USE OF ENTROPY IN THE KNOWLEDGE DISCOVERY ALGORITHMS WHICH GENERATE RULES ACCORDING TO COVERING APPROACH

Ömer AKGÖBEK[1], Ercan ÖZTEMEL[2]

[1] *Harran University Engineering Faculty,* Department *of Industrial Engineering, Sanliurfa, Turkey*
e-mail :akgobek@harran.edu.tr
[2] *Marmara University Engineering Faculty, Department of Industrial Engineering, İstanbul, Turkey*
e-mail : eoztemel@eng.marmara.edu.tr

ABSTRACT

The objective of this paper is to introduce the use of entropy for knowledge acquisition in the algorithms which use the covering approach in inductive learning. REX-1 and REX-2 algorithms, which generate rules based on the covering approach, are compared with other algoritbms using the same principle. These algorithms which adapt the mentioned approach generate rules using the search methods. As is used in the algorithms generating the decision tree, the entropy can be used as well in algorithms which utilize the covering approach. While generating rules by search methods, it is vital that the algorithms give priority to the attributes with high complexity in an example set. However, use of entropy attaches the priority to the attributes with lower complexity. ID3 and C4.5 algorithms may be cited among those using the entropy. Instead of direct rule generation, but they use the decision tree to induce rules.

Keywords - Knowledge Discovery, Rule Extraction, Decision-Trees, Entropy

KAPSAMA YAKLAŞIMINA GÖRE KURAL ÜRETEN BİLGİ KEŞFİ ALGORİTMALARINDA ENTROPİ KULLANIMI

ÖZET

Bu yayının amacı, endüktif öğrenmede kapsama yaklaşımını kullanan algoritmalarda bilgi kazancı için entropi kullanımını sağlamaktır. Kapsama yaklaşımına göre kural üreten REX-1 ve REX-2 algoritmaları aynı metodla kural üreten diğer algoritmalarla karşılaştırılacaktır. Bu algoritmalar arama metodlarını kullanarak kural üretirler. Entropi, karar ağacı üreten algoritmalarda kullanıldığı gibi kapsama yaklaşımını kullanan algoritmalarda da kullanılabilir. Arama metodları tarafından kurallar üretilirken örnek setindeki karmaşıklığı yüksek olan özelliklere öncelik verilmesi kaçınılmazdır. Ancak entropi kullanımı karmaşıklığı daha az olan özelliklere öncelik verir. Entropi kullanan algoritmalar arasında ID3 ve C4.5 sayılabilir. Fakat bu algoritmalar doğrudan kural üretmek yerine karar ağacını kurallara dönüştürürler.

Anahtar Kelimeler – Bilgi keşfi, Kural çıkarma, Karar ağaçları, Entropi

I. INTRODUCTION

Inductive learning is a process that uses sets of training examples to learn a concept. Many methods have been suggested to generate decision rules from learning examples. For that purpose, some algorithms are needed for generating rules which determine the description of the concepts to be learned. But the description bears only one out of many possible interpretations of the training data and, yet, it may present a meaning completely irrelevant to the meaning of the concept. Therefore, an inductive learning algorithm should be sufficient to draw multiple conclusions from learning examples [1].

A major problem in the design of learning algorithms is the generation of a complex description from noisy examples. Learning from noise corrupted data may result in a large number of complicated decision rules describing trivial instances. Hence, the resulting concept description may not reflect general situations. We call such a "overfitting" which refers to a tendency to force the rule induced from training data to agree with these data too closely, at the cost of generalization to other examples. Poor concept

description may also cause the overfitting. To overcome the noise-caused overfitting, many studies have been performed and some methods have been suggested. Among the solutions suggested, two approaches are mentioned here. The first is to allow a certain degree of inconsistent classification of training examples so as to describe the basic attributes of a concept in a general way. This approach is employed by the ID family of algorithms [2,3]. The C4.5 algorithm by Quinlan is a descendant of ID3 which converts its tree into rules and prunes both rule conditions and whole rules[4]. The second approach is to eliminate unimportant rules and only keep the ones covering the largest number of examples and consider them as general description of a concept [1].

I.1 Decision Tree and Rule-Based Algorithms

These algorithms generate concept descriptions from examples by following specific procedures, and by using a set of heuristics in separating examples of one class from other classes. Such algorithms are classified into two major families. The first is the decision tree-based algorithms, and the second is rule-based algorithms. An example of the first family algorithms is the ID family of algorithms such as ID3[2] and C4. The AQ family of algorithms is the examples for the second type of algorithms. Popular algorithms using this technique are the AQ family of algorithms [5,6], RULES family [7,8,9], ILA[10], REX-1[11] and REX-2[12].

I.1.1 Decision tree-based algorithms

These algorithms generate decision trees based on the divide-and-conquer approach. Decision tree-based algorithms usually use the information entropy measure to grow a decision tree by searching for a feature that gives maximum information gain. The procedure of growing a decision tree continues by dividing examples into smaller subsets until the training examples are correctly classified based on a user-specified termination criterion.
In real-world applications, training examples are usually insufficient to define a concept description uniquely. Therefore, learning algorithms need a flexibility to produce different generalizations from given examples. In decision tree-based algorithms, the description of a subset of examples in a leaf node of a tree is uniquely described as series of feature tests from the root to the bottom of a tree. This approach does not have the flexibility of describing a target concept in different ways.

I.1.2 Rule-based algorithms

These algorithms generate rules according to the covering approach. Rule-based algorithms have the ability to generate multiple descriptions of a concept. An example is the AQ15 algorithm where the empirical learning was treated by Michalski as the general covering problem[5]. The basic term of a cover, used in the AQ family of algorithms, implies that there may be multiple covers to cover positive training examples. This resulted in the development of procedures that produce a quasioptimal solution in polynomial time. Generally, AQ algorithms follow a greedy heuristics that tries to include/exclude as many as possible of positive/negative examples in searching for a complex. The AQ algorithms use a set of user specified description preference criteria to describe a subset of positive examples covered by a complex[1]. REX-1 and REX-2 are the type of algorithms which generate rules according to the covering approach and use the entropy in the process.

II. INFORMATION MEASUREMENT, ENTROPY AND KNOWLEDGE GAIN

Roughly speaking, entropy is the degree of disorder of a system. It is such an important physical concept that many disciplines employ entropic functions such as thermodynamic entropy, topological entropy. As the disorder of a system increases, any increasing function may be used as an entropic function [13,14]. Information value of example set is computed by equation (1),

$$Info(S) = -\sum_{i=1}^{m} \frac{S_i}{|S|} \cdot \log_2 \left(\frac{S_i}{|S|} \right) \quad (1)$$

where m denotes the number of classes in the example set, $|S|$ denotes the number of examples in the set, and S_i denotes the number of examples of the i^{th} class.
Entropy values are computed for each value in a class. Let T_1, T_2, ...T_n show the subsets which include the examples with an element. k denotes the number of elements in a subset, $freq(C_k, T)$ denotes the number of examples of C_k class in subset T and |T| denotes the total number of examples in the subset. Therefore, entropy for each value is computed with equation (2).

$$E(T) = -\sum_{i=1}^{k} \frac{freq(C_k, T)}{|T|} \cdot \log_2 \frac{freq(C_k, T)}{|T|} \quad (2)$$

Entropy for an attribute is equal to the addition of entropy value multiplied with the probability of the value (3).

$$E(A) = \sum_{i=1}^{n} \frac{T_i}{|T|} \cdot E(T_i) \quad (3)$$

where A denotes a attribute, n the number of values in a attribute, and $E(T_i)$ the entropy of i^{th} value.
Information gain of a attribute equals to the information value of the example set minus the entropy of the attribute. The information gain for attribute A in example set S is computed with equation (4). Info(S) is the same for all attributes, as it is the information gain for the whole example set.

$$Gain(S,A) = Info(S) - E(A) \quad (4)$$

Split information is computed for each attribute with equation (5).

$$SplitInfo(S,A) = -\sum_{i=1}^{A_n} \frac{S_i}{S} \log_2 \frac{S_i}{S} \quad (5)$$

where the split information is computed for attribute A in example set S.

A_n : Number of values of attribute A.
S_i : Number of examples where the i^{th} value of attribute A appears.
S : Total number of examples in the example set.

The gain ratio for each attribute is computed with eqn. (6).

$$GainRatio(S,A) = \frac{Gain(S,A)}{SplitInfo(S,A)} \quad (6)$$

Having sorted out the computed gain ratio values in descending order, the example set is re-arranged. Decision tree algorithms consider the attribute with the highest GainRatio as the root of the tree.

III. RULE GENERATION USING ENTROPY AND KNOWLEDGE GAIN

The proposed algorithm efficiently induces general rules from example sets (training data). We explain it using the example of Golf as given in Table 1[15].

Table 1. Golf Training Set

No	Weather	Temperature	Humidity	Wind	Decision
1	Sunny	High	High	Slight	Don't Play
2	Sunny	High	High	Strong	Don't Play
3	Cloudy	High	High	Slight	Play
4	Rainy	Normal	High	Slight	Play
5	Rainy	Low	Normal	Slight	Play
6	Rainy	Low	Normal	Strong	Don't Play
7	Cloudy	Low	Normal	Strong	Play
8	Sunny	Normal	High	Slight	Don't Play
9	Sunny	Low	Normal	Slight	Play
10	Rainy	Normal	Normal	Slight	Play
11	Sunny	Normal	Normal	Strong	Play
12	Cloudy	Normal	High	Strong	Play
13	Cloudy	High	Normal	Slight	Play
14	Rainy	Normal	High	Strong	Don't Play

The example set given in Table 1 consists of 14 examples, 4 attributes *(Weather, Temperature, Humidity, Wind)* and 2 classes *(Play, Don't Play)*. The attributes in the example and their values are given below:

Attribute	**Values**
Weather	Rainy, Sunny, Cloudy
Temperature	High, Medium, Low
Humidity	Normal, High
Wind	Slight, Strong

Calculate the entropy for each attribute and value. As it is seen, the attribute, W*eather*, has three values: *Rainy, Sunny and Cloudy*. The value, *Rainy*, of the attribute, *Weather*, appears in 5 examples three of which belong to the class, *Play,* and two of which belong to the class, *Don't Play.* Therefore, the entropy for {*Weather,Rainy*} can be computed as:

$$E_{Weather,Rainy} = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}$$

$$E_{Weather,Rainy} = 0.971\ bit$$

The value, *Sunny*, of the attribute, *Weather*, appears in 5 examples three of which belong to the class, *Don't Play,* and two of which belong to the class, *Play.* Therefore, the entropy for {*Weather, Sunny*} can be computed as:

$$E_{Weather,Sunny} = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}$$

$$E_{Weather,Sunny} = 0.971\ bit$$

The value, *Cloudy*, of the attributes, *Weather*, appears in 4 examples all of which belong to the class, *Play*. Therefore, the entropy for {*Weather, Cloudy*} can be computed as:

$$E_{Weather,Cloudy} = -\frac{4}{4}\log_2\frac{4}{4}$$

$$E_{Weather,Cloudy} = 0\ bit$$

From the above calculations, the entropy for the attributes, *Weather,* is computed as:

$$E_{Weather} = \frac{5}{14} x E_{Weather,Rainy} + \frac{5}{14} x E_{Weather,Sunny} + \frac{4}{14} x E_{Weather,Cloudy}$$

$$E_{Weather} = \frac{5}{14} x(0.971) + \frac{5}{14} x(0.971) + \frac{4}{14} x(0)$$

$$E_{Weather} = 0.694\ bit$$

Second attributes, *Temperature*, has three values: {*High, Low, Medium*}, third attributes, *Humidity*, has two values: {*High, Normal*}, and the fourth attributes, *Wind*, has two values: {*Slight, Strong*}. The entropies computed for each value of the attributes are presented in Table 2.

Table 2. Entropy values for the attributes and their values

Attribute	**Entropy (bit)**	**Value**	**Entropy (bit)**
Weather	0.694	Rainy	0.971
		Sunny	0.971
		Cloudy	0
Temperature	0.911	High	1
		Low	0.811
		Normal	0.918
Humidity	0.788	High	0.985
		Normal	0.592
Wind	0.892	Slight	0.811
		Strong	1

Info is computed as 0.940 for the example set. The *SplitInfo*, *Gain* and *GainRatio* for each characteristic are given in Table 3.

Table 3. Calculated values for characteristics

Att ibute	SplitInfo	Gain	GainRatio
Weather	1.577	0.264	0.156
Temperature	1.577	0.029	0.018
Humidity	1.000	0.152	0.152
Wind	0.985	0.048	0.049

Sort out the Information GainRatios calculated in descending order:

Weather (0.156) > **Humidity** (0.152) > **Wind** (0.049) > **Temperature** (0.018)

Considering the above sorting, the example set in Table 1 is rearranged according to the attributes, *Weather, Humidity*, *Wind,* and *Temperature,* and the results are given in Table 4.

Table 4. Re-arranged example set

No	Weather	Humidity	Wind	Temp.	Decision
1	Sunny	High	Slight	High	Don't Play
2	Sunny	High	Strong	High	Don't Play
3	Cloudy	High	Slight	High	Play
4	Rainy	High	Slight	Normal	Play
5	Rainy	Normal	Slight	Low	Play
6	Rainy	Normal	Strong	Low	Don't Play
7	Cloudy	Normal	Strong	Low	Play
8	Sunny	High	Slight	Normal	Don't Play
9	Sunny	Normal	Slight	Low	Play
10	Rainy	Normal	Slight	Normal	Play
11	Sunny	Normal	Strong	Normal	Play
12	Cloudy	High	Strong	Normal	Play
13	Cloudy	Normal	Slight	High	Play
14	Rainy	High	Strong	Normal	Don't Play

Having sorted the example set as in Table 4, Table 5 gives the set of rules obtained using REX-2.

Table 5. Rules generated with REX-2 algorithm for Golf Example

Rule	Rule Description
1	IF Weather=Cloudy THEN Decision=Play
2	IF Weather=Sunny AND Humidity=High THEN Decision=Don't Play
3	IF Weather=Rainy AND Wind=Slight THEN Decision=Play
4	IF Weather=Rainy AND Wind=Strong THEN Decision=Don't Play
5	IF Weather=Sunny AND Humidity=Normal THEN Decision=Play

The rules generated by REX-1 and C4.5 algorithms using the Golf Example are presented in Table 6. It is noted that both algorithms produced the same rules and the same number of rules just as REX-2 did.

Table 6. Rules generated by REX-1 and C4.5 algorithms (Golf problem)

Rule	Rule Description
1	IF Weather=Cloudy THEN Decision=Play
2	IF Humidity=High AND Weather=Sunny THEN Decision=Don't Play
3	IF Wind=Slight AND Weather=Rainy THEN Decision=Play
4	IF Wind=Strong AND Weather=Rainy THEN Decision=Don't Play
5	IF Humidity=Normal AND Weather=Sunny THEN Decision=Play

IV. CONCLUSION

In this section, REX-2 algorithm, which adapts the covering approach to generate rules using the entropy, is compared with other algorithms by using different example sets.

IV.1. Comparison of REX-2 with other algorithms, using the IRIS example set

The rules generated by the REX-1, REX-2, Rules-3, ID3 and Rules-3 Plus algorithms using the IRIS example set are given in Table 7a, 7b, 7c, 7d and 7e, respectively.

Table 7a. Rules generated by REX-1 (Iris example set)

Rule	Rule Description
1	IF 1.3≤PW<1.7 AND 3.95≤PL<4.93 THEN IRIS=Iris-versicolor
2	IF 0≤PW<0.51 THEN IRIS =Iris-setosa
3	IF 1.7≤PW<2.1 THEN IRIS =Iris-virginica
4	IF 0.9≤PW<1.3 THEN IRIS =Iris-versicolor
5	IF 2.1≤PW<2.5 THEN IRIS =Iris-virginica
6	IF 1≤PL<1.98 THEN IRIS=Iris-setosa
7	IF 4.93≤PL<5.91 AND 2.8≤SW<3.2 THEN IRIS =Iris-virginica
8	IF 1.3≤PW<1.7 AND 2.4≤SW<2.8 THEN IRIS =Iris-versicolor

Table 7b. Rules generated by REX-2 (IRIS data set)

Rule	Rule Description
1	IF 1≤PL<1.98 THEN IRIS=Iris-setosa
2	IF 1.7≤PW<2.1 THEN IRIS =Iris-virginica
3	IF 0.9≤PW<1.3 THEN IRIS =Iris-versicolor
4	IF 2.1≤PW<2.5 THEN IRIS =Iris-virginica
5	IF 3.95≤PL<4.93 AND 1.3≤PW<1.7 THEN IRIS =Iris-versicolor
6	IF 4.93≤PL<5.91 AND 2.8≤SW<3.2 THEN IRIS =Iris-virginica
7	IF 1.3≤PW<1.7 AND 2.4≤SW<2.8 THEN IRIS =Iris-versicolor

Table 7c. Rules generated by RULES-3 (Iris example set)

Rule	Rule Description
1	IF 6.56≤SL<7.13 AND 3.95≤PL<4.93 THEN IRIS=Iris-versicolor
2	IF 5.91≤PL<6.9 THEN IRIS =Iris-virginica
3	IF 0.9≤PW<1.3 THEN IRIS =Iris-versicolor
4	IF 4.93≤PL<5.91 THEN IRIS =Iris-virginica
5	IF 6≤SL<6.56 AND 3.95≤PL<4.93 THEN IRIS =Iris-versicolor
6	IF 4.86≤SL<5.43 AND 3.95≤PL<4.93 THEN IRIS =Iris-virginica
7	IF 1≤PL<1.98 THEN IRIS =Iris-setosa
8	IF 2.96≤PL<3.95 THEN IRIS =Iris-versicolor
9	IF 5.43≤SL<6 AND 1.3≤PW<1.7 THEN IRIS =Iris-versicolor
10	IF 5.43≤SL<6 AND 3.2≤SW<3.6 THEN IRIS =Iris-versicolor
11	IF 2.8≤SW<3.2 AND 1.7≤PW<2.1 THEN IRIS =Iris-virginica

Table 7d. Rules generated by ID3 (Iris example set)

Rule	Rule Description
1	IF 1≤PL<1.98 THEN IRIS =Iris-setosa
2	IF 4.93≤PL<5.91 THEN IRIS =Iris-virginica
3	IF 5.91≤PL<6.9 THEN IRIS =Iris-virginica
4	IF 3.95≤PL<4.93 AND 1.3≤PW<1.7 THEN IRIS =Iris-versicolor
5	IF 3.95≤PL<4.93 AND 0.9≤PW<1.3 THEN IRIS =Iris-versicolor
6	IF 3.95≤PL<4.93 AND 1.7≤PW<2.1 AND 2.4≤SW<2.8 THEN IRIS =Iris-virg.
7	IF 3.95≤PL<4.93 AND 1.7≤PW<2.1 AND 3.2≤SW<3.6 THEN IRIS =Iris-versi.
8	IF 2.96≤PL<3.95 THEN IRIS=Iris-versicolor

Tablo 7e. Rules generated by Rules-3 Plus (Iris example set)

Rule	Rule Description
1	IF 1≤PL<1.98 THEN IRIS=Iris-setosa
2	IF 3.95≤PL<4.93 AND 1.3≤PW<1.7 THEN IRIS =Iris-versicolor
3	IF 5.91≤PL<6.9 THEN IRIS =Iris-virginica
4	IF 4.93≤PL<5.91 THEN IRIS =Iris-virginica
5	IF 2.4≤SW<2.8 AND 1.7≤PW<2.1 THEN IRIS =Iris-virginica
6	IF 2.96≤PL<3.95 THEN IRIS =Iris-versicolor
7	IF 0.9≤PW<1.3 THEN IRIS =Iris-versiolor
8	IF 2.1≤PW<1.5 THEN IRIS =Iris-virginica
9	IF 3.2≤SW<3.6 AND 3.95≤PL<4.93 THEN IRIS =Iris-versicolor
10	IF 2.8≤SW<3.2 AND 1.7≤PW<2.1 THEN IRIS =Iris-virginica

It should be noted that while the number of rules and conditions generated by REX-1 was 8 and 11, respectively, REX-2 generated 7 rules and 10 conditions. On the other hand, ID3 produced 8 rules and 14 conditions. The algorithms ID3, Rules-3, Rules-3 Plus, Rules-4 and REX-1 were compared in terms of the number of rules and conditions generated. The results are given in Table 8.

Table 8. Number of rules and the mean of conditions per a rule (IRIS example set)

Algorithm	Number of Rules	Number of Conditions
RULES-3	11	17
RULES-3 PLUS	10	14
RULES-4	9	12
ID3	8	14
REX-1	8	11
REX-2	7	10

Compared with RULES family algorithms, REX-1 and REX-2 generated fewer rules and conditions. In addition, using the IRIS example set, the rate of efficiency in rule generation was 93.60%, 93.75% and 100% for Rules-4[9, 16], REX-1, and REX-2; respectively.

IV.2. Comparison of performance analyses of REX-2 with TDIDT and PRISM algorithms

In this section, we give some information on the test results of REX-2 with TDIDT and PRISM algorithms[17]. We use Monk1, Monk2, Monk3 and Soybean example sets and their testing data sets. The attributes of Monk data sets derived from real world problems are given in Table 10. Soybean data sets [14, 16] consist of 683 examples, 35 Attributes, and 19 classes. The results obtained with REX-2, TDIDT and PRISM algorithms using example sets of Monk1, Monk2, Monk3 and Soybean are presented in Table 9[17].

Table 9. Results obtained with the REX-2, TDIDT and PRISM algorithms

Example Set	TDIDT	PRISM	REX-2
Monk1	46	25	21
Monk2	87	73	83
Monk3	28	26	24
Soybean	109	107	98

Ps : INDUCED was used to obtain data from TDIDT and PRISM algorithms.

Mean number of conditions per a rule was obtained by the total number of conditions divided by the number of rules. It is seen that TDIDT algorithm generated the highest number of rules, compared with the other algorithms. The number of rules for TDIDT, PRISM and REX-2 are 270, 231 and 226, respectively.

Another preferred method of algorithm comparison is using the testing example sets. These sets are used to determine the rate of accuracy using the generated rules. That is, they test the results generated by an algorithm using an undefined example. Testing sets are obtained from the original testing sets. The rate of accuracy at the end of the tests is given in Table 10[18].

Table 10. Comparison of Rate of Accuracy using Testing Example Sets

Example Set	Number of Examples	TDIDT	PRISM	REX-2
Monk1	36	75.00%	77.78%	100.00%
Monk2	52	46.15%	53.85%	78.80%
Monk3	36	91.67%	83.33%	83.33%
Soybean	204	85.78%	84.80%	97.06%

Table 9 indicates that all of the algorithms, except TDIDT, produce almost the same results. However, the results obtained using the testing sets in Table 10 show that the introduced algorithm, REX-2, yields a very high rate of accuracy. One of the reasons for such a high rate may be the selection of attributes based on the entropy and information gain values.

V. DISCUSSION

Algorithms using the covering approach generate rules by performing only some search methods in the example sets. On the other hand, the algorithms benefiting from the divide-and-conquer approach generate a decision tree based on the entropy value and, then, induce rules out of the decision tree. Thanks to that feature, decision tree algorithms are able to generate a greater number of rules. Yet, some decision tree algorithms employ a technique called pruning which eliminates some unnecessary rules and thereby, resulting in a fewer number of generated rules [19]. As REX-2 algorithm uses both the covering approach and the entropy value and does not perform the pruning technique, it is capable of generating fewer rules and classifying any given example set with a higher rate of efficiency.

VI. REFERENCES

[1] Cios, K.J., Liu, N., Goodenday, L.S., "Generation of diagnostic rules via inductive machine learning", Kybernetes, vol. 22, no 5, 44-56, 1993.

[2] Quinlan, J.R., "Learning efficient classification procedures and their application to chess end games". In: Michalski; R.S., Carbonell, J.G. and Mitchell, T.M. (Eds), Machine Learning: An Artificial Intelligence Approach, Tioga Publishing Co, Palo Alto. CA, 463-482, 1983.

[3] Cheng, J., Fayyad, U.M., Irani, K.B., Qian, Z., "Improved decision trees: A generalized version of ID3", Proceedings of the Fifth International Conference on Machine Learning, Ann Arbor, Michigan, 100-106, 1988.

[4] Quinlan, J.R., "C4.5: Programs for Machine Learning", Morgan Kaufmann, San Mateo, CA, 1993.

[5] Michalski, R.S., "A theory and methodology of inductive learning", Machine Learning, Palo Alto, CA, 83-134, 1983.

[6] Kaufman, K.A., Michalski, R.S., "An Adjustable Rule Learner for Pattern Discovery Using the AQ Methodology", Journal of Intelligent Information Systems, 14, 199-216, 2000.

[7] Pham D. T., Aksoy M.S., "An algorithm for automatic rule induction", Artificial Intel. Eng., 8, 277-282,1993.

[8] Pham, D. T; Dimov, S.S., "An algorithm for incremental inductive learning". Proc. Instn. Mech. Engrs, vol. 211, part B, 239 – 249, 1997.

[9] Pham D. T, Dimov S.S., "The RULES-4 incremental inductive learning algorithm", Applications of Artificial Intelligence in Engineering XII, R.A. Adey G. Rzevski and R. Teti (Eds) Computational Mechanics Publications Southampton Boston, 163-166, 1997.

[10] Tolun, M. R., Abu-Soud S.M., "ILA:An inductive learning algorithm for rule extraction", Expert Systems With Applications, Vol: 14, 361-370, 1998.

[11] Akgöbek Ö., Aydin Y.S., Öztemel E., Aksoy M.S., "A new algorithm for automatic knowledge acquisition in inductive learning", Knowledge-Based Systems 19, 388-395, 2006.

[12] Akgöbek, Ö., "New algorithms for knowledge acquisition in inductive learning", Ph.D. Thesis, Sakarya University, Sakarya, Turkey, 2003.

[13] Klinkenberg, R., "Rule set quality measures for inductive learning algorithms", Master Thesis, University Of Missouri – Rolla, 1996.

[14] Piramuthu S., Sikora T. R., "Iterative feature construction for improving inductive learning algorithms", Expert Systems with Applications 36,3401-3406, 2009.

[15] Blake, C.L., Merz, C.J., "UCI Repository of Machine Learning Databases", [http://ftp.ics.uci.edu/pub/ml-repos/machine-learning-databases/]. Irvine, CA: University of California, Department of Information and Computer Science, 1998.

[16] Pham D. T., Dimov S. S., Salem Z., "Technique for selecting examples in inductive learning", ESIT 2000, Aachen, Germany, 2000.

[17] Bramer, M. A., "Inducer: A rule induction workbench for data mining", IFIP World Computer Congress Conference on Intelligent Information Processing, 2000, Beijing, Proceedings. Beijing: Publishing House of Electronics Industry, 499-506, 2000.

[18] Bramer M.A., "Automatic induction of classification rules from examples using N-Prism", Research and Development in Intelligent Systems XVI. Springer-Verlag, 99-121, 2000.

[19] Fournier, D., Cremilleux, B., "A quality index for decision tree pruning", Knowledge-Based System 15, 37-43, 2002.