Three Discretization Methods for Rule Induction

Jerzy W. Grzymala-Busse,[1,*] Jerzy Stefanowski[2]
[1] *Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, Kansas 66045*
[2] *Institute of Computing Sciences, Poznań University of Technology, 60-965 Poznań, Poland*

We discuss problems associated with induction of decision rules from data with numerical attributes. Real-life data frequently contain numerical attributes. Rule induction from numerical data requires an additional step called discretization. In this step numerical values are converted into intervals. Most existing discretization methods are used before rule induction, as a part of data preprocessing. Some methods discretize numerical attributes while learning decision rules. We compare the classification accuracy of a discretization method based on conditional entropy, applied before rule induction, with two newly proposed methods, incorporated directly into the rule induction algorithm LEM2, where discretization and rule induction are performed at the same time. In all three approaches the same system is used for classification of new, unseen data. As a result, we conclude that an error rate for all three methods does not show significant difference, however, rules induced by the two new methods are simpler and stronger.

1. INTRODUCTION

Classification systems may be created by means of *machine learning* techniques. In this approach an expert provides a set of learning examples. A learning algorithm is used to induce the classification knowledge from the learning set. In this paper, knowledge is expressed in the form of *decision rules*. A number of various algorithms have been already developed to induce rules.[1–3] The LEM2 algorithm,[4] based on *rough set theory*,[5,6] is one of such rule induction algorithms.

Rule induction techniques should not be applied initially to data bases containing *numerical* attributes, i.e., attributes with real number or integer

* Author to whom correspondence should be addressed; e-mail: Jerzy@eecs.ukans.edu
Contract grant sponsor: KBN Research.
Contract grant sponsor: CRIT 2-Esprit Project.
Contract grant number: 20288.

domains. Rules induced directly from numerical attributes are of poor quality (very short, weak and numerous). *Discretization techniques*, converting numerical attributes into discrete ones, are used to solve this problem. During discretization a number of cut-points are determined dividing the attribute domains into consecutive subintervals. Many discretization methods (see overviews in Ref. 7–9) can be applied as a *preprocessing step* before rule induction. Frequently, discretization is used for single numerical attributes applying Equal Interval Width, Equal Interval Frequency, or Minimal Entropy methods, see Ref. 7. Other approaches include Minimal Description Length,[4] discretization based on cluster analysis,[7] and other methods, e.g., Ref. 10. In general, no one discretization method is optimal for all situations.

In the past, LEM2 was extended to handle pre-discretized numerical attributes, see. Ref. 11. The extension of LEM2 attempted to extend intervals of attribute-value pairs in rules while inducing rules by LEM2. This version of LEM2 was used for discretization, as reported in Ref. 12. However, experimental results showed that it worked more efficiently for pre-discretized data.

We present a new approach to manipulate numerical data. Numerical attributes are not discretized before performing rule induction. Instead, a modified version of LEM2, called MODLEM, is applied directly to data with numerical attributes. Discretization and rule induction is performed simultaneously. Two versions of MODLEM, using different measures to evaluate elementary conditions: *class entropy* and *Laplacian accuracy*, are presented. We evaluated all of these approaches experimentally. Rule sets induced by both versions of MODLEM were compared with rule sets obtained in traditional way, i.e., discretization based on conditional entropy first and then rule induction by the 'pure' LEM2. For MODLEM and preliminary discretization plus LEM2 the same system was used for classifying testing data.

In the next section, the discretization technique based on conditional entropy is presented. Then, the basic version of LEM2 is cited. Section 4 describes the new algorithm MODLEM. Results of comparative experiments are given in Section 5. Discussion of these results and conclusions are presented in the final section.

2. DISCRETIZATION BASED ON ENTROPY

Most rule induction algorithms are restricted to discrete (symbolic) data. Before using these algorithms, a preliminary step must be performed called *discretization*. To be more specific, if a variable has numerical values from an interval $[a, b]$, then this interval is divided into subintervals $[a_1, a_2), [a_2, a_3), \ldots, [a_{m-1}, a_m]$, where $a_1 = a$, $a_{i-1} < a_i$, for $i = 2, 3, \ldots, m$, and $a_m = b$. These new subintervals may be replaced by new names or may be used as intervals, the only difference is syntactic. Thus the original, numerical variable is transformed into a discrete one. Numbers $a_2, a_3, \ldots, a_{m-1}$ are called *cut-points*.

In our experiments we used a discretization method in which first the best attribute was selected on the basis of minimal conditional entropy then the best cut-point was selected using the same criterion. Conditional entropy is defined

by

$$-\Sigma_{j=1}^{m} p(a_j) * \Sigma_{i=1}^{n} p(d_i \mid a_j) * \log p(d_i \mid a_j)$$

where $\{a_1, a_2, \ldots, a_m\}$ is the domain of an attribute a and $\{d_1, d_2, \ldots, d_n\}$ is the domain of a decision d. Thus, first the best attribute is selected taking into account information about all attributes and the decision. Then, for a selected attribute, the best cut-point is computed, again, taking into account the decision. The chosen cut-point divides the set of all examples into two subsets, S_1 and S_2. The remaining cut-points are selected by recursion on both S_1 and S_2. The algorithm terminates when the discretized data set becomes consistent. In our study, we use first the above discretization technique, then the LEM2 algorithm is applied to induce decision rules from transformed data.

3. RULE INDUCTION ALGORITHM LEM2

LERS[1] (Learning from Examples using Rough Sets) is a rule induction algorithm that uses rough set theory[5,6] to handle inconsistent data sets. Given an inconsistent data set, LERS computes the *lower approximation* and the *upper approximation* for each decision concept. LEM2 algorithm[1,3] of LERS induces a set of *certain rules* from the lower approximation, and a set of *possible rules* from the upper approximation. The procedure for inducing the rules is the same in both cases. The algorithm LEM2 uses the following ideas.

Let B be a non-empty lower or upper approximation of a concept represented by a decision-value pair (d, w). Let t be an attribute-value pair (a, v), and let T be a set of attribute-value pairs. Then the *block* of t, denoted $[t]$, is the set of examples for which attribute a has value v. Set B depends on a set T, of attribute-value pairs, if and only if:

$$\varnothing \neq [T] = \bigcap_{t \in T} [t] \subseteq B$$

A set T is a *minimal complex* of B if and only if B depends on T, and no proper subset T' of T exists such that B depends on T'. Let $\mathbf{T}$ be a non-empty collection of non-empty sets of attribute value pairs. Then $\mathbf{T}$ is a *local covering* of B if and only if the following conditions are satisfied:

1. Each member T of $\mathbf{T}$ is a minimal complex of B.
2. $\bigcap_{T \in \mathbf{T}} [T] = B$.
3. $\mathbf{T}$ is minimal, i.e., $\mathbf{T}$ has the smallest possible number of members.

The algorithm LEM2 is based on computing a single local covering for each approximation of the concept from the decision table. The user may select an option of LEM2 with or without taking into account attribute priorities. The other option differs from the one presented below in the selection of a pair $t \in T(G)$ in the inner loop WHILE. When LEM2 does not take attribute priorities into account, the first criterion is ignored. In our experiments all

attribute priorities were equal to each other. The procedure LEM2 with attribute priority is presented below.

Procedure LEM2
(**input:** a set B;
output: a single local covering **T** of set B);
begin
  $G := B$;
  T $:= \emptyset$;
  while $G \neq \emptyset$ **do**
    begin
    $T := \emptyset$;
    $T(G) := \{t \mid [t] \cap G \neq \emptyset\}$;
    while $T = \emptyset$ **or not** $([T] \subseteq B)$ **do**
      begin
        select a pair $t \in T(G)$ with the highest attribute priority, if a tie occurs, select a pair $t \in T(G)$ such that $|[t] \cap G|$ is maximum; if another tie occurs, select a pair $t \in T(G)$ with the smallest cardinality of $[t]$; if a further tie occurs, select first pair;
        $T := T \cup \{t\}$;
        $G := [t] \cap G$;
        $T(G) := \{t \mid [t] \cap G \neq \emptyset\}$;
        $T(G) := T(G) - T$;
      end; {while}
    for each t in T **do**
      if $[T - \{t\}] \subseteq B$ **then** $T := T - \{t\}$;
    T $:=$ **T** $\cup \{T\}$;
    $G := B - \bigcup_{T \in \mathbf{T}} [T]$;
  end {while};
  for each $T \in$ **T** **do**
    if $\bigcup_{S \in \mathbf{T} - \{T\}} [S] = B$ **then** **T** $:=$ **T** $- \{T\}$;
end {procedure}.

4. MODLEM ALGORITHM

Preliminary discretization of numerical attributes is not required by MODLEM. The algorithm MODLEM (first version proposed in Ref. 13) handles these attributes during rule induction, when elementary conditions of a rule are created. A similar idea of processing numerical data is also considered in other learning systems, e.g., C4.5 performs discretization and tree induction at the same time.[14]

In general, MODLEM algorithm is analogous to LEM2. MODLEM also uses rough set theory to handle inconsistent examples and computes a single local covering for each approximation of the concept. However, an elementary condition t is defined in a different way. In the original version of LEM2 elementary conditions are pairs (a, v), while in MODLEM t for numerical attributes are presented in the form of either $(a < v)$ or $(a \geq v)$, where v is a kind of threshold on the attribute domain. For a given attribute the choice of the form, $t = (a < v)$ or $t = (a \geq v)$, depends on which block $[t]$ covers more training examples from B, an approximation of the concept. A minimal complex

T is a conjunction of such conditions. If the same attribute is chosen twice while building a single rule, one may also obtain the condition $(a = [v_1, v_2))$ that results from an intersection of two conditions $(a < v_2)$ and $(a \geq v_1)$ such that $v_1 < v_2$. Moreover, algorithm MODLEM looks for such modified conditions in a different way, as presented in procedure *Find_best_condition*.

First, values of a numerical attribute a for all examples are sorted in increasing order. The candidates for cut-points as mid-points between successive values of points in the sorted order. We consider only mid-points between values characterizing examples belonging to different decision classes, following a result from Ref. 15. This property reduces the number of candidates for cut-points. Any cut-point is evaluated using minimal class entropy technique[4] and the best cut-point is found. Having the best cut-point we choose a condition $(a < v)$ or $(a \geq v)$ that covers more positive examples from the set B. The procedure is repeated for all other attributes. As a result, the best condition is found. If it is not sufficient for completing the rule, the strategy is repeated until the complete rule is induced.

The general schema of the MODLEM algorithm is given below. It is iteratively repeated for each set B being an approximation of a decision class.

```
Procedure MODLEM
(input: a set B; a set C of attributes;
output: a single local covering T of set B);
begin
  G := B;
  T := ∅;
  while G ≠ ∅ do
  begin
    T := ∅; {candidate for condition part of the rule}
    S := U; {set of objects currently covered by T}
    while (T = ∅) or not ([T] ⊆ B) do
    begin
      t := ∅; {candidate for elementary condition}
      t_eval := ∞; {evaluation measure for t}
      for each attribute a ∈ C do
      begin
        Find_best_condition (a, S, new_t, eval_new_t); (look for best condition
        for a}
        if eval_new_t < t_eval then {evaluate if new condition is better than
        previous}
          begin
            t := new_t;
            t_eval := new_eval;
          end;
      end; {for}
      T := T ∪ {t};
      S := S ∩ [t];
    end; {while not ([T] ⊆ B)}
    for each elementary condition t ∈ T do
      if [T − {t}] ⊆ B then T := T − {t};
    T := T ∪ {T};
    G := B − ∪T∈T [T];
```

```
    end; {while G ≠ ∅}
    for each T ∈ T do
      if ⋃_{S ∈ T−{T}} [S] = B then T := T − {T};
end {procedure}.

procedure Find_best_condition
(input attribute a;
       set S of objects;
output current best condition best_t
       evaluation measure eval_best__t of the best condition);
begin
  best_t := ∅;
  eval_best_t := ∞;
  sort H; {create a list of values for attribute a and objects from S sorted
           according to increasing values; H(i) is the ith value in the list}
  for i := 1 to length (H) − 1 do
  begin
    v := (H(i) + H(i + 1))/2; {consider only mid-points between values
                               characterizing examples belonging to
                               different classes}
    S_1 := {x ∈ S | a(x) < v}; {a(x) is the value of attribute a for object x}
    S_2 := {x ∈ S | a(x) ≥ v};
    eval_t := (|S_1|/|S_1 ∪ S_2|) * Ent(S_1) + (|S_2|/|S_1 ∪ S_2|) * Ent(S_2);
      {Ent—entropy calculated for S_j}
    if eval_t < eval_best then
    begin
      if |G ∩ S_2| ≥ |G ∩ S_1| then
        best_t := (c ≥ v) else best_t := (c < v);
        eval_best_t := eval_t
      end {if}
    end
end {procedure}.
```

In the above presentation we use the minimal class entropy measure to evaluate conditions. Optionally, the user can choose another measure called Laplacian accuracy.[16] This additional option is based on the fact that for some data, entropy has a tendency to induce 'pure' rules covering small number of examples, while the user may want to discover rules covering more examples. Laplacian accuracy is defined as $(n_c + 1)/n_{\text{tot}} + k)$, where k is the number of classes in the data set, n_c is the number of examples in the predicted class B covered by the rule (or conditions from T), and n_{tot} is the total number of examples covered by the rule. To the contrary to entropy, the higher values of Laplacian accuracy are more preferred to lower. The Laplacian accuracy was used previously in the modified CN2[7] to avoid entropy bias, with good results. In this paper the MODLEM algorithm was used in two versions, MODLEM-Entropy and MODLEM-Laplace, depending on the used evaluation measures.

The above description of the MODLEM algorithm was presented only for numerical attributes. However, MODLEM can also handle the combination of nominal and numerical attributes. For nominal attributes elementary conditions are presented in the usual form $(a = v)$. In function *Find_best_condition* each

value v of the nominal attribute is considered, two subsets: $S_1 = \{x \in S \mid a(x) = v\}$, $S_2 = S - S_1$ are calculated and used for evaluation.

5. CLASSIFICATION OF TESTING EXAMPLES

For both rule induction systems we use the same LERS classification system.[17] It is a modified bucket brigade algorithm.[18,19] The decision to which concept an example belongs to is made on the basis of three factors: strength, specificity, and support. They are defined as follows: *Strength* is the total number of examples correctly classified by the rule during training. *Specificity* is the total number of attribute-value pairs on the left-hand side of the rule. The matching rules with a larger number of attribute-value pairs are considered more specific. The third factor, *support*, is defined as the sum of scores of all matching rules from the concept. The concept C for which the support, i.e., the following expression:

$$\sum_{R \in Rul} Strength\_factor(R) * Specify\_factor(R)$$

is the largest is the winner and the example is classified as being a member of C, where *Rul* denotes the set of all matching rules R describing C.

If complete matching is impossible, all partially matching rules are identified. These are rules with at least one attribute-value pair matching the corresponding attribute-value pair of an example. For any partially matching rule R, the additional factor, called *Matching factor*(R), defined as a ratio of matching conditions to all conditions in the rule is computed. In partial matching, the concept C for which the following expression

$$\sum_{R \in Rul'} Matching\_factor(R) * Strength\_factor(R) * Specify\_factor(R)$$

is the largest is the winner and the example is classified to C, where Rul' is the set of all partially matching rules R describing C.

6. EXPERIMENTS

In order to compare the performance of all three approaches to handle numerical data we performed experiments on several real-life data sets. Table I gives the summary of data statistics. Most of these data sets are well-known data previously used for testing learning systems. They are available at the University of California at Irvine repository. The other data sets are taken from rough set applications. The next three tables, Tables II, III, IV present results of our experiments. Rule sets for all three methods are characterized in terms of their cardinality, the total number of conditions in a rule set (the lower the better) and average strength of the rule, i.e., average number of learning examples covered by a single rule (the higher the better). Also, the accuracy (the higher the better), computed by ten-fold cross validation, is listed.

Table I. Data sets.

Data Set	Number of Cases	Number of Attributes	Number of Concepts
Bank	66	5	2
Bupa	345	6	2
Buses	76	8	2
Glass	214	9	6
HSV	122	11	2
Iris	150	4	3
Pima	768	8	2
Bricks	216	10	2
Segmentation	210	19	7
German (numeric)	1000	24	2

Table II. MODLEM-Laplace.

Data Set	Number of Rules	Number of Conditions	Average Strength	Accuracy [%]
Bank	6	7	22	94
Bupa	101	228	5.6	68
Buses	5	5	35	97
Glass	80	139	3.6	58
HSV	54	96	2.9	63
Iris	12	24	19.8	91
Pima	188	400	9.1	74
Bricks	22	38	17.7	91
Segmentation	47	80	5.3	72
German (numeric)	253	774	8.4	73

Table III. MODLEM-Entropy.

Data Set	Number of Rules	Number of Conditions	Average Strength	Accuracy [%]
Bank	3	16	28	94
Bupa	79	219	6.5	66
Buses	4	5	35.3	97
Glass	43	111	6.8	72
HSV	35	92	4.7	57
Iris	10	20	20.3	94
Pima	125	426	13	74
Bricks	16	33	25.2	91
Segmentation	22	45	10.5	85
German (numeric)	182	751	9.3	73

Table IV. Discretization based on entropy and LEM2.

Data Set	Number of Rules	Number of Conditions	Average Strength	Accuracy [%]
Bank	10	13	7.5	97
Bupa	169	501	2.4	66
Buses	3	4	33.3	99
Glass	111	262	2.2	67
HSV	62	206	2.3	56
Iris	14	33	12.6	97
Pima	252	895	4.6	74
Bricks	25	61	9.2	92
Segmentation	108	322	2	64
German (numeric)	290	1226	5.5	74

7. CONCLUSIONS

Results of our experiments were pairwise compared using a nonparametric test: the two-tailed Wilcoxon matched-pair signed rank test for the 5% significance level.

- Total number of rules: the best method is MODLEM-Laplace, then MODLEM-Entropy, the worst method is first discretization based on entropy then LEM2.
- Total number of conditions in a rule set: MODLEM-Laplace and MODLEM-Entropy are of the same quality; the remaining method, first discretization based on entropy then LEM2, is worse.
- Average rule strength; the best method is MODLEM Entropy, then MODLEM Laplace, the worst method is first discretization based on entropy then LEM2.
- Accuracy: the differences for all three methods are statistically nonsignificant.

The MODLEM algorithm discretizes numerical attributes during rule induction. Thus the search space for MODLEM is bigger than the search space for original LEM2, which generates rules from already discretized attributes. Consequently, rule sets induced by MODLEM are much simpler and stronger. Some users may appreciate the fact that MODLEM does not require preprocessing of data (in the sense that the discretization is not visible as an independent part of the algorithm). In addition, the quality of rule sets measured by accuracy is indistinguishable for all three methods.

References

1. Grzymala-Busse JW. LERS—A system for learning from examples based on rough sets. In: Slowinski R, editor. Intelligent decision support handbook of applications and advances of the rough sets theory. Kluwer Academic Publishers; 1992. p 3–18.
2. Michalski RS, Bratko I, Kubat M, editors. Machine learning and data mining. New York: John Wiley & Sons; 1998.
3. Stefanowski J. On rough set based approaches to induction of decision rules. In: Polkowski L, Skowron A, editors. Rough sets in data mining and knowledge discovery, Vol. 1. Physica-Verlag; 1998. p 500–529.

4. Fayyad UM, Irani KB. Multi-interval discretization of continuous-valued attributes for classification learning. In: Proc of the 13th Int Conf on Machine Learning; 1993. p 1022–1027.
5. Pawlak Z. Rough sets. Int J Computer and Information Sci 1982;11:341–356.
6. Pawlak Z. Rough sets—Theoretical aspects of reasoning about data. Kluwer Academic Publishers; 1991.
7. Chmielewski M, Grzymala-Busse JW. Global discretization of continuous attributes as preprocessing for matching learning. Internat J Approx Reason 1996;15:319–331.
8. Dougherty J, Kohavi R, Sahami M. Supervised and unsupervised discretizations of continuous features. Proc of the 12th Int Conf on Maching Learning; 1995. p 194–202.
9. Susmaga R. Analyzing discretization of continuous attributes given a monotonic discrimination function. Intelligent data analysis, Vol. 1, No. 3; 1997. (Elsevier on line journal http://www-east.elsevie.com/ida).
10. Nguyen SH, Skowron A. A quantization of real value attributes: Rough set and Boolean reasoning approach. Proc of the 2nd Joint Annual Conf on Information Science, Wrightsville Beach, NC, Sept. 28–Oct. 1, 1995. p 34–37.
11. Grzymala-Busse JW, Lakshmanan A. LEM2 with interval extension: An induction algorithm for numerical attributes. In: Tsumoto S, editor. Proc of the Fourth International Workshop on Rough Sets, Fuzzy Sets, and Machine Discovery. Tokyo, Nov. 6–8, 1996. p 67–73.
12. Grzymala-Busse JW, Stefanowski J. Discretization of numerical attributes by direct use of the LEM2 induction algorithm with interval extension. Proc of 6th Symp Intelligent Information Systems, Zakopane, Poland: IPI Pan Press; 1997. p 159–168.
13. Stefanowski J. Rough set based rule induction techniques for classification problems. In: Proc 6th European Congress on Intelligent Techniques and Soft Computing. Aachen; Sept. 7–10, 1988, Vol. 1. p 109–113.
14. Quinlan JR. C4.5: Programs for machine learning. Morgan Kaufmann Publishers; 1993.
15. Fayyad UM, Irani KB. On handling of continuous-valued attributes in decision tree generation. Maching Learning Journal 1992;8:87–102.
16. Clark P, Boswell R. Rule induction with CN2: some recent improvements. In: Kodratoff Y, editor. Proc of 5th European Working Session on Learning–ESWL 91, Porto, Portugal. Springer Verlag; 1991. p 151–163.
17. Grzymala-Busse JW. Managing uncertainty in machine learning from examples. In: Dabrowski M, Michalski R, Ras Z, editors. Proc of 3rd Int Symp on Intelligent Systems. Wigry, Poland. Warszawa: IPI PAN Press; 1994. p 70–84.
18. Booker LB, Goldberg DE, Holland JF. Classifier systems and genetic algorithms. In: Carbonell JG, editor. Machine learning, paradigms and methods. Cambridge, MA: The MIT Press; 1990. p 235–282.
19. Holland JH, Holyoak KJ, Nisbett, RE. Induction. Processes of inference, learning, and discovery. Cambridge, MA: The MIT Press; 1986.