS represent a good alternative, simple but effective, compared to other methods, because its acceptable processing time, accuracy and good performance in the feature selection task (feature selection ratio). For the case of the real Electric billing database, PFS obtains similar, or better, accuracy and feature reduction as Kullback-Leibler-2, CFS or ReliefF, but with the half of processing time. Similar comments apply for the rest of the experimented datasets.

The proposed PFS algorithm has several advantages: (1) it requires a linear amount of memory; (2) its generalization accuracy and processing time is competitive against other methods, although its time complexity is a function of the epochs without accuracy improvement (Perceptron´s convergence); (3) does not realize exhaustive or combinatorial search; (4) finds some attribute inter-dependencies; and (5) obtains acceptable feature reductions. The main disadvantage is its limitation to classify only linear separable datasets.

Some future works arise with respect to PFS improvement. For example: apply kernel functions to overcome the linear separability class limitation; try with other learning stopping criteria; realize experiments using a metric (e.g., *F-score*) to do a first attribute elimination, and then apply PFS. Another future work will be the application of the formalism to other very large power system databases such as the national power generation performance database, the national energy control databases, the energy de-regulated market and the Mexican electric energy distribution database.

References

1. Guyon, I., Elisseeff, A., An introduction to variable and feature selection, Journal of machine learning research, 3 (2003) 1157-1182.
2. Piramuthu, S., Evaluating feature selection methods for learning in data mining applications, Proc. 31st annual Hawaii Int. conf. on system sciences (1998) 294-301.
3. Molina, L., Belanche, L., Nebot, A., FS algorithms, a survey and experimental evaluation, IEEE Int.conf.data mining, Maebashi City Japan (2002) 306-313.
4. Mitra, S., et.al., Data mining in soft computing framework: a survey, IEEE Trans. on neural networks, vol. 13, no. 1, January (2002) 3-14.
5. Kohavi, R., John, G., Wrappers for feature subset selection, Artificial Intelligence Journal, Special issue on relevance (1997) 273-324.
6. Gallant, S.I.: Perceptron-Based Learning Algorithms, in IEEE Transactions on Neural Networks, 1 (1990) 179-191.
7. Chen,Y., Lin,C., Combining SVMs with various feature selection strategies. To appear in the book "Feature extraction, foundations and applications", Guyon, I. (ed) (2005).
8. Jutten, C., Fambon, O., Pruning methods: a review, European symposium on artificial neural networks, April (1995) 129-140.
9. Newman, D.J. & Hettich, S. & Blake, C.L. & Merz, C.J. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science (1998).

10. Agrawal, R., Imielinski, T, Swami, A., Database mining: a performance perspective, IEEE Trans. Knowledge data engrg. Vol. 5, no. 6 (1993) 914-925.
11. Yu, L., Liu, H., Efficient feature selection via analysis of relevance and redundancy, Journal of Machine Learning Research 5 (2004) 1205-1224.
12. www. cs.waikato.ac.nz/ml/weak (2004).
13. www. ia.uned.es/~elvira/ (2004).
14. Stoppiglia, H., Dreyfus, G., et.al., Ranking a random feature for variable and feature selection, Journal of machine learning research, 3 (2003) 1399-1414.
15. Brank, J., Grobelnik, M., Milic-Frayling, N. & Mladenic, D., Interaction of feature selection methods and linear classification models. Proceedings of the ICML-02 Workshop on Text Learning, Sydney, AU (2002).
16. Ruck, D.W., Rogers, S.K., Kabrisky, M. Feature selection using a multilayer perceptron. Journal of Neural Network Computing, 2 (1990) 40-48.
17. Ruiz, R., Aguilar, J., Riquelme, J., SOAP: efficient feature selection of numeric attributes, VIII Iberamia, workshop de minería de datos y aprendizaje, Spain (2002) 233-242.