Interestingness in Attribute-Oriented Induction (AOI): Multiple-Level Rule Generation

Maybin K. Muyeba and John A. Keane

Department of Computation, UMIST, Manchester, M60 1QD, UK
muyeba,jak@co.umist.ac.uk

Abstract. Attribute-Oriented Induction (*AOI*) is a data mining technique that produces simplified descriptive patterns. Classical *AOI* uses a predictive strategy to determine distinct values of an attribute but generalises attributes indiscriminately i.e. the value 'ANY' is replaced like any other value without restrictions. *AOI* only produces interesting rules by using interior concepts of attribute hierarchies. The COMPARE algorithm that integrates *predictive* and *lookahead* methods and of order complexity *O (np)*, where *n* and *p* are input and generalised tuples respectively, is introduced. The latter method determines distinct values of attribute clusters and greatest number of attribute values with a '*common parent*' (except parent 'ANY'). When generating rules, a rough set approach to eliminate redundant attributes is used leading to more interesting multiple-level rules with fewer 'ANY' values than classical *AOI*.

1 Introduction

Attribute-Oriented Induction (*AOI*) [1] is a data mining technique that produces descriptive patterns by generalisation. Each attribute has a predefined *concept hierarchy* often referred to as *domain knowledge*. Concept hierarchy values (specifically *interior concepts* as shown in figure 1) are used repeatedly to replace low-level attribute values and generate a *prime relation*. Attributes that do not have concept hierarchies or posses a large number of distinct values (e.g. *keys* to relations) are dropped except in the *key-preserving AOI* [2]. The number of allowed attributes and tuples in the final table is controlled by *attribute* and *rule thresholds* T_a and T_r respectively. The end result is a *final generalised relation* or *rule table* of descriptive patterns. Rules produced in this way can use preserved *keys* to perform efficient data queries. For example, a rule describing American cars with low mileage can be used to query specific properties like engine size, type of steering etc from the database. Generally, *leaf concepts* and value 'ANY' are not interesting to the user as they represent known facts. The interesting values however lie in interior concepts of an attribute's concept hierarchy according to [3]. Classical *AOI* does not consider interestingness of the generated rules as the value 'ANY' or 'don't-care' is used without restrictions.

D.A. Zighed, J. Komorowski, and J. Zytkow (Eds.): PKDD 2000, LNAI 1910, pp. 542-549, 2000.

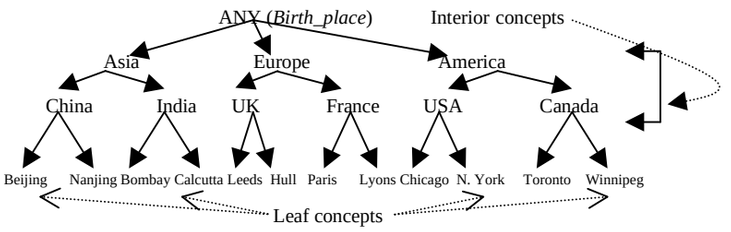

Fig. 1. Concept hierarchy for attribute '*Birth_place*'

We propose an approach to obtain interesting multiple-level rules (rules with high and low-level concepts) and regulate the degree of the rule table by generalising clusters of attribute values with a 'similar' *next* high-level concept (a '*common parent'*). A rough set approach is used to eliminate irrelevant attributes for rule generation [4].

The rest of the paper is organised as follows: section 2 presents related work; section 3 introduces concepts and terminologies; section 4 introduces the algorithm and analysis; section 5 shows an application of the approach using public domain data; section 6 presents conclusions and future work.

2 Related Work

Interestingness measures are either objective or subjective [5]. Objective approaches to foster interestingness in *AOI* can be categorised as *lookahead* and *predictive* [6]. The former are based on selecting an attribute for generalisation using heuristic measures based on the attribute's concept hierarchy structure [8]. Predictive strategies use thresholds [4] or predetermine the interestingness potential of interior concepts of an attribute's hierarchy. The rough set approach is used to determine dependencies between attributes. In, classification, information gain is used to choose an attribute to use as a root of the classification tree [9]. This results in correctly classified trees. Our approach uses distinct attribute values to choose an attribute and generalise clusters of values for the attribute to produce multiple-level rules [7].

3 Terminology and Definitions

Suppose a database has *m* attributes and *n* tuples. Assume that a concept hierarchy H_i is defined for each attribute A_i, i=2..*m* and a prime relation of *p* tuples exists.

Definition 3.1 A Concept hierarchy H_i of A_i is a *poset* (partially ordered set) $(H,\prec)$ where H_i is a finite set of concepts and $\prec$ is a partial order on H_i.

Definition 3.2. An attribute A_i is *generalisable* if there exists a concept hierarchy for the attribute (i.e. there are higher level concepts which subsume the given attribute values). Otherwise it is *nongeneralisable*.

Definition 3.3 A concept *y* of attribute A_i is the *nearest ancestor* of concept *x* if *x*, $y \in H_i$ with $x \prec y$, $x \neq y$, and there is no other concept $z \in H_i$ such that $x \prec z$ and $z \prec y$.
Definition 3.4 A concept y of attribute A_i is an *interior concept* of H_i if *y* is a non-leaf concept and y ≠ 'ANY'.
Definition 3.5 Two concept values a_r, $a_s \in A_i$ have a '*common parent*' $p \in H_i$ if *p* is the nearest ancestor of a_r and a_s.
Definition 3.6 A *key* $\mathbf{t}_p(A_i)$ of a tuple $\mathbf{t}_p$ is the first numeric attribute value, i=1, *p*<=*n*.
Definition 3.7 A *characteristic rule* [7] is a rule whose left hand side is the query condition of the target data and the right hand side is a conjunction of generalised (attribute, value) pairs.
Definition 3.8 Two tuples $\mathbf{t}_p$ and $\mathbf{t}_q$ are *equivalent* if $\mathbf{t}_p(A_i) = \mathbf{t}_q(A_i)$, i≠1, i= 2,..,m.
Definition 3.9 A *merge* of two tuples $\mathbf{t}_p$ and $\mathbf{t}_q$ occurs by determining their equivalence, incrementing the count of total equivalent tuples by one and deleting one tuple.

The first task the algorithm in the next section deals with is maintaining interestingness through the *AOI* generalisation process. After reducing the complexity of the data by *AOI*, the second part which is rule generation, introduces a rough set approach. Rules are then represented as decision rules. When decision rules are generated, rules redundancies and rule inconsistencies may arise. Our proposed algorithm inherently solves this problem by the nature of the generalisation process (see application example). Definitions and detail theory of decision rules, rule redundancy and inconsistencies are given in [4].

4 Common Parent Rule Algorithm (COMPARE)

The COMmon PArent RulE (COMPARE) algorithm performs *AOI* primitives as follows:

1. Retrieve the input and generalise each tuple during input so that no tuple has leaf concepts at rule generation stage except for attributes with a concept hierarchy depth[1] of 1. This also reduces the number of tuples in memory to a small relation.
2. An a*ttribute threshold* T_a and an evaluation function[2] for choosing the next attribute for generalisation are used. For each chosen attribute, all attribute values with a '*common parent*' (excluding 'ANY') are grouped together and counted. The group with the highest count is generalised until thresholds are reached.
3. *Rule generation* only considers reduction of number of tuples by further generalisation of selected attribute values and controlled by a rule threshold Tr. Significant values[3] are used to eliminate less significant attributes by starting with tuples differing in one attribute value. This approach, like in [4], ensures that we remain with non-redundant attributes for rule generation.

[1] Depth of root node is zero and any other node is 1 more than the depth of its parent
[2] The function d_{weight}/t_{weight} is used in [4]
[3] The chi-square statistic X^2

4.1 Analysis

When an attribute A_i is chosen for generalisation using the evaluation function, attribute values with a '*common parent*' in the concept hierarchy H_i are identified and grouped together. When *k* similar attribute values are encountered (i.e. they have a '*common parent*'), *k* keys are inserted in a $(p+c)x(p+c)$ array and a count of the flags (denoting each flag as 'z') representing 'a similar value encounter' is recorded, where *c* allows for all 'z' flags and is small. The number of *distinct* values in each group is evaluated taking all 'z' flags into account and the group with the highest value is a candidate for further generalisation. This process is repeated with other groups until the evaluation function is satisfied using threshold $\boldsymbol{T}_a$.

INPUT Task relevant data of *N* tuples, *m* attributes A_i, i=1,..,m, thresholds T_a, T_r
OUTPUT A set of interesting multiple-level rules
Step 1. Input tuples, generalise each tuple
Merge equivalent tuples and store in a prime table
Step 2. For (*i* = 1 to *m*)
Cluster values of $\mathbf{A}_i$ with '*common parents*'
Generalise cluster with highest distinct values
While (count distinct $\mathbf{A}_i > T_a$)
Generalise cluster with next greatest distinct values
Evaluate count distinct of $\mathbf{A}_i$
Step 3. Compute significant values (SIG_i) for each attribute $\mathbf{A}_i$
Step 4. While (threshold $\boldsymbol{T}_r$ not reached)
For each $\mathbf{A}_i$, starting with lowest SIG_i of **A,** $i \in (2,..,m)$
If two tuples differ in only ONE attribute $\mathbf{A}_i$
Replace attribute values with 'ANY'
merge equivalent tuples and evaluate threshold
Step 5. Output tuples in final table

Fig. 2. The COMPARE algorithm

The complexity of the algorithm in figure 2 is evaluated as follows: To input *n* tuples to a generalised prime relation is $O(np)$ [2] and finding distinct values for each attribute A_i is $O(m)$, $0<i\leq m$, for *m* attributes and *p* generalised values [6]. For *h* clusters of A_i where $C_{ji}=\{C_{1i},..,C_{hi}\}$, $|C_{ji}| < p/m$, $\forall\ j=1,..,h$, i.e. the size of clusters C_{ji} are not too different for each attribute. Inserting 'similar value encounter' in the $(p+c)x(p+c)$ array is less than $O(p^2)$ and negligible. Assuming the cardinality of each corresponding cluster is the same for each attribute (i.e. $|C_{11}|=|C_{12}|=|C_{13}|$ etc except for concept distributions), then the number of tuples is expressed as

$$\sum_{i=1}^{m}\sum_{j=1}^{h}(|C_{ji}|) = p.$$

After choosing attribute A_i, to find common parent for attribute values in j^{th} cluster C_{ji} for *k* concepts and generalising, the complexity is $O(k|C_{ji}|)$. For *h* clusters of *m*

attributes, the complexity is O ($kmh|C_{ji}|$) in the worst case, where each C_{ij} is generalised for small k and m. Since $|C_{ji}|$<p/m, $hm|C_{ji}|=p$, $\forall_{j,i}$ and therefore the order complexity is O (kp) or just O (p).

The complexity of the algorithm by using attribute thresholds is O (np) + O (m) + O (p) = O (np). To merge the prime relation using least significant values for m attributes and inserting in a final table of q tuples, the complexity is O (pqm^2). Thus, if we say $p=n_i$, and $q=n_g$ as in [6], we have O ($m^2n_in_o$). Thus we have order complexity O (np)+O (p)+ O (m^2pq). For small p and q in large databases, the complexity is reduced to O (np).

4.2 Example Application

Table 1. Car Information

Cno	Model	Fuel	Disp	Weight	Cyl	Power	Turbo	Comp	Tran	Mileage
1	Ford Escort	EFI	Medium	876	6	High	Yes	High	auto	Medium
2	Dodge Shadow	EFI	Medium	1100	6	High	No	Medium	manu	Medium
3	Ford Festiva	EFI	Medium	1589	6	High	No	High	manu	Medium
4	Chevrolet Corvette	EFI	Medium	987	6	High	No	Medium	manu	Medium
5	Dodge Stealth	EFI	Medium	1096	6	High	No	High	manu	Medium
6	Ford Probe	EFI	Medium	867	6	High	no	Medium	manu	Medium
7	Ford Mustang	EFI	Medium	1197	6	High	no	High	manu	Medium
8	Dodge Daytona	EFI	Medium	798	6	High	yes	High	manu	High
9	Chrysler Le Baron	EFI	Medium	1056	4	Medium	no	Medium	manu	Medium
10	Dodge Sprite	EFI	Medium	1557	6	High	no	Medium	manu	Low
11	Honda Civic	2-BBL	Small	786	4	Low	no	High	manu	High
12	Ford Escort	2-BBL	Small	1098	4	Low	no	High	manu	Medium
13	Ford Tempo	2-BBL	Small	1187	4	Medium	no	High	auto	Medium
14	Toyota Corolla	EFI	Small	1023	4	Low	no	High	manu	High
15	Mazda 323	EFI	Medium	698	4	Medium	no	Medium	manu	High
16	Dodge Daytona	EFI	Medium	1123	4	Medium	no	Medium	manu	Medium
17	Honda Prelude	EFI	Small	1094	4	High	yes	High	manu	High
18	Toyota Paseo	2-BBL	Small	1023	4	Low	no	Medium	manu	High
19	Chevrolet Corsica	EFI	Medium	980	4	High	yes	Medium	manu	Medium
20	Chevrolet Beretta	EFI	Medium	1600	6	High	no	Medium	auto	Low
21	Chevrolet Cavalier	EFI	Medium	1002	6	High	no	Medium	auto	Medium
22	Chrysler Le Baron	EFI	Medium	1098	4	High	no	Medium	auto	Medium
23	Mazda 626	EFI	Small	1039	4	Medium	no	High	manu	High
24	Chevrolet Corsica	EFI	Small	980	4	Medium	no	High	manu	High
25	Chevrolet Lumina	EFI	Small	1000	4	Medium	no	High	manu	High

{Honda_Civic,..,Honda_Accord} ⊂ Honda
{Toyota_Tercel,…,Toyota_Camry} ⊂ Toyota
{Mazda 323,…, Mazda_626} ⊂ Mazda {Ford,…,Chevrolet} ⊂ USA{Car}
{Ford_Escort,…,Ford_Taurus} ⊂ Ford {Honda,…,Mazda} ⊂ Japan {Car}
{Chev_Covertte,…,Chev_Corsica} ⊂ Chevrolet {Japan (Car),…, USA(Car)} ⊂ Any (Model)
{Dodge_stealth,…,Dodge_Dynasty} ⊂ Dodge {Light, Medium,Heavy} ⊂ Any (Weight)
{0..800} ⊂ Light
{801..1200} ⊂ Medium
{1201..1600} ⊂ Heavy

Fig. 3. A concept hierarchy for Table 1

Consider a collection of car information in Table 1 and its associated concept hierarchy shown in figure 3. From the tables, the attributes '*Fuel*', '*Disp*', '*Cyl*', '*Power*', '*Turbo*', '*Comp*', '*Tran*' and '*Mileage*' have at most 3 distinct values and their concept hierarchies are not given. If we choose to investigate characteristics of

cars that determine *mileage* from the given data, then *mileage* will be a *decision attribute* or target class and all other attributes will be a conjunction of generalised attribute value pairs. This eliminates the attributes '*Fuel*', '*Disp*', '*Cyl*' and '*Turbo*' by analogy and expert knowledge. Another attribute '*Cno*' for car number to act as a *key* for each tuple as defined earlier is introduced.

4.3 Results

A Generalised Car Information System is first obtained as shown in table 2.

Table 2. A table generated from COMPARE algorithm (18 tuples)

Cno	Model	Weight	Power	Comp	Tran	Mileage	Count
1	USA_CAR	Medium	High	High	Auto	Medium	1
2	USA_CAR	Medium	High	Medium	Manu	Medium	4
3	USA_CAR	Heavy	High	High	Manu	Medium	1
5	USA_CAR	Medium	High	High	Manu	Medium	2
8	USA_CAR	Light	High	High	Manu	High	1
9	USA_CAR	Medium	Medium	Medium	Manu	Medium	2
10	USA_CAR	Heavy	High	Medium	Manu	Low	1
11	Honda	Light	Low	High	Manu	High	1
12	USA_CAR	Medium	Low	High	Manu	Medium	1
13	USA_CAR	Medium	Medium	High	Auto	Medium	1
14	Toyota	Medium	Low	High	Manu	High	1
15	Mazda	Light	Medium	Medium	Manu	High	1
17	Honda	Medium	High	High	Manu	High	1
18	Toyota	Medium	Low	Medium	Manu	High	1
20	USA_CAR	Heavy	High	Medium	Auto	Low	1
21	USA_CAR	Medium	High	Medium	Auto	Medium	2
23	Mazda	Medium	Medium	High	Manu	High	1
24	USA_CAR	Medium	Medium	High	Manu	High	2

Table 3. Significant values for attributes

Attribute	χ^2	Attribute	χ^2
Weight	17.54	Tran	4.53
Model	12.86	Comp	3.84
Disp	7.08	Fuel	0.63
Cyl	5.94	Turbo	0.63
Power	5.68		

Attribute and rule thresholds of 3 and 9 were used to compare the results obtained in [6]. Significant values for attributes in table 2 are shown in table 3. Applying the COMPARE algorithm generates table 4. The values 'ANY' are represented by '-'. After attribute thresholds are satisfied, rule thresholds need to be satisfied by further processing of table 4. Table 5 shows the generated table using the user-defined rule threshold $\boldsymbol{T}_r$=9.

Table 4. A table after merging tuples differing by one attribute (12 tuples)

Cno	Model	Weight	Power	Comp	Tran	Mileage	Count
1	USA_CAR	Medium	High	-	-	Medium	8
3	USA_CAR	Heavy	High	High	Manu	Medium	1
6	USA_CAR	Meium	Medium	-	Manu	Medium	4
8	USA_CAR	-	Medium	High	Manu	High	3
10	USA_CAR	Heavy	Low	Medium	Manu	Low	1
11	Honda	Light	Low	High	Manu	High	1
13	USA_CAR	Medium	Medium	Hgh	Auto	Medium	1
14	Toyota	Medium	Low	-	Manu	High	2
15	Mazda	Light	Medium	Medium	Manu	High	1
17	Honda	Medium	High	High	Manu	High	1
20	USA_CAR	Heavy	High	Medium	Auto	Low	1
23	Mazda	Medium	Medium	High	Manu	High	1

Table 5. Final Table (8 tuples)

Cno	Model	Weight	Power	Comp	Tran	Mileage	Count
1	USA_CAR	Medium	High	-	-	Medium	9
3	USA_CAR	-	-	High	Manu	Medium	2
8	-	Light	-	High	Manu	High	2
9	USA_CAR	Medium	Medium	-	-	Medium	3
10	USA_CAR	Heavy	High	Medium	-	Low	2
14	Toyota	Medium	Low	-	Manu	High	2
15	Mazda	Light	Medium	Medium	Manu	High	4
17	-	Medium	-	High	Manu	High	4

Notice that for attribute '*Model*' in table 5, the concepts 'Toyota' and 'Mazda' are less general than 'USA_CAR' or 'JAPAN_CAR'. With a rule threshold, no rule inconsistencies arise as the number of rules would be few. Arguably, a small rule threshold means producing more general rules that may not be interesting. Rule 1 (first row of table 5) shows that about 32% of cars having 'Medium' weight, 'High' power and 'Medium' mileage are USA cars. If any of the less significant attributes like '*Tran*', '*Comp*', or '*Power*' were removed, no rule inconsistencies arise or even when all three are removed together. The rules with keys 1 and 9 are logically included in the rule with *key* 3 when '*Comp*' and 'Tran' are generalised to 'ANY'. If attributes '*Tran*', '*Comp*', '*Mode*l' and '*Power*' are removed, rules with *keys* 1 and 9 are inconsistent with rules with *keys* 14 and 17 i.e. „if (weight=medium) then (mileage=Medium)" is inconsistent with „if (weight=medium) then (mileage=High)" respectively. So we keep the attribute '*Model*'. Similarly, if '*Power*', '*Model*' and '*Tran*' were removed, the rule „if (weight=medium) then (mileage=Medium)" with *keys* 1 and 9 is inconsistent with rule „if (weight=medium) then (mileage=High)" with *key* 14 unless '*Model*' or '*Power*' is kept and so on.

Table 6. Final Table (9 tuples)

Model	Weight	Power	Comp	Tran	Mileage
-	Heavy	-	Medium	-	Low
USA_CAR	Medium	High	-	-	Medium
USA_CAR	Medium	-	Medium	-	Medium
-	Medium	-	-	Auto	Medium
USA_CAR	-	Low	-	-	Medium
-	Heavy	-	High	-	Medium
-	-	Medium	High	Manu	High
Japan	-	-	-	-	High
-	Light	-	-	-	High

Using this approach, fewer or no inconsistent rules are generated unless most attributes are dropped. This is because step 2 of the algorithm clusters attribute values with *common parents* making tuples less equivalent. Analysing table 6 from [6] shows that attribute '*Model*' has no values 'Toyota' and 'Mazda' as they have been over generalised to value 'Japan'. Also, more significant attributes like 'Model' and 'Weight' (table 5) have larger numbers of 'non-ANY' attribute values than those in table 6. This comparison is also true for less significant attributes. In general, the number of 'don't-care' or 'ANY' values in table 5 (12 of them) as compared to those in table 6 (26 of them) shows how cautiously the algorithm generalises the data by delaying replacement of 'ANY'. This is advantageous in preserving interestingness of the rules.

5 Conclusion

We have presented the COMPARE algorithm of complexity O (*np*) that integrates predictive and *lookahead strategies*, by use of thresholds and '*common parent*' respectively, for rule interestingness in *AOI*. Using these two approaches and the rough set approach for removing noisy data at rule generation, interesting multiple-level rules are produced in *AOI*. Further work can be summarised as follows:

♦ Repeatedly check rule thresholds when merging. Both our approach and classical *AOI* overlook this. For example, a rule threshold of 9 may produce 8 rules instead of 9 (see tables 5,6). In table 4, merging tuples with *keys* represented by *key* pairs (1,6), (8,23) reduces the tuples from 12 to 10. The next merge is pair (10,20) and the rule threshold of 9 is satisfied. Therefore, we need not merge the next pair (11,17). In large databases, this would be important for preserving interestingness.

♦By Integrating with relevant attribute selection [6] prior to mining and the interestingness approach presented, more interesting rules may be produced.

References

1. Han, J.; Cercone, N. and Cai, Y. 1991. „*Attribute-Oriented Induction in Relational Databases*“ In G. Piatetsky-Shapiro and W. J. Frawley, editors, Knowledge Discovery in Databases, pp 213-228.
2. Muyeba, K. M., and Keane, J.A 1999. „*Extending Attribute-Oriented Induction as a Key-Preserving Data Mining Method*“ In Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'99), Prague, Czech. Republic, pp 448-455.
3. Fudger, D. and Hamilton, H. 1993. „*Heuristic Evaluation of Databases for Knowledge Discovery with DBLEARN"* International Workshop on Rough Sets and Knowledge Discovery, Banff, Canada, pp 29-39.
4. Shan, N.; Howard, J. H. and Cercone, N. 1996. „*GRG: Knowledge Discovery Using Information Generalisation, Information Reduction and Rule Generation*“ In International Journal of Artificial Intelligence tools, 5: (1&2), pp 99–112.
5. Silberschatz, A. and Tuzhilin, A. 1995. „*On subjective measures of Interestingness in Knowledge Discovery*“ In Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95), Montreal, Canada, pp 275-281.
6. Barber, B. and Howard, H. J. 1997. „*A Comparison of Attribute Selection Strategies for Attribute-Oriented Generalisation*“ In International Symposium on Methodologies for Intelligent Systems (ISMIS'97), Charlotte, NC, pp 106-116.
7. Yongjian, F. 1996. „*Discovery of Multiple-level Rules from Large Databases*“, Ph.D. thesis, Simon Fraser University, Canada.
8. Carter, C.L.; Hamilton, H. J. and Cercone, N. 1994. „*The Software Architecture of DBLEARN*“, Technical Report CS-94-04, University of Regina.
9. Quinlan, J. R. 1986. „*Induction of decision trees*“, Machine Learning, Vol. 1, pp 81-106.