Fundamenta Informaticae 18 (1993), 193–207

Selected Algorithms of Machine Learning from Examples

Jerzy W. GRZYMALA-BUSSE
Department of Computer Science, University of Kansas
Lawrence, KS 66045, U. S. A.

Abstract. This paper presents and compares two algorithms of machine learning from examples, ID3 and AQ, and one recent algorithm from the same class, called LEM2. All three algorithms are illustrated using the same example. Production rules induced by these algorithms from the well-known Small Soybean Database are presented. Finally, some advantages and disadvantages of these algorithms are shown.

1. Introduction

This paper presents and compares two algorithms of machine learning, ID3 and AQ, and one recent algorithm, called LEM2. Among current trends in machine learning: similarity-based learning (also called empirical learning), explanation-based learning, computational learning theory, learning through genetic algorithms and learning in neural nets, only the first will be discussed in the paper. All three algorithms, ID3, AQ, and LEM2 represent learning from examples, an approach of similarity-based learning. In learning from examples the most common task is to learn production rules or decision trees. We will assume that in algorithms ID3, AQ and LEM2 input data are represented by examples, each example described by values of some attributes.

Let U denote the set of examples. Any subset C of U may be considered as a *concept* to be learned. An example from C is called a *positive example* for C, an example from $U – C$ is called a *negative example* for C. Customarily the concept C is represented by the same, unique value of a variable d, called a *decision*. Thus, each value w of a decision d describes a unique concept C.

The task of learning from examples is to learn the *description* D of the concept C [13]. The description D learned from concept C may be expressed by a set of rules or a decision tree. The description D, characterizing the set C, is easier to comprehend for humans. At the same time, description D is usable by other computer programs, such as expert systems.
Two important properties: *completeness* and *consistency* characterize descriptions. The first one, completeness, is defined as follows: For any example e from concept C there exists a description D such that D describes e. The second one, consistency, means that for every example e from concept C there are not two different descriptions D and D' such that both D and D' describe e.

Algorithms, used in learning from examples, usually produce *discriminant descriptions*, defined as complete and consistent, although some produce *characteristic descriptions*, defined just as complete. All algorithms, ID3, AQ, and LEM2, presented in this paper, produce discriminant descriptions.

Many algorithms, used in learning from examples, are equipped with some ways of generalization of their outputs. The most common method of generalization is called *dropping conditions* and is used for simplification of rules. Say that the original production rule is

$$C_1 \wedge C_2 \wedge \cdots \wedge C_j \rightarrow A,$$

where C_1, C_2,..., C_j are conditions and A is an action. Usually conditions are presented as ordered pairs (a, v), where a is an attribute and v is its value, and action A is presented as an ordered pair (d, w), where d is a decision and w is its value. Thus (d, w) describes some concept C. If the production rule

$$C_1 \wedge C_2 \wedge \cdots \wedge C_{i-1} \wedge C_{i+1} \wedge \cdots \wedge C_j \rightarrow A,$$

obtained from the original rule by dropping the condition C_i, describes some examples from the concept C and none from $U - C$, then it is said to be obtained by dropping condition from the original production rule.

All three algorithms: ID3, AQ and LEM2 are *general*—may be used for the entire spectrum of different real-life problems. Though there exist *incremental* versions of all three algorithms, we will focus our attention on *nonincremental* ones. Thus, the entire data are visible to the algorithms at once, as opposed to incremental algorithms, where data are seen gradually, one example at a time. We also will assume that input data are *consistent*, i.e., that two examples characterized the same way by all attributes belong to the same concept. Finally, any example has a specified value for every attribute, i.e., no missing values of attributes are considered.

2. ID3 Algorithm—Information Gain Version

The ID3 algorithm [19] is a successor of CLS algorithm [10]. Many algorithms based on ID3 have been developed, see [5, 11, 12, 15, 16, 17, 20–28]. The main task of ID3 is constructing a decision tree. Nodes of the tree are labeled by attributes while arches are labeled by values of an attribute. Our assumption is that the set U of examples is partitioned into at least two concepts. The attribute that is a label of the root is selected on the basis of the maximum of information gain criterion. Let a be an attribute with values a_1, a_2,..., a_l and let d be a decision with values d_1, d_2,..., d_k. Then the information gain $I(a \rightarrow d)$ is equal to $H(d) - H(d \mid a)$, where $H(d)$ is the entropy of decision d,

$$H(d) = -\sum_{i=1}^{k} p(d_i) \cdot \log p(d_i),$$

and $H(d \mid a)$ is the conditional entropy of decision d given attribute a,

$$\begin{aligned} H(d \mid a) &= \sum_{j=1}^{l} p(a_j) \cdot H(d \mid a_j) \\ &= -\sum_{j=1}^{l} p(a_j) \cdot \sum_{i=1}^{k} p(d_i \mid a_j) \cdot \log p(d_i \mid a_j). \end{aligned}$$

At this point, the corresponding decision tree, created by ID3, has the root, labeled by the attribute a and outgoing arches from the root, each such arch corresponds to a value a_j of the attribute. The set of all examples with the same value a_j of attribute a consists of a new set S

Table 1

	Attributes			*Decision*
	Temperature	**Headache**	**Nausea**	**Flu**
1	high	yes	no	yes
2	very_high	yes	yes	yes
3	normal	no	no	no
4	high	yes	yes	yes
5	high	no	yes	no
6	normal	yes	no	no
7	normal	no	yes	no

of examples. When all members of *S* are members of the same concept *C*, they do not need to be further partitioned, so *S* is a label of the leaf of the tree. However, when *S* contains examples from at least two different concepts, the node of the tree labeled by *S* is the root of a new subtree. An attribute to be a label for the root of this new subtree is selected among remaining attributes again on the basis of the maximum of information gain criterion.

Table 1 is an example of the decision table with three attributes: Temperature, Headache, and Nausea, seven examples, and two concepts. The first concept is the set {1, 2, 4} of examples that have value yes for the decision Flu, the second concept is the set {3, 5, 6, 7} of examples that have value no for the same decision. The entropy of the decision Flu is

$$H\,(Flu) = -\tfrac{3}{7}\cdot\log\tfrac{3}{7} - \tfrac{4}{7}\cdot\log\tfrac{4}{7} = 0.985.$$

Note that the entropy *H* (*Flu*) is computed from relative frequencies rather than from probabilities. The first candidate to be a label for the root of decision tree is the attribute *Temperature*. The corresponding partitioning of set *U* of examples is presented in Figure 1.

The conditional entropy of decision *Flu* given attribute *Temperature* is

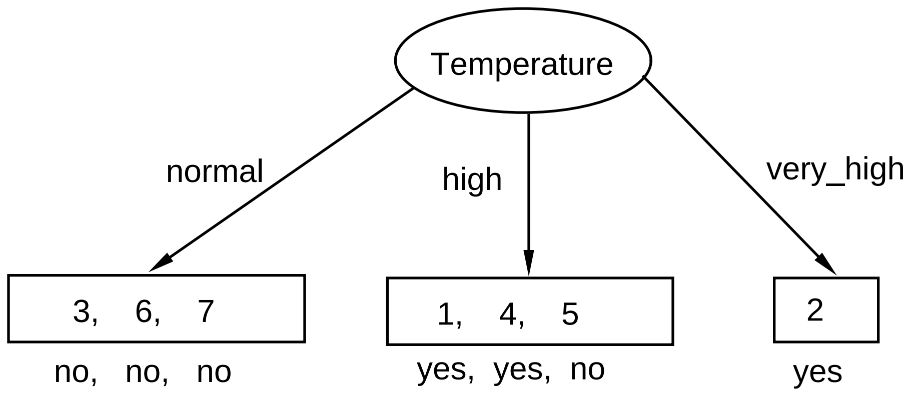

Figure 1. Decision tree—first version

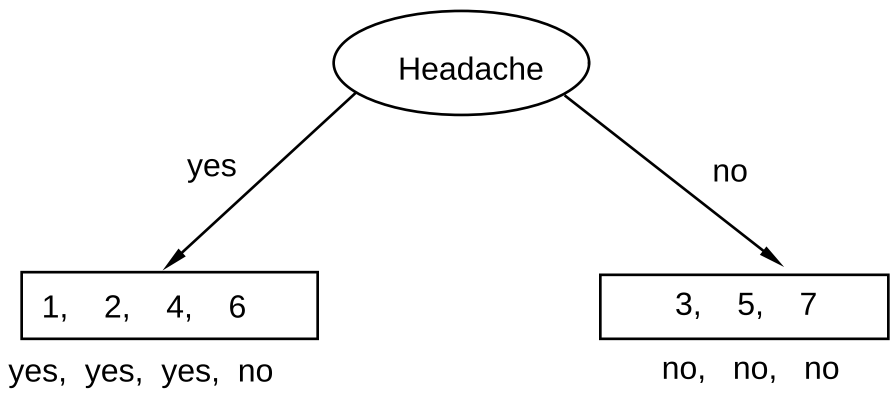

Figure 2. Decision tree—second version

$$H\,(Flu \mid Temperature) = -\tfrac{3}{7}\cdot 0 - \tfrac{3}{7}\cdot(\tfrac{1}{3}\cdot\log\tfrac{1}{3} + \tfrac{2}{3}\cdot\log\tfrac{2}{3}) - \tfrac{1}{7}\cdot 0 = 0.394,$$

and the corresponding information gain is

$$I\,(Temperature \rightarrow Flu) = 0.985 - 0.394 = 0.591.$$

The next candidate to be a label for the root of the decision tree is attribute *Headache*, see Figure 2.

In this case,

$$H\,(Flu \mid Headache) = -\tfrac{4}{7}\cdot(\tfrac{1}{4}\cdot\log\tfrac{1}{4} + \tfrac{3}{4}\cdot\log\tfrac{3}{4}) - \tfrac{3}{7}\cdot 0 = 0.464,$$

and

$$I\,(Headache \rightarrow Flu) = 0.985 - 0.464 = 0.521.$$

For the remaining attribute, *Nausea*, the partitioning of examples is presented in Figure 3,

$$H\,(Flu \mid Nausea) = -\tfrac{4}{7}\cdot(\tfrac{1}{2}\cdot\log\tfrac{1}{2} + \tfrac{1}{2}\cdot\log\tfrac{1}{2}) - \tfrac{3}{7}\cdot(\tfrac{1}{3}\cdot\log\tfrac{1}{3} + \tfrac{2}{3}\cdot\log\tfrac{2}{3}) = 0.965,$$

and

$$I\,(Nausea \rightarrow Flu) = 0.985 - 0.965 = 0.020.$$

It is clear that the attribute *Temperature* is the best candidate to be a label for the root of

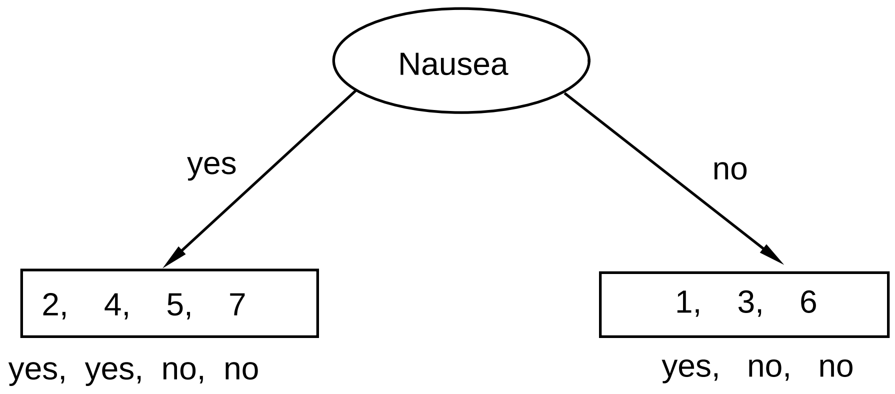

Figure 3. Decision tree—third version

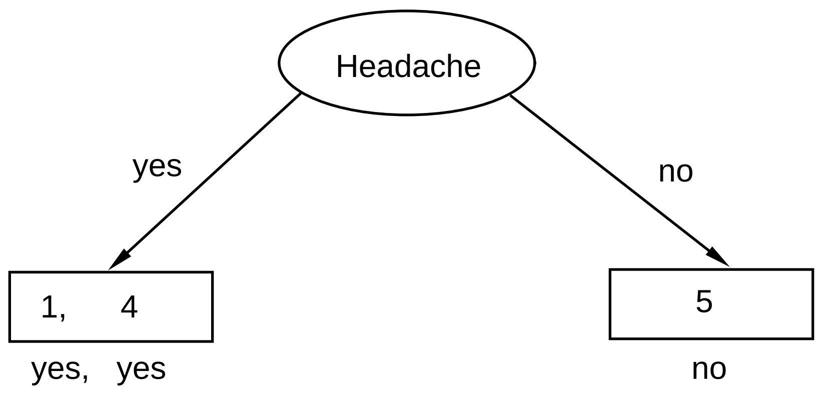

Figure 4. Decision tree—second level

the decision tree since the corresponding information gain is maximal. As follows from Figure 1, the next step is to find an attribute to distinguish examples from the set {1, 4, 5} (all examples from this set are characterized by value *high* of attribute *Temperature*) since examples 1 and 4 belong to another concept then example 5. Remaining attributes are *Headache* and *Nausea*. As follows from Figures 4 and 5, attribute *Headache* should be selected since the corresponding information gain is maximal. The resulting decision tree is presented in Figure 6.

From the decision tree (see Figure 6) the following rules may be induced

$$\text{(Temperature, normal)} \rightarrow \text{(Flu, no)},$$

$$\text{(Temperature, high)} \wedge \text{(Headache, yes)} \rightarrow \text{(Flu, yes)},$$

$$\text{(Temperature, high)} \wedge \text{(Headache, no)} \rightarrow \text{(Flu, no)},$$

$$\text{(Temperature, very\_high)} \rightarrow \text{(Flu, yes)}.$$

From the rule

$$\text{(Temperature, high)} \wedge \text{(Headache, yes)} \rightarrow \text{(Flu, yes)}$$

neither condition (Temperature, high) nor condition (Headache, yes) may be dropped.

However, from the rule

$$\text{(Temperature, high)} \wedge \text{(Headache, no)} \rightarrow \text{(Flu, no)}$$

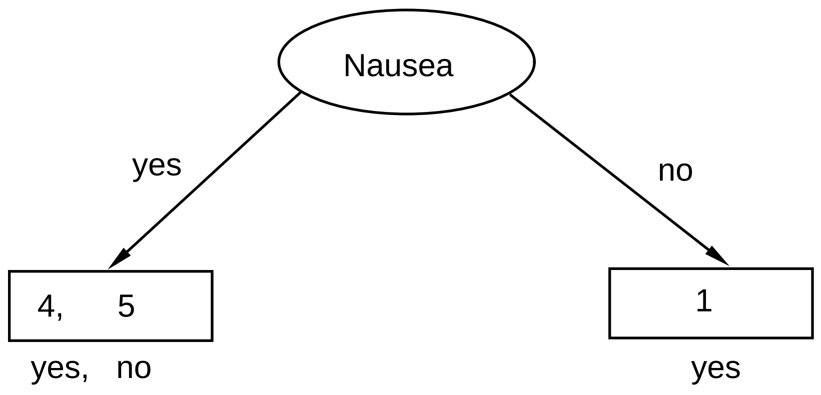

Figure 5. Decision tree—second level

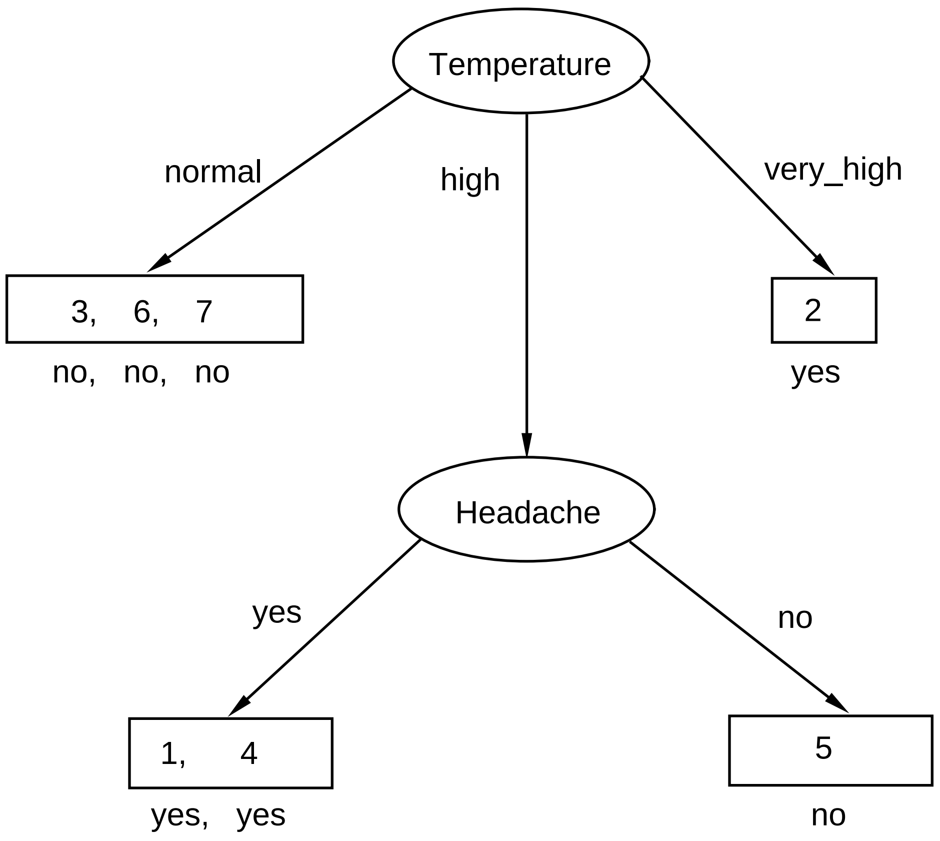

Figure 6. Final decision tree

the first condition, (Temperature, high), may be dropped, the resulting rule is

$$(\text{Headache, no}) \rightarrow (\text{Flu, no}).$$

3. ID3 Algorithm—Gain Ratio Version

The information gain version of ID3, listed in Section 2, builds the decision tree with some bias—attributes with greater number of values are preferred by the algorithm. In order to avoid this bias, another version of ID3 [21] has been developed. This version uses another criterion for attribute selection, called *gain ratio*. The gain ratio is defined as follows

$$\frac{I(a \rightarrow d)}{H(a)}$$

Table 2

Attribute	**Gain Ratio**
Temperature	0.408
Headache	0.529
Nausea	0.020

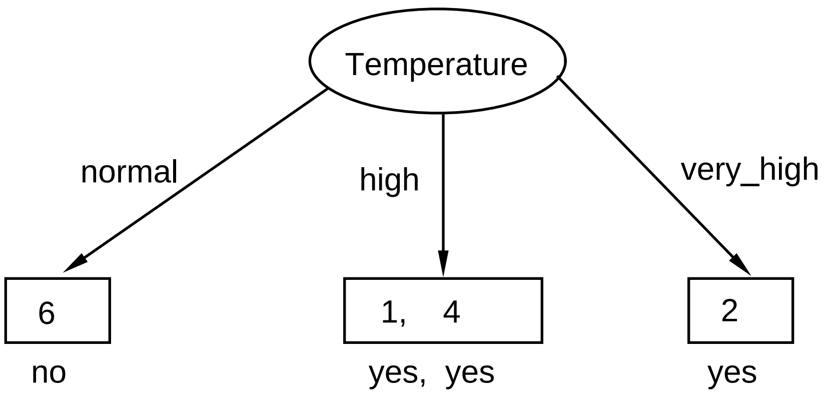

Figure 7. Decision tree—second level

where *a* is an attribute and *d* is a decision. The gain ratio version of ID3 is the same as the algorithm ID3 described in Section 2 with only one change—information gain criterion is replaced by gain ratio criterion. Gain ratios for all attributes from Table 1 are presented in Table 2.

The attribute *Headache* should be selected as the label of the root for the decision tree. The next question is what an attribute should be selected to distinguish examples from the set {1, 2, 4, 6}. The subtrees, resulting from the remaining two candidates, attributes *Temperature* and *Nausea*, are presented in Figures 7 and 8. It is immediately clear that attribute *Temperature* is the correct choice, since it partitions examples into classes with the same values of the decision.

The final decision tree, induced by the gain ratio version of ID3, is presented in Figure 9. From the decision tree, presented in Figure 9, the following rules may be induced

$$\text{(Headache, yes)} \wedge \text{(Temperature, normal)} \rightarrow \text{(Flu, no)},$$

$$\text{(Headache, yes)} \wedge \text{(Temperature, high)} \rightarrow \text{(Flu, yes)},$$

$$\text{(Headache, yes)} \wedge \text{(Temperature, very\_high)} \rightarrow \text{(Flu, yes)},$$

$$\text{(Headache, no)} \rightarrow \text{(Flu, no)}.$$

The simplified rules, after dropping conditions, are

$$\text{(Temperature, normal)} \rightarrow \text{(Flu, no)},$$

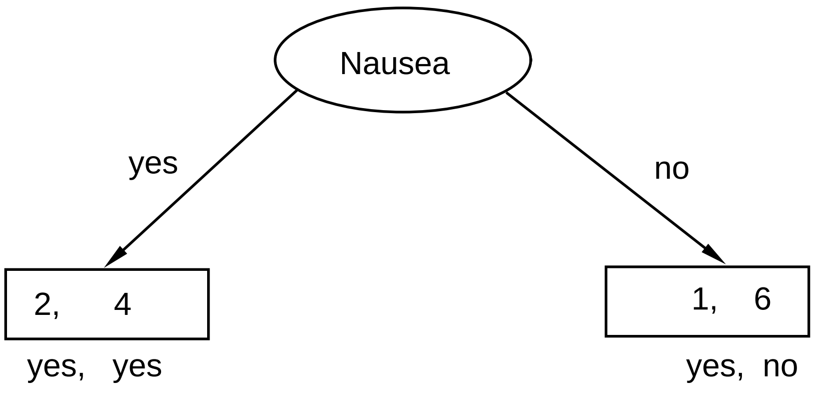

Figure 8. Decision tree—second level

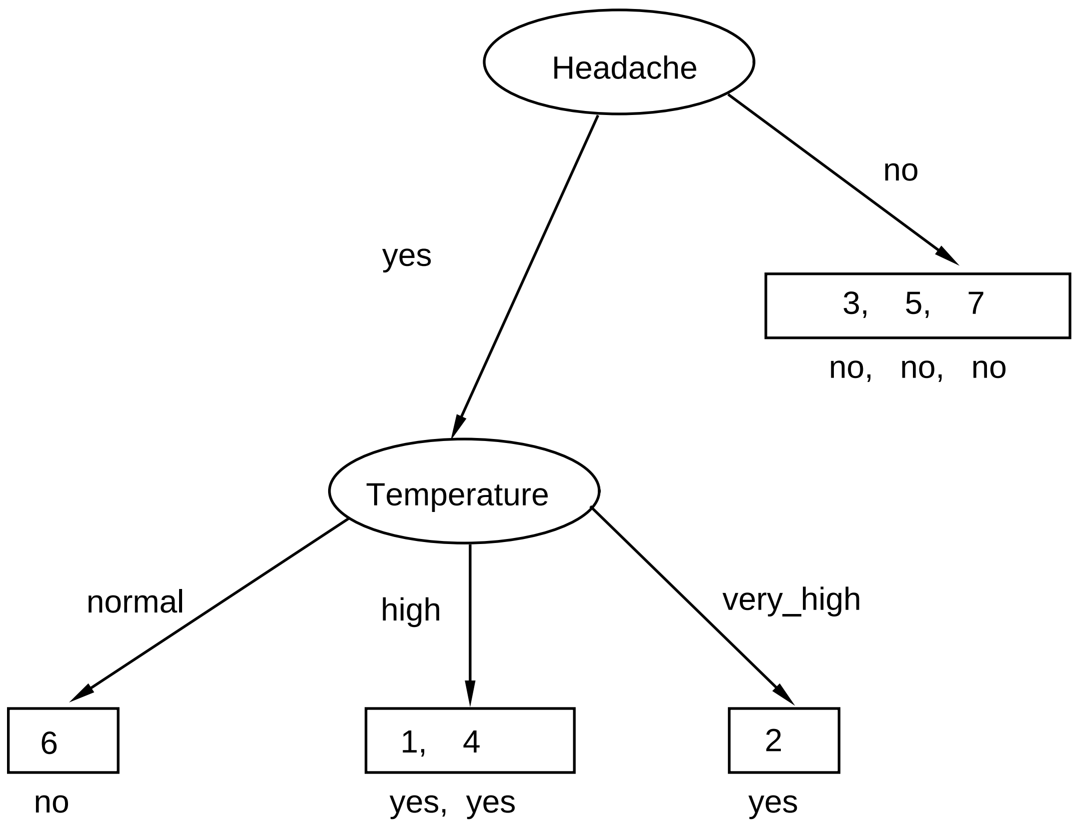

Figure 9. Final decision tree

(Headache, yes) $\wedge$ (Temperature, high) $\rightarrow$ (Flu, yes),

(Temperature, very_high) $\rightarrow$ (Flu, yes),

(Headache, no) $\rightarrow$ (Flu, no).

4. AQ Algorithm

Another method of learning from examples, developed by R. S. Michalski and his collaborators in the early seventies, is an algorithm called AQ. Many versions of the algorithm, under different names, have been developed. The newer version is AQ15 [14].

Let us start by quoting some definitions from [14]. A *seed* is a member of the concept, i.e., a positive example. A *selector* is an expression that associates a variable (attribute or decision) to a value of the variable, e.g., a negation of value, a disjunction of values, etc. A *complex* is a conjunction of selectors. A *partial star* $G(e \mid e_1)$ is a set of all complexes describing the seed $e = (x_1, x_2,..., x_k)$ and not describing a negative example $e_1 = (y_1, y_2,..., y_k)$. Thus, the complexes of $G(e \mid e_1)$ are all selectors of the form $(x_i, \neg y_i)$, where $x_i \neq y_i$. A star $G(e \mid F)$ is constructed from all partial stars $G(e \mid e_i)$, for all $e_i \in F$, and then by conjuncting these partial stars by each other, using absorption law to eliminate redundancy. For a given concept C, a *cover* is a disjunction of complexes describing all positive examples from C and not describing any negative examples from $F = U - C$.

The main idea of the AQ algorithm is to generate a cover for each concept by computing stars and selecting from them single complexes to the cover.

For the example from Table 1, and concept C = {1, 2, 4} described by (*Flu*, *yes*), set F

of negative examples is equal to {3, 5, 6, 7}. A seed is any member of *C*, say that it is example 1. Then the partial star *G*(1 | 3) is equal to

{(Temperature, ¬normal), (Headache, yes)}.

Obviously, partial star *G*(1 | 3) describes negative example 5. The partial star *G*(1 | 5) equals

{(Headache, yes), (Nausea, no)}.

The conjunct of *G*(1 | 3) and *G*(1 | 5) is equal to

{(Temperature, ¬normal) ∧ (Headache, yes),

(Temperature, ¬normal) ∧ (Nausea, no),

(Headache, yes) ∧ (Headache, yes),

(Headache, yes) ∧ (Nausea, no)},

after using absorption law, this set is reduced to the following set

{(Temperature, ¬normal) ∧ (Nausea, no),

(Headache, yes)}.

The preceding set describes negative example 6. The partial star *G*(1 | 6) is equal to

{(Temperature, ¬normal)}.

The conjunct of the preceding two sets:

{(Temperature, ¬normal) ∧ (Nausea, no),

(Temperature, ¬normal) ∧ (Headache, yes)}

already is a star *G*(1 | *F*). Both complexes contain two selectors. However, the first complex describes only one positive example 1, while the second complex describes all three positive examples: 1, 2, and 4. Therefore, the complex

(Temperature, ¬normal) ∧ (Headache, yes)

should be selected to be the only member of the cover of *C*. The corresponding rule is

(Temperature, ¬normal) ∧ (Headache, yes) → (Flu, yes).

If rules without negation are preferred, the preceding rule may be replaced by the following two rules

(Temperature, high) ∧ (Headache, yes) → (Flu, yes),

(Temperature, very_high) ∧ (Headache, yes) → (Flu, yes), or, after dropping conditions,

$$\text{(Temperature, high)} \wedge \text{(Headache, yes)} \rightarrow \text{(Flu, yes)},$$

$$\text{(Temperature, very\_high)} \rightarrow \text{(Flu, yes)}.$$

The production rules for the concept {3, 5, 6, 7} may be induced by AQ in a similar way. Note that the AQ algorithm demands computing conjuncts of partial stars. In the worst case, time complexity of this computation is $O(n^m)$, where n is the number of attributes and m is the number of examples. The authors of AQ suggest using the parameter MAXSTAR as a method of reducing the computational complexity. According to this suggestion, any set, computed by conjunction of partial stars, is reduced in size if the number of its members is greater than MAXSTAR. Obviously, the quality of the output of the algorithm is reduced as well.

5. LEM2 Algorithm

The LEM2 algorithm [3] is a local algorithm, dealing with attribute-value pairs, as opposed to global algorithms, such as ID3 or AQ, dealing with entire attributes. Another example of a local algorithm has been presented in [4]. The algorithm LEM2 does not need dropping conditions for rules because it is local. There exists a global algorithm LEM as well [6]. Both LEM and LEM2 algorithms are used as modules in the algorithm LERS for learning from examples based on rough sets [1, 2, 7–9]. The idea of a rough set has been introduced in [18].

The following is a summary of the main ideas of the LEM2 algorithm. A block of an attribute-value pair $t = (a, v)$, denoted $[t]$, is the set of all examples that for attribute a have value v. A concept, described by the value w of decision d, is denoted $[(d, w)]$, and it is the set of all examples that have value w for decision d. Let C be a concept and let T be a set of attribute-value pairs. Concept C *depends on* a set T if and only if

$$\emptyset \neq [T] = \bigcap_{t \in T} [t] \subseteq C.$$

Set T is a *minimal complex* of concept C if and only if C depends on T and T is minimal. Let $\mathbb{T}$ be a nonempty collection of nonempty sets of attribute-value pairs. Set $\mathbb{T}$ is a *local covering of C* if and only if the following three conditions are satisfied:

(1) each member of $\mathbb{T}$ is a minimal complex of C,

(2) $\bigcup_{T \in \mathbb{T}} [T] = B$,

and

(3) $\mathbb{T}$ is minimal, i.e., $\mathbb{T}$ has the smallest possible number of members.

For each concept C, the LEM2 algorithm induces production rules by computing a local covering $\mathbb{T}$. Any set T, a minimal complex which is a member of $\mathbb{T}$, is computed from attribute-value pairs selected from the set $T(G)$ of attribute-value pairs relevant with a current goal G, i.e., pairs whose blocks have nonempty intersection with G. The initial goal G is equal to the concept and then it is iteratively updated by subtracting from G the set of examples described by the set of minimal complexes computed so far. Attribute-value pairs from T which are selected as the most relevant, i.e., on the basis of maximum of the cardinality of $[t] \cap G$, if a tie occurs, on the basis of the smallest cardinality of $[t]$. The last condition is equivalent to the maximal conditional probability of goal G given attribute-value

Table 3

t	[t] ∩ G
(Temperature, high)	{1, 4}
(Temperature, very_high)	{2}
(Headache, yes)	{1, 2, 4}
(Nausea, no)	{1}
(Nausea, yes)	{2, 4}

pair *t*.

For the example from Table 1, the blocks of all attribute-value pairs are

[(Temperature, high)] = {1, 4, 5},

[(Temperature, very_high)] = {2},

[(Temperature, normal)] = {3, 6, 7},

[(Headache, yes)] = {1, 2, 4, 6},

[(Headache, no)] = {3, 5, 7},

[(Nausea, no)] = {1, 3, 6],

[(Nausea, yes)] = {2, 4, 5, 7}.

Say that the concept C is the set {1, 2, 4}. Initially, the goal *G* is equal to *C*. The set *T*(*G*) of all attribute-value pairs relevant with goal *G* is

{(Temperature, high), (Temperature, very_high),

(Headache, yes), (Nausea, no), (Nausea, yes)}.

As follows from Table 3, the attribute-value pairs (Headache, yes) should be selected as the most relevant with goal *G*.

Furthermore, [(Headache, yes)] = {1, 2, 4, 6} $\not\subseteq$ {1, 2, 4}, so (Headache, yes) is not a

Table 4

t	[t] ∩ G
(Temperature, high)	{1, 4}
(Temperature, very_high)	{2}
(Nausea, no)	{1}
(Nausea, yes)	{2, 4}

Table 5

t	[t] ∩ G
(Temperature, very_high)	{2}
(Headache, yes)	{2}
(Nausea, yes)	{2}

minimal complex of *C*. The algorithm LEM2 looks for another attribute-value pair *t*. Remaining attribute-value pairs, relevant with *G*, are listed in Table 4.

There is a tie between (Temperature, high) and (Nausea, yes). The attribute-value pair (Temperature, high) is selected because |{(Temperature, high)]| = 3 and |[(Nausea, yes)]| = 4. The first minimal complex *T* is equal to

{(Headache, yes), (Temperature, high)},

because [(Headache, yes)] ∩ [(Temperature, high)] = {1, 4}.

The set *T* describes examples {1, 4}, thus the goal *G* is equal to the set consisting of the remaining example 2. The set *T*(*G*) of all relevant attribute-value pairs is

{(Temperature, very_high), (Headache, yes), (Nausea, yes)}.

As follows from Table 5, there is a tie between all three attribute-value pairs. The cardinality of the block of (Temperature, very_high) is minimal, hence this pair should be selected. At the same time, {(Temperature, very_high)} is a minimal complex. Thus, the local covering of the concept *C* = {1, 2, 4} is the following set

{{(Headache, yes), (Temperature, high)},

{(Temperature, very_high)}}.

The corresponding production rules are

(Headache, yes) ∧ (Temperature, high) → (Flu, yes),

(Temperature, very_high) → (Flu, yes).

The production rules for the concept {3, 5, 6, 7} may be induced by LEM2 in a similar way.

6. Production Rules Induced from the Small Soybean Database

The Small Soybean Database has been frequently used in the area of machine learning to compare different algorithms. The database presents soybean disease diagnosis. It has 35 attributes *a*1, *a*2,..., *a*35 and 47 examples. The following production rules have been induced by the preceding algorithms.

ID3—Information Gain Version (without dropping conditions)

$(a22, 2) \rightarrow (class, D4)$,
$(a22, 3) \rightarrow (class, D2)$,
$(a22, 0) \rightarrow (class, D1)$,
$(a22, 1) \wedge (a28, 0) \rightarrow (class, D1)$,
$(a22, 1) \wedge (a28, 3) \rightarrow (class, D3)$.

ID3—Gain Ratio Version (without dropping conditions)

$(a23, 0) \wedge (a22, 1) \rightarrow (class, D3)$,
$(a23, 0) \wedge (a22, 3) \rightarrow (class, D2)$,
$(a23, 0) \wedge (a22, 2) \rightarrow (class, D4)$,
$(a23, 1) \rightarrow (class, D1)$.

AQ

$(a23, \neg 0) \rightarrow (class, D1)$,
$(a26, \neg 0) \rightarrow (class, D2)$,
$(a22, \neg 0) \wedge (a28, \neg 0) \rightarrow (class, D3)$,
$(a12, \neg 0) \wedge (a35, \neg 0) \rightarrow (class, D4)$.

LEM2

$(a21, 3) \rightarrow (class, D1)$,
$(a3, 0) \rightarrow (class, D2)$,
$(a4, 0) \wedge (a22, 1) \rightarrow (class, D3)$,
$(a22, 2) \rightarrow (class, D4)$.

7. Conclusions

This paper presents three different algorithms of machine learning from examples. The first algorithm, ID3, although very simple, produces decision trees instead of production rules. Production rules must then be induced from the decision tree. Thus the entire algorithm is biased. Another bias is introduced by the criterion for the choice of attributes as labels for the decision tree.

The main advantage of the second algorithm, AQ, is that induced production rules may easily express negation. Thus the production rules may be simpler than the production rules induced by other algorithms. However, the time complexity of AQ is exponential, and with the use of the parameter MAXSTAR the induced rules may be of poor quality. Moreover, if induced rules need to be converted into different format, e.g., when negation is not desired, the quality of the new rules may further deteriorate.

Algorithm LEM2 is of polynomial time complexity and is inducing the minimal discriminant description. However, as with any machine learning algorithms of polynomial complexity, there is no guarantee that the induced description is optimal, i.e., that there is no other minimal discriminant description with the smaller number of conditions.

References

[1] A. Budihardjo, J. W. Grzymala-Busse, and L. Woolery, Program LERS_LB 2.5 as a tool for knowledge acquisition in nursing, *Proc. of the 4th Int. Conf. on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems* (1991) 735–740.

[2] C. C. Chan and J. W. Grzymala-Busse, Rough-set boundaries as a tool for learning rules from examples, *Proc. of the ISMIS–89, 4th Int. Symp. on Methodologies for Intelligent Systems* (1989) 281–288.

[3] C. C. Chan and J. W. Grzymala-Busse, On the attribute redundancy and the learning programs ID3, PRISM, and LEM2, *Report TR-91–14, Department of Computer Science, University of Kansas*, 1991.

[4] J. Cendrowska, PRISM: An algorithm for inducing modular rules, *Int. J. Man-Machine Studies* **27** (1987) 349–370.

[5] P. Clark and T. Niblett, The CN2 induction algorithm, *Machine Learning* **3** (1989) 261–283.

[6] J. S. Dean and J. W. Grzymala-Busse, An overview of the learning from examples module LEM1, *Report TR-88-2, Department of Computer Science, University of Kansas*, 1988.

[7] J. W. Grzymala-Busse, Knowledge acquisition under uncertainty—A rough set approach, *J. Intelligent & Robotic Systems* **1** (1988) 3–16.

[8] J. W. Grzymala-Busse, An overview of the LERS1 learning system, *Proc. of the 2nd Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems* (1989) 838–844.

[9] J. W. Grzymala-Busse and D. J. Sikora, LERS1—A system for learning from examples based on rough sets, *Report TR-88-5, Department of Computer Science, University of Kansas*, 1988.

[10] E. B. Hunt, J. Marin, and P. J. Stone, *Experiments in Induction*, Academic Press, 1966.

[11] I. Kononenko, ID3, sequential Bayes, naive Bayes and Bayesian neural networks, *Proc. of the 4th European Working Session on Learning* (1989) 91–98.

[12] I. Kononenko and I. Bratko, Information-based evaluation criterion for classifiers performance, *Machine Learning* **6** (1991) 67–80.

[13] R. S. Michalski, A theory and methodology of inductive learning. In: R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), *Machine Learning,* Morgan Kaufmann, 1983, 83–134.

[14] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, The AQ15 inductive learning system: An overview and experiments, *Report 1260, Department of Computer Science, University of Illinois at Urbana-Champaign*, 1986.

[15] J. Mingers, An empirical comparison of selection measures for decision-tree induction, *Machine Learning* **3** (1989) 319–342.

[16] J. Mingers, An empirical comparison of pruning methods for decision tree induction, *Machine Learning* **4** (1989) 227–243.

[17] T. Niblett and I. Bratko, Learning decision rules in noisy domains, *Proc. of Expert Systems '86, the 6th Annual Tech.. Conference of the British Computer Society, Specialist Group on Expert Systems*, 1986, 25–34.

[18] Z. Pawlak, Rough sets, *Int. J. Computer and Information Sci.* **11** (1982) 341–356.

[19] J. R. Quinlan, Semi-autonomous acquisition of pattern-based knowledge. In: J. E. Hayes, D. Michie, and Y.-H. Pao (Eds.), *Machine Intelligence* **10**, Ellis Horwood, 1982, 159–172.

[20] J. R. Quinlan, Learning efficient classification procedures and their application to chess end games. In: R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), *Machine Learning,* Morgan Kaufmann, 1983, 461–482.

[21] J. R. Quinlan, Induction of decision trees, *Machine Learning* **1** (1986) 81–106.

[22] J. R. Quinlan, Decision trees as probabilistic classifiers, *Proc of the 4th Int. Workshop on Machine Learning*, 1987, 31–37.

[23] J. R. Quinlan, Generating production rules from decision trees, *Proc. of the 10th Int. Joint Conf. on AI*, 1987, 304–307.

[24] J. R. Quinlan, The effect of noise on concept learning. In: R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, (Eds.), *Machine Learning. An Artificial Intelligence Approach. Vol. II*, . Morgan Kaufmann Publishers, Inc., 1986, 149–166.

[25] J. R. Quinlan, Probabilistic decision trees. In: Y. Kodratoff and R. Michalski (Eds.), *Machine Learning. An Artificial Intelligence Approach. Vol. III*, Morgan Kaufmann Publishers, Inc., 1990, 140–152.

[26] R. L. Rivest, Learning decision lists, *Machine Learning* **2** (1987) 229–246.

[27] J. C. Schlimmer and D. Fisher, A case study of incremental concept induction, *Proc. of the AAAI-86, 5th Nat. Conf. on AI*, 1986, 496–501.

[28] P. E. Utgoff, ID5: An incremental ID3, *Proc. of the 5th Int. Conf. on Machine Learning*, 1988, 107–120.