ATTRIBUTE SELECTION FOR CLASSIFICATION

Patricio Serendero
Dep. of Electronics & Computer Science, U. of Algarve,
Campus Gambelas, Faro, Portugal

Miguel Toro
Dep. of Languages and Inf. Systems, U. of Seville,
Av. Reina Mercedes s/n, Seville, Spain

ABSTRACT

The selection of attributes used to construct a classification model is crucial in machine learning, in particular with instance similarity methods. We present a new algorithm to select and rank attributes based on weighing features according to their ability to help class prediction. The algorithm uses the same structure that holds training records for classification. Attribute values and their classes are projected into a one-dimensional space, to account for various degrees of the relationship between them. With the user deciding on the degree of this relation, any of several potential solutions can be used as criterion to determine attribute relevance. This low complexity algorithm increases classification predictive accuracy and also helps to reduce the feature dimension problem.

KEYWORDS

Data Mining**, c**lassification, attribute selection, relevant attributes, exclusive attributes

1. INTRODUCTION

Two well-known problems arise when building classifiers which use decision tree structures and instance-based methods. First, the input order of attributes determines heavily the predicting skills of the algorithm. Choosing the wrong order of attributes (or features) could move apart in the hyperspace, values that otherwise would be closer. Secondly, some attributes contribute more than others in building the prediction hypothesis [Aha, 1994]; attributes considered *irrelevant* increase the computational cost and can mislead distance metrics calculations [Indyk, 2000]. This is particularly true for nearest neighbour algorithms, which find the class of unknown instances using the geometric concept of proximity or similarity built around the notion of distance between points in an *n-dimensional* space. As the position of any instance is defined by the value of its attributes, if these are not relevant, then the basic assumption is violated. Based on these, attributes are classified as *relevant* or *irrelevant*, in terms of their degree of contribution to the classification model [Kohavi et al., 1997; Lebowitz, 1985][1]. Feature selection is used for this reason, and is defined as the process of identifying and removing as much irrelevant and redundant information as possible with the goal of improving classification accuracy. Because we use a tree structure to hold instances, our method requires first setting in the input order, attributes with larger discriminatory power with respect to classes, as done in some rule induction algorithms [Quinlan, 1986; Cover et al., 1997].

The complexity of feature selection algorithms depends on the number and quality of its attributes. Searching relevant attributes cannot be exhaustive in many cases. The dimension of datasets is exponential in the number of attributes. Hence, verifying every possible combination of attributes is, in many cases, out of the question [Lesh et al., 1998]. Because of this, we developed a low computational algorithm with the following goals: a) Establish a criterion to determine *relevant* attributes. b) Rank attributes at preprocessing time based on this relevance and c) Reduce the number of attributes. The overall goal is to diminish the

[1] These authors still identify *redundant* attributes, a situation which we do not address here.

algorithm's complexity as well as to increase or at least preserve its predictive skills. Our results show a steady improvement of our classification algorithm when ordered features are used with this simple method.

2. DEFINITIONS

2.1 Basic definitions

Notation $\{r \hat{I} R, c \hat{I} L | \forall \ exp(r) \bullet exp1(r)\}$ *stand* for the set of values of $exp1$ when r and c take values in R and L, and exp is true. If $exp1(r) = r$ then the expression is reduced to $\{r \hat{I} R | \forall \ exp(r)\}$.

Let's assume the existence of a data set R composed of a finite set of N records r of type:

$$r = <v_1, v_2,.., v_i,.., v_n, c>, v_i \hat{I} T_i, c \hat{I} L \text{ and } v_i(r) = v_i . \quad (1)$$

Each record is formed by the Cartesian product of a finite sequential set of attributes A_i, belonging to set S having v_i values $\in T_i$. Each record is associated with one of m classes $c_1, c_2,..,c_m$, belonging to set L. Each attribute's domain is partitioned into a finite number of user-defined intervals within domain T_i. These intervals are represented by integers with values from 1 to s_i. We assume the existence of function $ord_i()$, which converts an attribute value into the corresponding interval value:

$$p_i = ordi(v_i), v_i \in T_i . \quad (2)$$

Using function ord_i in (2), every attribute value $v_i \in T_i$ is converted into a pattern element with value p_i. Every p_i value will fit into one of s_i partitions belonging to attribute A_i. Together, all p_i elements form a vector called pattern p containing n element values. We call P the set of all patterns p obtained from R.

$$p = <p_1, p_2,.., p_i,.., p_n>, \forall \ p \in P, \forall \ p_i \in (1..s_i) \text{ where } p_i = ord_i(v_i(r)) . \quad (3)$$

Notice that the number of partitions s_i is not the same for all attributes[2].
We define functions $pat(r)$ and $label(r)$ such that:

$$pat(r) = p \text{ and } label(r) = c \text{ if } r = <v, c>, v = <v_1,v_2,..,v_n> . \quad (4)$$

In every pattern p from (3) we find n sub-patterns q_i, which correspond to its prefix:

$$q_i = <p_1, p_2,.., p_i>, i = (1..n). \text{ So } p = <q_i, u> \text{ where } u \text{ is the suffix portion.} \quad (5)$$

We define function $freq(p)$, which returns the number of records exhibiting pattern p:

$$freq(p) = \#\{r \hat{I} R \mid pat(r) = p\} . \quad (6)$$

Function $freq$ can be equally applied to a sub-pattern q_i:

$$freq(q_i) = \#\{r \hat{I} R \mid pat(r) = <q_i, u>\} . \quad (7)$$

We define function $freq()$, which is applied to the k^{th} interval from attribute A_i, giving the total number of records in k. A variation of this function includes restricting the number of objects belonging to class c.

[2] This is due to changes in the number of partitions for selected attributes as we show later in section 6.

$$freq(A_{i,} k) = \#\{r \in R \mid ord_i(v_i(r)) = k \}$$
$$freq(A_{i,} k, c) = \#\{r \in R \mid ord_i(v_i(r)) = k \wedge label(r) = c \} \quad (8)$$

Assuming that partition granularity is such that allows all patterns to have a given label, we define function *labels(p)*, which return the set of labels associated with the subset of records with pattern *p*.

$$labels(p) = \{r \hat{I} R, c \hat{I} L \mid pat(r) = p \wedge label(r) = c \bullet c \} . \quad (9)$$

The number of class labels attached to a given pattern *p* is:

$$nlabels(p) = \# labels(p) . \quad (10)$$

Using the Equation in (10) we define the *strength of a pattern* as:

$$strength(p) = \#\{ i \hat{I} (1..n) \mid nlabels(q_i) = 1 \} \quad (11)$$

2.2 Other definitions

Definition 1. A*ttribute A_i is said to be semi-exclusive for partition k, iff function semk() is true.* Function *semk()* is defined as:

$$semk(A_i, k, \varphi) = \exists c \hat{I} L \bullet ((freq(A_i, k, c) / freq(A_i, k)) \geq \varphi) \quad (12)$$

Parameter j is a user-defined value representing the fraction of records in interval *k* with class *c*. The special case when j = 1, is referred to as an *exclusive interval*, meaning that all A_i values in this interval belongs to the same class. Attributes exhibiting this type of value are also described in the literature as *primary [*Turney, 1996], [Kohavi et al., 1997].

Definition 2. *The degree of exclusiveness of a pattern corresponds to the fraction of p_i elements within pattern p conforming to the exclusive values property.* It is calculated with the following function:

$$Semp(p, \varphi) = \#\{i \hat{I} (1..n) \mid semk(A_i, p_i, \varphi)\} / n . \quad (13)$$

Definition 3. *The degree of relevance of attribute i denoted with $d_{i,}$ is the ratio between the total number of records in semi-exclusive partitions and N.* Large values of *d* mean a more *relevant* attribute.

$$d_i = \#\{r \hat{I} R, k \hat{I} (1..s_i) \mid semk(A_i, k, \varphi \wedge ord_i(v_i(r)) = k \bullet r\} / N . \quad (14)$$

The opposite represents *irrelevant* attributes.

Definition 4. The shape of a pattern is defined as:

$$shape(p) = (p_2 - p_1), .., (p_n - p_{n-1}) \quad (15)$$

And the distance between two patterns is:

$$d(p, p') = \sum_i | p_i - p_i' | \quad (16)$$

Using equations from Definition 4 we can define the following:

Definition 5. The shape similarity function between shapes is defined as:

$$sf(p^1,p^2)=d(shape(p^1), shape(p^2)) \quad (17)$$

In Figure 1 we show semi-exclusive intervals and the calculation of d for two attributes, A_1 and A_2. An asterisk means more than one class for a given partition; i.e. *nlabels* $(p_i) > 1$.

A_1

s_i	0	1	2	3	4	5	6	7	8	9	total
freq	77	32	62	49	79	20	12	23	11	43	408
class	*	*	*	*	*	*	*	*	4	4	54

d 54/408=0.132

A_2

s_i	0	1	2	3	4	5	6	7	8	9	total
freq	227	23	35	17	17	14	12	13	5	45	408
class	*	2	*	*	4	4	*	4	*	4	112

d 112/408=0.274

Figure 1. Attribute projection in one-dimensional space for attributes A_1 and A_2.

3. OVERVIEW OF THE CLASSIFICATION ALGORITHM

We have previously developed a classification algorithm for supervised learning based on instances and the nearest neighbour paradigm in [Serendero et al., 2001]. Classification is done extracting two nearest patterns p^+ and p^- with respect to a query pattern p^x. The extraction of p^+ is done first in a recursive way. Any sub-pattern q_i is of the form $q_i =< q_{i-1}, k>$. Starting with $i = 1$, assuming an empty sub-pattern $q_0{}^+ = <>$ and knowing element q_{i-1}, the problem consist in finding the next sub-pattern by calculating some value for k^+, which satisfies the following property:

$$\forall k \in K(q_{i-1}) \bullet (|k^+ - k^x| \leq |k - k^x|) . \quad (18)$$

The set $K(q)$is defined by:

$$K(q) = \{ k \mid <q, k> \in P\} \quad (19)$$

Hence, k^+ is the closest element to k^x among the elements in $K(q_{i-1})$. If two values of k^+ verifies Equation (18), then we chose the one where $freq(q_i^+)$ is a maximum. The algorithm searches next for pattern p^-. The search mechanism is the same as for p^+, but with set $K(q)$ using this time pattern p^+ previously calculated:

$$K(q) = \{k \mid <q, k> \in P \wedge ((nlabels(<q, k>) > 1) \vee ((labels(<q, k>) \cap labels(qi+)) = \varnothing)\} \quad (20)$$

The new pattern should have its class distinct from the p^+ class.

A final step consists in applying function *merit()* to both selected patterns. The pattern with the largest *merit(p)* is the winner. This function represents an aggregation of several criteria, and is defined as:

$$merit(p) = \sum_i w_i \cdot a_i(p) \quad (21)$$

Every criterion in a_i has weight w_i. These criteria a_i (i = 1..6), numbered in no particular order are:

- The d*egree of* exclusiveness of a pattern, calculated as $a_1(p) = semp(p)$ from (13).
- The *strength of a pattern*, calculated from Equation (11). So $a_2(p) = strength(p)$.

- The *similarity shape* from Definition 5 and Equation (17) applied to a pattern against p^x to calculate the most similar shape: $\alpha_3(p) = sf(p^x, p)$.
- A number related to the *distance* to p^x, calculated as $\alpha_4(p) = d(p^x, p)$ from Equation (16).
- The *frequency* of a pattern, where a a larger frequency is a better option, other conditions being equal. Function frequency is calculated as $\alpha_5(p) = freq(p)$ using function in (6).
- Let be *mc* the *majority class*, the class with the largest frequency then $\alpha_6(p)$ equals 1 if $label(p) = mc$ and 0 otherwise.

Weights are obtained preprocessing the training dataset. The weight of each criterion corresponds to its degree of accuracy correctly classifying patterns. Each criterion is tested individually by setting all other weights to zero. The goal is to optimise *T*, the *degree of accuracy* of each criterion. It is calculated as:

$$T = N^o \text{ of records correctly classified / } N; \text{ and being the error } \varepsilon = 1 - T \qquad (22)$$

The application of function *merit ()* at running time, allows selecting the best pattern. Its class is assigned to p^x. A more detailed explanation of this process is left for a next article.

In the algorithm implementation all training patterns are stored into a *trie* [Fredkin, 1960], including frequencies and class information at the sub-pattern level. These structures have proven to be very fast on search problems [Bergman, 1994;Merret et al., 1996; Alber et al., 2001], which is one of the main problems in the near neighbour paradigm. The large storage required by tries is partially solved keeping the file on disk. Also, several known compress tools are available for tries such as *Patricia trees* [Gonnet et al., 1991], *X-tree* [Berchtold et al., 1996] and *Burst tries* [Heinz et al., 2002].

4. ATTRIBUTE SELECTION

4.1 Determining most *relevant* attributes

In general, our method ranks attributes by their capacity of predicting classes without taking directly into consideration other attributes from the original sequence. We postulate that this capacity increases, when an attribute exhibits a larger *degree of relevance* as stated in Definition 3. For instance, attribute A_i is *relevant* if some α percentage of its instances with value v_j is associated with class c_l. In this sense, we soften the Boolean definition found in [Kohavi et al., 1997]. Our objective is *to find the most discriminative attributes from the point of view of usefulness to the predictor, with the purpose of improving its prediction accuracy* [Guyon, 2001]. This heuristic criterion has been used successfully before [Liu et al., 2000].

A sub-pattern q of size i can be a common prefix for distinct labels, i.e. when $nlabels(q_i) > 1$, representing areas of larger entropy with respect to classes in the data hyperspace, not allowing any conclusion on class membership. Inversely, sub-patterns where $nlabels(q_i) = 1$, represent homogeneous regions, where smaller values for i (shorter sub-patterns) represent larger areas. If a new instance to be classified falls into one of these areas, its chances of correct classification increase. Most of its neighbours will share the same label. For this reason, we are interested in looking at the entire data space from the viewpoint of attributes with larger number of examples where classes are "*visible*" directly from them, thus avoiding endless combination of possibilities as done in traditional methods [Kohavi et al., 1997; Miller, 1990; Brodley et al., 1995]. We consider these attributes more *relevant* than others. Now, the shortest sub-pattern contains only one element $q_1 = <p_1, c>$, usually associated with several classes. We want to find which attributes perform better than others in this situation. To do this, all values of an attribute are projected into a one-dimensional space previously partitioned into equal interval widths. In the trie structure used for implementation this corresponds to build the tree with a single level. This resembles the 1R classification system [Holte, 1993] although in this system the ranking of features is based directly on error rates. In our case we are interested in the total number of sub-patterns q_1, found in *semi-exclusive intervals* (from Definition 1), for a given attribute A_i and j values. Attributes showing more individual patterns in these intervals are also more *relevant* (Definition 4). Based on this, relevance can be set as a measure of attribute comparison as explained next.

4.2 Ranking Attributes by their relevance

Ranking attributes is done knowing which attributes are comparatively more *relevant* than others. To do this we compute each *attribute's relevance* from (14), and rank them in decreasing order according with the value of *d*, as the example shown in Fig. 1 for A_1 where $d = 0.132$. This ranking gives as result list *ß*, which represents all attributes ordered by their degree of relevance:

$$ß = <d_1, d_2, ..., d_i, ..., d_n>, \ |d_i| \geq |d_{i+1}| \ i \in (1..n). \qquad (23)$$

Attributes where $d_i = 0$, are orderly pushed to the list's end. The list represents the input order of attributes used by the classification algorithm. As shown in Section 5 doing this improves the predictive accuracy of the classification algorithm.

4.3 Reducing the number of attributes by their degree of relevance

Reducing the number of irrelevant attributes drastically reduces the running time of a learning algorithm and yields a more general concept, easier to understand by the domain expert. This reduction can be achieved using the concept of *attribute exclusiveness* as defined in Section 3. This is done by eliminating from list *ß* in (23) all attributes where d is small or zero. This reduction, though, cannot be done without a cost. The trade-off is done at the expense of losing some predictive accuracy. With this constrain in mind, our goal is *to find a minimum subset of attributes S' such that when the classification algorithm is applied accepting some error e, we can obtain a new predictive accuracy T´* as $S' \subset S$, that satisfies $T' \leq T + e$. The new set *S'* obtained from list *ß* in (24) includes only *relevant* attributes discarding all others. The classification algorithm rebuilds the tree using the new sequence in *S'*. At running time, and using some user-user-defined error over the existing prediction value *T* from (22), a new *T'* value is obtained. Error *e* is a function of cost and quality [Brodley, 1995]. If Equation (24) is satisfied and $(T' - T) \leq e$, then *S'* is adopted as the new set of attributes. Otherwise, the threshold value of *f* should be reduced and list *ß* rebuilt. As a result this will increase the number of attributes in subset *S'* and hopefully will also increase prediction accuracy *T'* diminishing the value of *e*.

5. RESULTS

We have tested these techniques on seven datasets from the UCI repository [Murphy et al., 1994]. All records with unknown attribute values were eliminated. Ten–fold cross validation was used. Accuracy results were averaged.

Table 1. New Attribute Order using f = 0.75

Nº	Number Attrib.	Dataset	N. º Records		New Attribute Order Number represents original ordinal number (**Bold face** = attribute is *relevant*)	Relevant Attributes (%)
			Training	Test		
1	24	Hypothyroid	1598	1063	**18,23,21,1,20,22,7,5,13,24**,19,17,16,15,14, 12,11,10,9,8,6,4,3	41.7
2	24	Dermatology	218	140	**20,22,27,29,6,12,8,25,33,34,24,15,10,31,26, 30,14,23,7**,32,28,21,19,18,17,16,13,11,9,5,4, 3,2,1	57.6
3	33	Adult	28468	15060	**3,9,14,2,4,5,7,8,13,6**,1,10,11,12	71.4
4	13	Diabetes	462	306	**5,6,2,7,4,8**,3,1	75.0
5	12	Forest covert	15120	565892	**1,10,5,6,4,12,8,7,9,3**,11,2	83.3
6	12	Pendigits	7494	3498	**6,12,3,7,11,4,2,15,5,14,16,8,1,10**,9,13	87.5
7	16	Cancer–W.	407	273	**7,2,1,8,3,9,4,6,5**	100.0

The average decrease in error rate after ordering attributes of 4.3% in Table 2 is similar to results reported for different datasets in a previous article where this technique was applied [Serendero et al., 2001]. This

confirms that the gain in predictive accuracy is significant and steady when applied to different data domains. It also confirms the need for attribute ordering when tree structures and instance-based methods are

Table 2. Variation in predictive error rate after ordering and reduction in the number of attributes

Nº	Datasets	Classification: Predictive Error Rate (%)			Variation (%) due to:	
		Original (A)	Ordered (B)	Reduced(C)	Order (B-A)	Reduction (C-B)
1	Forest covert	28.2	20.5	23.9	-7.7	+3.4
2	Dermatology	10.2	4.5	6.5	-5.7	+2.0
3	Diabetes	27.8	22.2	22.5	-5.6	+0.3
4	Pendigits	9.3	5.3	2.0	-4.0	-3.3
5	Cancer –W.	5.5	2.2	1.8	-3.3	+0.4
6	Adult	18.9	16.0	10.6	-2.9	-5.4
7	Hypothyroid	1.6	0.7	1.1	-0.9	-0.4
			Average		-4.3	

used, as is our case. Although not conclusive due to experiment size, variation in error rates after attribute reduction (C-B) seems to be related to the intensity of attribute *"pruning"* (Table 1). As expected, a large reduction in the number of attributes results in greater error rates. In 43% of cases reducing the number of attributes increases predictive accuracy, meaning that the ranking method works well. Attributes at the list's end are indeed irrelevant for predictive purposes. Accuracy in the remaining datasets shows a small loss not greater than 3.3% if compared with a 56% average reduction on the number of attributes, and hence, in problem complexity. In the face of large datasets with high dimensions where traditional feature reduction methods represent very high computational costs, this method can be a fair solution.

Our feature selection methods integrated into the classification algorithm helps to produce competitive results as shows its comparison with Quinlan's landmark tool C4.5 (also known as See5) in Table 3. In general, our classifier shows better performance than C4.5 for datasets with smaller number of classes and worse when the opposite is true. This is explained by the fact that in order to speed up the search process, our algorithm only looks for two close patterns accounting for only two classes.

Table 3. Comparing classification accuracy against the popular C4.5

Nº	Dataset	Error in accuracy (%)	
		C4.5	Ours
1	Adult(Census 94,USA)	14.6	10.6 ± 1.2
2	Forest cover	29.1	20.5 ± 2.5
3	Cancer-W	5.8	2.2 ± 0.2
4	Hypothyroid	0.7	0.7 ± 0.3
5	Dermatology	3.9	4.5 ± 0.3
6	Pima Indian Diabetes	26.8	22.2 ± 1.4
7	Pendigits	3.4	2.0 ± 0.5

Source for C4.5 results: [Chou et al., 2000; Li et al., 2000; Murphy et al., 1994; Chawla, et al., 2001].

6. DISCUSSION

In this article we present a simple and low-cost feature selection method, useful for algorithms using decision trees and instance-based methods in supervised learning. Results show on average a decrease of over 4% in classification predictive error when attributes are ranked by their degree of relevance. This result is consistent with previous results obtained on different datasets and offers a simple solution to ranking features.

Accuracy increases even further in 43% of cases after attribute reduction. This not only confirms the correctness of this simple method for ranking attributes, but also the fact that it works as a good filter indication on the predictive skills of attributes. In the remaining 57% of cases a loss in prediction not greater than 3.3% did represent eliminating an average of 56% of attributes for all datasets considered. This is very important to help reducing algorithm complexity in high dimensional datasets and therefore the comprehension of the domain expert in data relations.

Future work includes modifying slightly the algorithm to extract as many patterns as classes exist in a dataset with the goal of increasing predictive accuracy in datasets with more than two classes. This should not significantly increase search time if we consider the excellence of tries with respect to search performance.

REFERENCES

Aha, D. W, Bankert, R., 1994. Feature Selection for Case-Based Classification of Cloud Types: An Empirical Comparison. In *Case-Based Reasoning: Workshop, Technical Report WS-94-01*, CA, USA., AAI Press.

Alber, I. E., et al.,, 2001. Fast Retrieval of Multi and Hyperspectral Images Using Relevance Feedback. P*roceedings of the International Geoscience and Remote Sensing Symposium.* Vol. 3, pp. 1149-1151.

Andersson, A, Nilsson, S., 1994. Faster Searching in Tries and Quad trees – An Analysis of Level Compression. *Proceedings. of the Second European Symposium on Algorithms*, pp 82-93.

Berchtold, S. et al., 1996. The X-tree: An Index Structure for High-Dimensional Data. *Proceedings of the 2nd International Conference on Very Large Databases*, Bombay, India, pp. 28- 39.

Berman, A.P., 1994. A New Data Structure For Fast Approximate Matching. *Technical Report, 1994-03-02, Dept. of Computer Science, University of Washington.*

Brodley, C. E., 1995. Multivariate Decision Trees. *Machine Learning*, Vol. 19(1), pp. 45-77.

Chawla, N. et al., 2001. Creating Ensembles of Classifiers. *Proceedings of the 2001 IEEE International Conference on Data Mining*, CA, USA, pp. 580-81.

Chou, Y., Shapiro, L. G., 2000. A Hierarchical Multiple Classifier Learning Algorithm. *Proceedings of the Intl. Conference on Pattern Recognition*, Vol. 2, pp. 152-155.

Cover, T. Hart, P. 1967. Nearest neighbour pattern classification. *IEEE Transactions on Information Theory*, Vol. 13:1, pp 21-27.

Dougherty, J., et al., 1995. Supervised and Unsupervised Discretization of Continuous Features. *Proc. of the 12th International Conference on Machine Learning*, Morgan Kaufman Publisher, San Fco. USA., pp. 94-202.

Fredkin, E., Trie Memory, 1960. *Communications of the ACM*, Vol.3: 9, pp. 490-500.

Gonnet, G. H., Baeza-Yates, R., 1991. *Handbook of Algorithms and Data Structures*, Addison-Wesley, UK.

Guyon, I., 2001. Introduction to the NIPS 2001, *Workshop on Variable and Feature Selection.* BC., Canada.

Heinz, S., Zobel, J., 2002. Williams, H.E., Burst Tries: A Fast, Efficient Data Structure for String keys. *ACM Transactions on Information Systems*, 20(2): pp. 192-223.

Holte, R.C., 1993. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning*, 11, pp 63-91.

Indyk, P., 2000. Dimensionality Reduction Techniques for Proximity Problems. *Proc.11th.ACM-SIAM Symposium on Discrete Algorithms*, pp. 371,378.

Kohavi, R, John, G., 1997. Wrappers for Feature Subset Selection. *Artificial Intelligence Journal, Special issue on relevance*, Vol. 97, Nº 1-2, pp. 273-324.

Lebowitz, M., 1985. Categorizing Numeric Information for Generalization. *Cognitive Science*, (9) pp. 285-308.

Li, J., Dong, G., Ramamohanarao, K., 2000. Instance-Based Classification by Emerging Patterns. *Proc.Fourth European Conf. On Principles and Practice of Knowledge Discovery in Databases*, Springer-Verlag, pp. 191-200.

Lesh, N., et al., 1998. Mining Features for Sequence Classification. *MERL, Technical Report: TR98-22.*

Liu, B., et al., 2000. Improving an Association Rule Based Classifier. *4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Lyon, Springer-Verlag, pp. 504-509.

Merrett, T.H. et al., 1996. Database Structures, Based on Tries, for Text Spatial and General Data. *Int. Symposium on Cooperative Database Systems for Advanced Applications*, Kyoto, pp. 316:324.

Miller, A.J., 1990. *Subset Selection in Regression.* Chapman & Hall, New York, USA.

Murphy, P.M., Aha, D.W., 1994. *UCI Repository of machine learning databases.*University of California, Irving, Department of Information and Computer Science. Patrick M. Murphy (Repository Librarian).

Quinlan, J., 1986. Induction of Decision Trees. *Machine Learning*, Vol. 1, pp. 81-106.

Serendero, P., Toro, M., 2001. Supervised Learning Using Instance-based Patterns. *Proceedings of the IX Conferencia de la Asociación Española para la Inteligencia Artificial*, Spain, Vol. I, pp. 83-92.

Turney, P.D. 1996. The management of context-sensitive features: A review of strategies. *13th International Conference on Machine Learning (ICML96)*,Bari, Italy, July, pp. 60-66.