On the Use of Data-Mining Techniques in Knowledge-Based Systems

Prof. Mihaela OPREA PhD.
Department of Informatics, University Petroleum-Gas of Ploieşti

In the last years, the class of machine learning algorithms were extended with new techniques as for example, data mining techniques. The purpose of data-mining techniques is to extract valuable implicit regularities contained in large databases. The paper presents an analysis of some data-mining techniques used in the development of a knowledge-based system.
***Keywords**: Machine Learning, Inductive Learning, Data Mining Techniques, Knowledge Based Systems.*

Introduction

The development of an efficient knowledge-based system (KBS) involves the development of an efficient knowledge base that has to be complete, coherent and non-redundant. The step of knowledge acquisition is one of the major bottlenecks in the stage of knowledge base development. Usually, for each application domain there are several sources of knowledge (human experts, the specialized literature which includes textbooks, books, reviews, collection of data during the run of similar systems, etc). In order to make knowledge extraction as much as correct as possible (i.e. in order to keep the correctness of the knowledge as it is kept at the source) different techniques could be applied [1]. Among these techniques, data mining techniques and, more general, knowledge discovery techniques became the most used in the recent years.

Data mining (DM) is a subfield of Machine Learning that enables finding interesting knowledge (patterns, models and relationships) in very large databases. It is the most essential part of the knowledge-discovery process, which combines databases, statistics, artificial intelligence and machine learning techniques. The basic techniques

for data-mining include: decision-tree induction, rule induction, instance-based learning, artificial neural networks, Bayesian learning, support vector machines, ensemble techniques, clustering, and association rules. Knowledge Discovery Databases (KDD) is the process of extracting and refining useful knowledge from large databases. It involves three stages: inductive learning, deductive verification and human intuition. Inductive learning focuses on data and tries to generate hypotheses from them. Deductive verification evaluates the evidential support from some previously given hypotheses, while human intuition helps the discovery guiding so that it gathers the information wanted by the user, in a certain time window. Data mining could be applied to any domain where large databases are saved. Examples of DM applications: prediction problems such as the prediction of paper making defects on-line [2], fault diagnosis, process and quality control in manufacturing environments [3], learn general rules for credit worthiness from financial databases, etc. In this paper, we briefly present some DM techniques, based on induction, and we make an analysis of these techniques applied in two cases of KBS development.

Data-Mining Techniques

The main steps followed by the data-mining process are: *human resource identification* (the domain expert, the data expert and the data mining expert), *problem specification* (problem decomposition in sub-problems), *data prospecting* (analysis of the state of the data required for solving the problem, identification of the attributes), *domain knowledge elicitation* (domain specific constraints on the search space, hierarchical generalizations defined on the identified attributes), *methodology identification* (traditional statistics, fuzzy logic and rough sets, evidence theory, case-based reasoning, rule induction, etc), *data pre-processing* (remove outliers in the data, predict and fill-in missing values, noise modeling etc), *pattern discovery* (algorithms that

automatically discover patterns from the preprocessed data), *knowledge-post-processing* (filter the discovered knowledge, knowledge validation, etc), *refinement* (redefining the data used in the discovery, refinement of the parameters of the DM algorithm, etc).

Four classes of knowledge can be discovered:

1) *Classification rules* – A classification rule attempts to predict the value of a discrete dependent variable from various known attributes. The most used learning algorithms are the induction of decision trees using a divide-and-conquer algorithm [4], and the induction of classification rule sets using a separate-and-conquer strategy [5].

2) *Regression rules* – Inducing regression rules is similar to the induction of classification rules, only that instead of predicting a discrete classification, the induced rules have to predict a numeric value. The methods used in this case range from the basic linear regression methods to the induction of regression trees or regression rules.

3) *Clusters/Taxonomy* – The task of clustering means the autonomous discover of meaningful patterns in the data without training examples. The best clustering techniques include AUTOCLASS [6] and COBWEB [7].

4) *Dependencies* – The discovery of data dependencies includes the discovery of association rules (dependencies between binary attributes), functional dependencies, and partial determinations that do not necessarily hold in all cases. An example of such a system is CLAUDIEN [8] which discovers general clause dependencies in a first-order logic framework.

Induction analyses a number of examples of good and bad profiles, trying to determine which data attribute is most different between the good and bad examples. The inductive process could be represented by a decision tree, with splits in the tree being based on the values of attributes in the processed data. The induction tree can be read and easily understood by the user. This has a significant impact on the development times since the developers can more quickly understand the combination of attributes for a certain application. Let BD be a database that contains N records: BD = $\{R_1, R_2, \ldots, R_N\}$. Each record is a set of unique tokens taken from the alphabet Σ, and the number of tokens may vary from record to record, i.e. $R_i = \{s_1, \ldots, s_{ni} \mid s_j \in \Sigma, 0 \le n_i \le |\Sigma|\}$. A pattern is defined in the same manner as a record. A pattern *p* occurs in a record *R* if $p \cap R = p$. A rule consists of a pair of patterns, *p* and c, which are called premise (or precursor) and conclusion (or successor). The deductive form of a rule is IF *p* THEN c.

For any given rule, we can construct the contingency table (Figure 1) which describes the frequency of co-occurrence of the corresponding patterns in a database. Suppose Number(*p*, *c*) denotes the number of records in BD that contains both *p* and *c*, meaning that $(p \cap R = p) \wedge (c \cap R = c)$. If either *p* or *c* are negated then the argument must not appear in the records.

	s	*not s*	
p	N_1	N_2	r_1
not p	N_3	N_4	r_2
Totals:	t_1	t_2	T

Fig. 1 The contingency table

Rule strength is measured by the G statistic. G is a statistical measure of association, with large values indicating that *p* and *c* co-occur more or less frequently than one would expect by random chance. For the contingency table shown in Figure 1, G is given by equation (1).

$$G = 2 \Sigma_{i=1,4} N_i \log (N_i / TN_i) \quad (1)$$

TN_i is the expected value of N_i under the assumption of independence, and is computed from the row margins and the table total. For example TN_1 is the probability that *p* and *c* will co-occur in a database record, given that they are independent, times the number of records in the database and is computed as $TN_1 = r_i t_i / T$. Strong dependencies are indicated by high values of G and they capture the structure in the database as they give us the

relationships between their constituent patterns, that occurrences of those patterns which are not independent.

The induction algorithms make a search in the space of all possible pairs of patterns defined over $\sum$ and returns the strongest dependencies found. The algorithms return all the rules associated with the highest values of G.

Some of the induction algorithms do not use a decision tree for rule extraction (e.g. ILA [9], DCL [10]), while others use a decision tree (e.g. ID3, C4.5 [1]). ID3 algorithm divides the training set into homogeneous subsets without eference to the class of the subset. This algorithm is concerned with finding the attribute which is most relevant overall even though some values of that attribute may be irrelevant. C4.5 is an extension of ID3 that handles uncertain data at the expense of increasing the classification rate.

The ILA algorithm works in an iterative way, each iteration searching for a rule that covers a large number of training examples of a single class. Having found a rule, ILA removes those examples it covers from the training set by marking them and appends a rule at the end of its rule set. The ILA algorithm will produce a list of rules without generating a decision tree. The DCL algorithm is an improved version of ILA, for disjunctive concept learning, which generates rules with AND/OR operators from a set of training examples. It will produce fewer number of rules than most of other induction algorithms.

Examples of knowledge base development

We start our analysis with an example of an abstract knowledge base for a problem which analyses three parameters: P_1, P_2 and P_3 and with the goal, *Decision*. Table 1 presents the training examples set, M_1.

Table 1. The set of training examples, M_1.

No.	P_1	P_2	P_3	Decision
1.	A_{11}	A_{23}	A_{32}	D_1
2.	A_{11}	A_{22}	A_{33}	D_2
3.	A_{12}	A_{22}	A_{33}	D_2
4.	A_{13}	A_{22}	A_{33}	D_2
5.	A_{11}	A_{22}	A_{32}	D_1
6.	A_{11}	A_{21}	A_{11}	D_0
7.	A_{11}	A_{21}	A_{31}	D_3
8.	A_{13}	A_{21}	A_{32}	D_3
9.	A_{13}	A_{21}	A_{31}	D_0
10.	A_{12}	A_{21}	A_{33}	D_2
11.	A_{13}	A_{23}	A_{32}	D_1

The domains of values for each variable analyzed and goal are as follows:

$$Dom(P_1)=\{A_{11}, A_{12}, A_{13}\},$$
$$Dom(P_2)=\{A_{21}, A_{22}, A_{23}\},$$
$$Dom(P_3)=\{A_{31}, A_{32}, A_{33}\},$$
$$Dom(Decizie)=\{D_0, D_1, D_2, D_3\}.$$

After applying the ID3 and ILA algorithms we have obtained the following rules:

Rule 1

IF $P_3 = A_{33}$ THEN *Decision* = D_2;

Rule 2

IF $P_3 = A_{31}$ THEN *Decision* = D_0;

Rule 3

(ID3) IF $P_2 = A_{23}$ AND $P_3 = A_{32}$ THEN *Decision* = D_1;

(ILA) IF $P_2 = A_{23}$ THEN *Decision* = D_1;

Rule 4

IF $P_2 = A_{22}$ AND $P_3 = A_{32}$ THEN *Decision* = D_1;

Rule 5

IF $P_2 = A_{21}$ AND $P_3 = A_{32}$ THEN *Decision* = D_3;

Except rule 3 all the rules were identical for both algorithms, ID3 and ILA. Rule 3 is simplified by the ILA algorithm i.e., the condition that is unnecessary was eliminated.

The second example considers the investment projects analysis by taken into account the following parameters: level of investment (LI), global risk (GR) and the investment recovery time (RT). The goal is the decision of accepting, rejecting or postponing a given investment project proposals.

The domain of values for each variable are as follows:

Dom(LI) = {High, Low, Medium}
Dom(GR) = {High, Low, Medium}
Dom(RT) = {Long, Medium, Short,

Very_short}
Dom(Decision) = {Yes, No}

Table 2 presents the set of training examples, M_2.

Table 2. The set of training examples, M_2.

Nr.	LI	GR	RT	Decision
1.	High	Medium	Very_short	Yes
2.	High	High	Medium	No
3.	High	Medium	Medium	Yes
4.	Medium	Low	Short	Yes
5.	Low	High	Very_short	Yes
6.	High	High	Long	No
7.	Low	High	Long	No

We have generated the following rules by using the ILA and DCL algorithms.

Rule 1

(ILA) IF LI = Medium THEN Decision = Yes;

(DCL) IF (LI = Medium) OR (GR = Low) OR (RT = Short) THEN Decision = Yes;

Rule 2

(ILA, DCL) IF RT = Very_short THEN Decision = Yes;

Rule 3

(ILA, DCL) IF RT = Long THEN Decision = No;

Rule 4

(ILA) IF LI = High AND GR = High THEN Decision = No;

(DCL) IF (LI = High AND GR = High) OR (RT = Medium AND GR = High)
THEN Decision = No;

Rule 5

(ILA, DCL) IF GR = Medium THEN Decision = Yes;

ILA has generated more simpler rules than DCL, as it can be seen the case of rules 1 and 4.

Table 3 shows the classification errors for two test sets with unknown examples, obtained after applying the algorithms ID3, ILA and DCL.

Table 3. Experimental results (the classification errors).

Test set	**ID3**	**ILA**	**DCL**
M_1	47.2%	29.7%	**25.3%**
M_2	34.8%	21.8%	**15.6%**

The best behavior was given by the DCL algorithm and the ILA algorithm. All the experimental results are also dependent on the nature of the data from the test sets.

Conclusion

The application of a data mining approach in knowledge base development involves a set of techniques for searching through data sets, looking for hidden correlations and trends which are inaccessible using conventional data analysis techniques [11]. In the recent years, the DM techniques were applied to the knowledge extraction step in the development of a KBS due to the improvements it can bring to the efficiency of knowledge base development. In this paper we have analyzed some of the induction techniques that are currently used to rule extraction, showing the results for two examples. As a general conclusion, the ILA and DCL algorithms has proved to be the best choice for rule extraction in a knowledge-based system.

References

[1] T. Mitchell, *Machine Learnin*g, McGraw-Hill, 1997.

[2] R. Milne, M. Drummond, and P. Renoux, Predicting Paper Making Defects On-line Using Data Mining, *Studies in Informatics and Control*, Vol. 6, No. 4, December 1997, 329-337.

[3] Al. Büchner, S. Anand, and J. Hughes, Data Mining in Manufacturing Environments: Goals, Techniques and Applications, *Studies in Informatics and Control*, Vol. 6, No. 4, December 1997, 319-327.

[4] J. R. Quinlan, *C4.5: Programs for machine learnin*g, Morgan Kaufmann, 1993.

[5] J. R. Quinlan, Learning logical definitions from relations, *Machine Learning*, Vol. 5, 1990, 239-266.

[6] P. Cheeseman, and J. Stutz, Bayesian classification (AutoClass): Theory and results, *Advances in knowledge discovery and data minin*g, MIT Press, 1995, 153-180.

[7] D. H. Fisher, Iterative optimization and simplification of hierarchical clusterings, *Journal of Artificial Intelligence Research*, vol. 4, 1996, 147-178.

[8] L. De Raedt, and L. Dehaspe, Clausal discovery, *Machine Learning*, vol. 26, No. 2/3, 1997, 99-146.

[9] M. Tolun, and S. Abu-Soud, ILA: an inductive learning algorithm for rule extraction, *Expert Systems with Applications*, Vol. 14, 1998, 361-370.

[10] S. Abu-Soud, and M. Tolun, A Disjunctive Concept Learning Algorithm for Rule Generation, *Proceedings of the 17th IASTED International Conference Applied Informatics*, ACTA Press, 1999, 665-667.

[11] S. S. Anand, D. A. Bell, and J. G. Hughes, A General Framework for Data Mining Based on Evidence Theory, *Data and Knowledge Engineering*, Vol. 18, 1996, 189-223.