Methodology of Data Mining: Preprocessing, Extracting Knowledge, and Postprocessing

IVAN BRUHA
Department Computing & Software
McMaster University
Hamilton, Ont., L8S4K1
CANADA

Abstract

Real-world datasets processing becomes very attractive disciplines in artificial intelligence (AI) applications for both research and industry. It is commonly called *Data Mining* (DM) and *Knowledge Discovery in Databases* (KDD). Their goal is to extract pieces of knowledge from usually very large databases. They consists of a usually robust sequence of procedures that have to be carried out so as to derive reasonable and understandable results.

The crucial component of DM symbolizes an inductive process which induces the above pieces of knowledge; usually it is *Machine Learning* (ML). However, most of the machine learning algorithms require perfect data in a reasonable format. Therefore, some *preprocessing* routines as well as *postprocessing* ones should fill up the entire chain of data processing. This paper overviews and discusses the knowledge discovery process as a series of several steps which include preprocessing of data, machine learning itself, and postprocessing of the resulting knowledge induced.

Keywords: data mining, knowledge discovery in databases, machine learning, preprocessing, postprocessing, attribute selection, attribute mining, knowledge integration

1 Data Mining: An Overview

Data Mining (DM) tools process usually very large databases in a profound and robust way. Since data are collected and stored at a very large acceleration these days, there has become an urgent need for a new generation of robust software packages to extract useful information or knowledge from large volumes of data. Research in DM develops methods and techniques to process large data in order to receive a knowledge (which is hidden in these databases) that would be compact, more or less abstract, but understandable, and useful for further applications. The paper [1] defines data mining as 'a nontrivial process of identifying valid, novel, and ultimately understandable knowledge in data'.

Quite a few names for extracting a useful knowledge (models) from databases have been launched; e.g., knowledge extraction, data analysis, information discovery, knowledge discovery in databases (KDD), data mining. Some researches claim that data mining is a subset of KDD (see e.g. [14]), some declare that it is the opposite way [25], others postulate that these terms are equivalent. In this paper we will follow the last interpretation.

In our understanding, data mining or knowledge discovery points to the overall process of determining a useful knowledge from databases, i.e. extracting high-level knowledge from low-level data in the context of large databases. Data mining can be viewed as a multidisciplinary activity because it exploits several research disciplines of artificial intelligence (AI) such as machine learning, pattern recognition, expert systems, knowledge acquisition, as well as mathematical disciplines such as statistics, theory of information, uncertainty processing.

The input to a data mining process is a *database*, i.e. a collection (a set) of objects. An *object* (also called *case, event, evidence, fact, instance, manifestation, record, observation, statement*) is a unit of the given problem. Its formal description can be of either quantitative (numerical) or qualitative (symbolic) character. Their collections can be of various forms, too, e.g. vectors, lists, strings, graphs, ground facts.

The overall output of the knowledge discovery process is a collection of pieces of knowledge dug from a database. They should exhibit a high-level description in a particular language. Such a *knowledge base* (*model, concept description*) is usually represented by a set of production (decision) rules, decision trees, collection of prototypes (cases, representative exemplars), etc.

The entire chain of data mining consists of these steps:

(1) *Selecting the problem area*. Prior to any processing, we first have to find and specify an application domain, and to identify the goal of the data mining process from the customer's viewpoint. Also, we are to choose a tool for representing such a goal.

(2) *Collecting the data*. Next, we have to choose the tools for representing objects, and to collect data as the formally represented objects. If a domain expert is available, then the expert could suggest what fields (attributes, features) are the most informative. If not, then the simplest method is a 'brute-force' which indicates that we measure everything available and only think that the right (informative, relevant) attributes are among them.

(3) *Preprocessing of the data*. A data set collected is not usually appropriate for an immediate induction (knowledge acquisition); it comprises in most cases a noise, missing values, the data are not consistent, etc. Also, we should use any suitable method for selecting

and ordering attributes (features) according to their informativity.
(4) *Extracting pieces of knowledge*. Now, we are at stage of selecting a paradigm for extracting pieces of knowledge (e.g., statistical methods, neural net approach, symbolic/logical learning, genetic algorithms). We have to realize that there is no optimal algorithm which would be able to process correctly any database. Second, we are to follow the criteria of the end-user (e.g., interest in understanding the model extracted rather than its predictive capabilities). Afterwards, we apply the algorithm selected and derive (extract) new knowledge.
(5) *Postprocessing of the knowledge derived*. The knowledge extracted in the previous step could be further processed. We can evaluate the extracted knowledge, visualize it, or merely document it for the end user. Also, we may interpret the knowledge and incorporate it into an existing system, and check it for potential conflicts with previously induced knowledge.

It is worth mentioning here that we may return from any step of DM process to any previous step and change our decisions. Thus, this process involves significant iteration and represents a quite time-consuming mechanism with many loops. Most work has been done in step 4. However, the other steps are also important for the successful application of knowledge discovery in practice.

2 Data Collection and Preprocessing

(a) Choosing Tools for Object Representation

The input to a DM process is a database, i.e. a set of objects. An object as a unit of the given problem must be formally described as a collection of elementary descriptions. Therefore, we have to choose the tools for object representation. The most common is the attribute representation of objects. Elementary properties, called *attributes*, are selected on actual objects. An object is thus represented by a list of attributes and their values; each pair `attribute = value` is called a *selector*. E.g. here we have an *attribute list*:

```
[ hair = black & eyes = blue &
  height = 165 & salary = 65000 ]
```

Each attribute has its domain of possible values. We distinguish three basic types of attributes: symbolic (discrete, nominal, categorical) such as `colour`, continuous (numerical) such as `weight`, and structured ones (whose value domain has a tree-oriented graph structure). This categorization depends on the task to be processed.

(b) Mapping and collecting data

After selecting a proper representation, we choose the attributes to be measured on objects (either following a suggestion of a domain expert or using the 'brute-force' method). Also, we have to determine attribute names and names of their values. Data collected are thus mapped into a single naming convention and uniformly represented.

(c) Scaling large datasets

Almost all learning algorithms assume that data are in the main memory and pay no attention how the algorithm could deal with extremely large databases when only a limited number of data can be viewed. One possible solution is called *windowing*. About 10 to 20% objects are randomly selected from a database and a data mining algorithm is invoked. For the next step, only a small portion of wrongly classified objects from the rest of the database in processed, etc.

Quite a new possibility of solving the problem of extremely large databases is called *batch-incremental* mode of a learning algorithm. A large database is processed in relatively small subsets; this is a natural scenario especially, if extremely large databases are collected in the relative small dispatches. Hence, each dispatch (batch) is analyzed independently and thus several knowledge bases are generated for each batch. Then, a knowledge integration algorithm has to be called to combine or merge these knowledge bases.

(d) Handling noise and errors

They are generally two origins of errors. *External* errors are introduced from the world outside the system itself. *Internal* errors are caused by poor properties of the learning (data mining) system itself, e.g., poor search heuristics or preference criteria. If the system is deficient in some sense, then it will occasionally fail to predict correctly the class of some unseen objects. It may happen by either a system's limited mechanism or limited computational power; see e.g. [26].

External errors have the following four sources:

- *Random data errors*: Data often contain random or noisy components in it, caused by the inherent unpredictability of some events in nature. If we deal with noisy data we can exploit two techniques: transformation of data, or modification of a learning algorithm.
- *Other external errors* are caused by limited description language, incomplete description, limited amount of data, and intractable data.
- *Systematic errors* such as the calibration error can be processed as follows: use an independent knowledge source to detect errors, derive general conclusion from tests, explain how errors arose, and eliminate the errors.
- *An imperfect teacher*: It is known that the learner learns concepts according to the information in a database whose author (designer) is a *teacher*. Therefore, the learner 'copies' its teacher's way of looking and interpreting a real-world problem. Hence, if the teacher makes *systematic* errors there no way to find errors in the database designed by the teacher. However, if the teacher makes *temporary* mistakes, then the qualitative background knowledge can be used to validate the results presented by the imperfect teacher, or a preprocessor itself may observe the 'real world' and verify whether the information provided by the teacher is right.

(e) Processing unknown attribute values

When processing real-world data, one important aspect in particular is the processing of *unknown* (*missing*) attribute values. This topic has been discussed and analyzed by several researchers; see e.g. [4], [8], [12], [22]. The most important direction in this topic is the *source of 'unknownness'* [19]:

(1) a value is *missing* because it was forgotten or lost; (2) a certain attribute is *not applicable* for a given object , e.g., it does not exist for a given object; (3) an attribute value is *irrelevant* in a given context; (4) for a given object a designer does not care about a value of a certain attribute (*dont-care* value).

The paper [24] surveys and investigates quite a few techniques for unknown attribute values processing for the TDIDT (Top Down Induction Decision Trees) family and indicates seven best combinations of processing the missing attribute values. The paper [8] portrays the performance of six unknown value processing strategies in an algorithm employing the *covering* paradigm:

(i) ignore the example with unknown values;
(ii) consider the unknown value as an additional regular value for a given attribute; or
(iii) substitute the unknown value by a suitable value which is either the most common value, a proportional fraction, randomly selected value, or any value of the known values of the attribute that occurs in the training set.

(f) Discretization/fuzzification of numerical attributes and processing of continuous classes

The symbolic, logical learning algorithms are able to process symbolic, categorical data only. However, real-world problems involve both symbolic and numerical attributes. Therefore, there is an important issue in DM to discretize numerical (continuous) attributes.

The task of discretization of numerical variables is well known to statisticians. Different approaches are used, for instance, discretization into a given number of categories using equidistant cutpoints, or categorization based on mean and standard deviation. All these approaches are 'class-blind', since they do not take into account that objects belong to different classes.

Most of the newer learning algorithms can also deal with numerical data. They are 'class-sensitive', which means that the procedures perform the discretization according to the class-membership of the training objects.

In the TDIDT family (ID3, C4.5), the algorithms for discretization are mostly based on *binarization* within a subset of training data created during tree generation [23]. A domain of a certain numerical attribute is split into two intervals by calculating the threshold according to a certain statistics. An improvement of the above binarization methods is *recursive binarization* [11]; the domain of a numerical attribute is first binarized into two intervals; afterwards, each of these intervals is recursively binarized to subintervals, until a certain stopping condition is satisfied. [2] introduces an *iterative discretization*. It exploits a procedure for splitting domains of numerical attributes generally to more than two intervals.

A discretization procedure can be invoked either *off-line* (numerical attributes are discretized by a preprocessor, before the actual inductive process) or *on-line* (within the inductive algorithm), see, e.g. [9].

In many application areas, discretization of numerical attributes into crisp intervals does not correspond to real situations. Small difference in the value of an attribute cannot completely change the class of an object. Thus, it seems more realistic to *fuzzify* the numerical ranges into fuzzy intervals.

Similar to the above issue is the processing of continuous classes. Most inductive symbolic algorithms process the discretized (symbolic) classes. In some applications, however, we are faced by a continuous (numerical) classes. Also, there exist two fundamental approaches. The first one splits the domain of a continuous class to a set of intervals by a preprocessor, i.e. *off-line* splitting. The other approach performs the splitting *on-line*, i.e. during the inductive process.

In both cases, the simpler versions replace such an interval by $m \pm \sigma$, where m is the mean of the class values on the given interval, and σ is the variance. More sophisticated versions allow an expression to be a regression function of continuous attributes, see e.g. Retis [17] and HTL [27].

(g) Grouping of values of symbolic attributes

It is a known problem that attributes exhibiting too many values are overestimated within the process of selecting the most informative attributes, both for inducing decision trees [12] and decision rules [3], [10]. To overcome this overestimation, the values of multivalued attributes are grouped into two subsets that maximize attribute informativity. Hence, this *binarization* normalizes the informativity of all attributes with respect to the numbers of values.

(h) Attribute mining

If we employ the recommendation of a domain expert or just use the 'brute-force' method for extracting attributes for a given task, we cannot be definitely sure that they are the informative ones for the given target. In real-world data, the representation of data often exhibits too many attributes, but only a few of which may be related to the target concept. Consequently, *attribute mining*, which consists of attribute selection, ordering, construction, and transformation, is a very useful process.

Attribute selection and *ordering* procedures order the entire set of (input) attributes according to their informativity, and select a subset of the most informative attributes, some of them are the input attributes but some could be defined as a mapping of existing input ones. Well-known is Karhunen-Loeve method, see e.g. [15]. Recently, the algorithm Relief [18] and several its extensions [20] for attribute selection has been designed.

If attributes are inappropriate for the target concept, a data mining system needs to be capable of generating (constructing) new appropriate attributes. This is done by so-called *attribute construction*. Given a set of existing attributes and a set of so-called constructive operators (a part of the background, domain-specific knowledge of the given problem), the attribute construction results in producing a set of new attributes. Various systems differ in the way how they search in the space of all possible new attributes; see, e.g. the system Struct [28], or ID2-of-3 [21].

Another approach, *attribute transformation*, decomposes the original set of training examples to several smaller subsets and generates hierarchical attribute trees whose nodes are new (intermediate) attributes; see, e.g. the algorithms HINT [13], [29].

(i) Consistency checking

Inconsistency of data may not be eliminated by the previous steps of preprocessing. A typical case of inconsistency comprises two or more identical objects (with the same attribute values) that belong to different classes (concepts). There are two general approaches to handle the inconsistency of data. The first one is done off-line, i.e. by a preprocessor (usually by removing the inconsistent data), or within the DM process itself. Another possibility is to utilize the loop facility of the knowledge discovery process, i.e. to return to one of the previous steps and perform it again for different parameters.

3 Extracting Knowledge

The input to this step of DM is the database of preprocessed objects (examples), optionally with a background knowledge, and its output is an extracted knowledge. The term *data mining* has mostly been used by statisticians, data analysts, and the management information systems. The researchers working in AI have been using the term *machine learning* for such activities. In fact, both terms overlap and the difference between the above two disciplines has not been precisely resolved.

The paradigms for machine learning have been discussed in literature in a deep and precise fashion for many years. Therefore, we will not go in this survey paper to details nor to list the references to huge amount of research papers devoted to these paradigms. We will just briefly introduce them:

- Symbolic/logic learning methods involve the divide-and-conquer paradigm that generates decision trees, e.g. TDIDT family (ID3, C4.5), or the covering paradigm that induces a set of decision rules, e.g. AQx and CNx families.
- Case-based methods extract representative examples (prototypes) from the database to approximate the knowledge "hidden" in the database, or they derive new prototypes (not appearing in the database). These techniques include the nearest-neighbour methods and case-based reasoning.
- Statistical methods encompass various techniques including discriminant-function (parametric) methods, non-parametric (distribution-parameters-estimate) methods, linear and nonlinear regression, Bayesian methods, cluster methods, etc. They mostly process numerical-based databases.
- Neural nets are applied to numerical data, and include various topologies and a wide variety of learning techniques.
- Genetic algorithms emulate the biological evolution, and are utilized for optimization processes, similarly to simulated annealing. There are quite a few attempts to incorporate the genetic algorithms into machine learning.

Another taxonomy of learning is based on the way of processing of training examples:

- *Batch (one-trial) mode* assumes all training examples to be presented to the learner at once since all of them are required for acquiring a concept description.
- In *incremental (sequential) mode*, the learner reads training examples subsequently; it forms one or more hypotheses of the concept, and gradually refines these hypotheses when reading the next available example.

Let us define the entire problem more formally. Let X be the space of all objects of the given task. A *concept (class)* can be characterized as a set $C \subseteq X$ of objects having common properties in the given task. A *training example (learning example, training data)* is actually a pair

`[ object, desired_concept ]`

Both object and its desired class (concept) are provided by a teacher (designer of the project). A finite set $T \subseteq X$ of training examples given to the learner is called *training set (learning set)*. A *positive* example of a concept is an object belonging to this concept, a *negative* example is not.

Besides an object description language, the specification of a learning task has to comprise a language for concept descriptions (hypotheses, models, knowledge bases). Such a language is of types similar to those that can be used for representing knowledge in general. There exist various formalisms, but decision trees and sets of decision rules are mostly used in machine learning.

The problem of learning a concept C from examples can be specified as follows: Given a finite training set $T \subseteq X$, find an expression (formula) `Descr` (called *concept description*, *concept hypothesis*, or *inductive assertion, model*) in a concept description language, such that

- the concept description is *complete*, i.e. each positive example of the concept (class) C in T matches `Descr`, and
- the concept description is *consistent*, i.e. no negative example in T matches `Descr` .

4 Knowledge Postprocessing

(a) Knowledge filtering: Rule truncation and post-pruning

Since most real-world datasets are noisy, a learning algorithm generates leaves of a decision tree or decision rules that cover a very small number of training examples. It happens because a learning algorithm tries to split subsets of training examples to even smaller subsets that would be genuinely consistent (i.e., would contain examples of one class only). This issue is commonly called *overfitting*.

Hence, a decision tree or a set of decision rules induced without considering noise could be too large (i.e., containing a large number of nodes or rule conditions) and thus hard to understand, and (what is much worse) its leaves or rules would be supported by relatively small sets of training data. The leaves or decision rules formed on the basis of fewer examples reflect small trends in the training set, are more suspectable to noise, and thus have less predictive power (i.e. unreliably classify unseen objects). Indeed, it is better to create a leaf for a larger training set even with a few negative examples rather than to split the training set into small subsets of examples of one class only.

Consequently, the constraint to the absolute consistency of a class (concept) description should be relaxed for noisy data, i.e., we should abandon the

requirement that the decision tree or a set of decision rules induced by a learner classify all training data correctly. The underlying idea is quite straightforward: we hope that the misclassified training examples are those that comprise noise. It happens very often that a slightly inconsistent concept description is more predictive.

Postpruning of decision trees goes along a decision tree from the root down towards leaves. Being at a certain node it decides according to a criterion whether to retain this node as it is, or to remove the subtree under this node and, thus, change it to a leaf. It is worth noticing that the inductive algorithm itself can perform pruning during the process of constructing a decision tree; it is called *prepruning (forward pruning)*.

Truncation of rules simplifies a set of decision rules usually in two steps. First, each rule is simplified separately by dropping irrelevant selectors. Second, it eliminates any rule that is either a subset of another rule, or a rule which would decrease the overall performance of the model (knowledge base) only by a negligible percentage, or a rule that is below a certain quality threshold; see, e.g. [23].

(b) Interpretation and explanation

Now, we may use the acquired knowledge directly for prediction or in an expert system shell as a knowledge base. If the knowledge discovery process is performed for an end-user, we usually document the derived results. Another possibility is to visualize the induced knowledge, or to transform it to an understandable form for the user-end. Also, we may check the new knowledge for potential conflicts with previously induced knowledge.

In this step, we can also summarize the rules and combine them with a domain-specific knowledge provided for the given task.

It should be noticed that especially expert system applications should be accompanied by the explanation facility for an end-user to accept the decision of the expert system.

(c) Evaluation

After a learning system induces concept hypotheses (models) from the training set, their *evaluation* (or *testing*) should take place. There are several widely used criteria for this purpose:

- *Classification accuracy* is usually defined as the percentage of unseen objects correctly classified by the induced concept (class) descriptions. Alternatively, *classification error rate* can be defined as its complement. There are in fact three schemes for evaluation of the knowledge induced:
  - The designer of the task provides both training and *testing* sets that should be disjoint. The acquired knowledge base is then evaluated on the testing set provided, and the result of evaluation is the classification accuracy measured on this testing set.
  - If there is just one dataset, then we can *split* it randomly to a training set (say 70% size) and testing set (30%) and measure the classification accuracy on the testing set; this procedure is to be repeated several times.
  - The *N-cross-validation* seems to be the most favourite scheme. The original data are partitioned randomly into *N* subsets; each subset is used for testing whereas the remaining *N-1* subsets are used for learning. This learning and testing process is thus done *N* times, and the classification accuracy is then calculated as the average accuracy from N^2 runs.

  Some evaluation schemes do not use the above, rather simple definition of classification accuracy as the percentage, but so-called *loss matrix*, supplied by the end-user. Each element of this matrix expresses the magnitude of loss (error) which arises if an object belonging to one class is actually classified to a different class.
- *Comprehensibility* or understandability of the induced knowledge is very important to exhibit interesting issues about the application domain. From this viewpoint, symbolic paradigm is a winner in competing with e.g. neural nets (since the weights in a neural network do not express anything comprehensible). This subjective evaluation is performed by asking a human expert to classify objects of the given task by using the induced knowledge [16].
- *Computational complexity* comprises the memory size for storing induced concept descriptions, the runtime speed for learning and for classification, etc.

Also, it is worth noticing that classification could be accompanied by some conflict situations, particularly when several rules belonging to different classes match a tested (unseen) object. One of the possible solutions to this conflict is to associate each decision rule induced with a numerical factor which can express its properties and characterize a measure of belief in correctness of the rule, its power, predictability, reliability, likelihood, etc. A collection of these properties is symbolized by a function commonly called the *rule quality*. A survey of formulas for rule quality can be found e.g. in [6], [7].

(d) Knowledge integration

The traditional decision-making systems have been dependant on a single technique, strategy. Therefore, their accuracy and successfulness is not usually too high. New sophisticated decision-supporting systems utilize results obtained from several models, each usually (but not required) is based on different paradigm, or combine or refine them in a certain way. Thus, such a multistrategy (hybrid) system consists of two or more individual 'agents' which interchange information and cooperate together. The first project in this subject can be found in [5].

Knowledge integration, a relatively new field of AI, is focusing on three scenes:

- *Knowledge Modification (Revision)*. The input is an existing knowledge base (model) `KB` and a new database `DB`. A data mining algorithm revises (modifies) the current knowledge base `KB` according to the knowledge hidden in the current database `DB`. The new knowledge base (model) `KB1` thus exhibits the "old" knowledge `KB` updated on the basis of a new information extracted from the database `DB`.
- *Knowledge Merging*. The input to this system are several knowledge bases (models) `KB1`,...,`KBn` generated by several algorithms. The output is a single knowledge base (model) which arises by merging the input knowledge bases.
- *Knowledge Combination*. This system has at its

disposal several knowledge bases (models) KB1,...,KBn generated by several data mining algorithms. These bases remain independent. Each knowledge base builds its decision about prediction/classification independently. The results of these models are then combined in order to produce the final decision about the class/prediction. In this system, the mechanism of quality of knowledge bases is usually exploited.

5 Conclusion

This paper surveys DM as a sequence of several steps including preprocessing and postprocessing. If an inductive technique is to be successfully applied in a real-world domain, the raw data has to be preprocessed that the data analysis be performed correctly and efficiently. Data preprocessing also helps the designers to understand better the nature of the data and the process.

Postprocessing is also important for the success of DM techniques. The raw output of a data mining algorithm has several drawbacks: the rules induced are not of equal importance and quality, and also, there is always a possibility that the end-user is not interested in some irrelevant rules. Postprocessing thus particularly assists the end-users to interpret, validate, and apply the rules in a better fashion.

There are, however, several other issues we have not mentioned in this paper; e.g. uncertainty, redundancy, overgeneralization, brittleness, learning from a very few training examples, irrelevant formal descriptions, representativeness of training examples, discovery of underlying principles. As for the above topics, we ask the reader to a large list of papers and monographs that (because fo space limitations) cannot be introduced here.

References:

[1] R. Agrawal, G. Psaila, "Active data mining", *1st International Conference on Knowledge Discovery and Data Mining*, Menlo Park, Calif., 1995, pp. 3-8

[2] P. Berka, I. Bruha, "Various discretizing procedures of numerical attributes: Empirical comparisons", *8th European Conf on Mach. Learning, Workshop Statistics, Mach. Learning, and Knowledge Discovery in Databases*, Heraklion, Crete, 1995, 136-141

[3] P. Berka, I. Bruha, "Discretization and grouping: Preprocessing steps for data mining", in: J.M. Zytkow, M. Quafafow (eds.), *Principles of Data Mining and Knowledge Discovery*, Lecture Notes in AI, Springer, 1998, pp. 239-245

[4] P.B. Brazdil, I. Bruha, "A note on processing missing attribute values: A modified technique". *Workshop on Machine Learning, Canadian Conference on AI*, Vancouver, 1992

[5] P. Brazdil, L. Torgo, "Knowledge Acquisition via Knowledge Integration", in: *Current trends in Knowledge Acquisition*, IOS Press, 1990

[6] J. Tkadlec, I. Bruha, "Formal aspects of a multiple-rule classifier", *International Journal of Pattern Recognition and Artificial Intelligence*, **17**, 4 (2003), 581-600

[7] I. Bruha, "Quality of decision rules: Definitions and classification schemes for multiple rules", in: G. Nakhaeizadeh, C.C. Taylor (eds.), *Machine Learning and Statistics: The Interface*, John Wiley, 1996, pp. 107-131

[8] I. Bruha, F. Franek, "Comparison of various routines for unknown attribute value processing: Covering paradigm", *Intern'l Journal of Pattern Recognition and Artif.Intelligence*, 10, 8, 1996, 939-955

[9] I. Bruha, P. Berka, "Numerical attributes in symbolic learning algorithms: Discretization and fuzzification", *2nd International Conference Neural Networks and Expert Systems in Medicine*, Plymouth, UK, 1996, pp. 131-138

[10] I. Bruha, S. Kockova, "A support for decision making: Cost-sensitive learning system", *Artificial Intelligence in Medicine*, 6, 1994, pp. 67-82

[11] J. Catlett, " On changing continuous attributes into ordered discrete attributes", *EWSL-91*, Porto, Springer-Verlag, 1991, pp. 164-178

[12] B. Cestnik, I. Kononenko, I. Bratko, "Assistant 86: A knowledge-elicitation tool for sophisticated users", in: I. Bratko, N. Lavrac (eds.), *Progress in Machine Learning*: EWSL'87, Sigma Press, 1987

[13] J. Demsar et al., "Constructing intermediate concepts by decomposition of real functions", *ECML-97*, Prague, 1997, 93-107

[14] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, "From data mining to knowledge discovery in databases", *AI Magazine*, 1996, pp. 37-53

[15] K. Fukunaga, W.L.G. Koontze, "Application of the Karhunen-Loeve expansion to feature selection and ordering", *IEEE Trans. on Computers*, C-19, 4, 1970, pp. 311-318

[16] R.J. Henery et al., "Statlog: Comparative testing of statistical and machine learning algorithms". *Techn. Rep.*, Strathclyde Univ., Glasgow, 1991

[17] A. Karalic, "Employing linear regression in regression tree leaves", *ECAI-92*, John Wiley, 1992, pp. 440-441

[18] K. Kira, L.A. Rendell, "A practical approach to feature selection", *ICML-92*, 1992, pp. 249-256

[19] I. Kononenko, "Combining decisions of multiple rules", in: B. du Boulay, V. Sgurev (eds.), *Artificial Intelligence V: Methodology, Systems, Applications*, Elsevier Science Publ., 1992, pp. 87-96

[20] I. Kononenko, E. Simec, M. Robnik-Sikonja, "Overcoming the myopia of inductive learning algorithms with RELIEFF", *Applied Intelligence*, 7, 1997, pp. 39-55

[21] P.M. Murphy, M.J. Pazzani, "ID2-of-3: Constructive induction of M-of-N concepts for discriminators in decision tree", *ICML-91*, 1991, pp. 183-187

[22] J.R. Quinlan, "Induction of decision trees", *Machine Learning*, 1, 1986, pp. 81-106

[23] J.R. Quinlan, "Simplifying decision trees", *International J. Man-Machine Studies*, 27, 1987, pp. 221-

[24] J.R. Quinlan, "Unknown attribute values in ID3", *International Conference ML*, 1989, 164-168

[25] Stolorz et al., "Fast spatio-temporal data mining of large geophysical datasets", *1st International Conference on Knowledge Discovery and Data Mining*, Menlo Park, Calif., 1995, pp. 300-305

[26] P.V. Tadepulli, "Learning in intractable domains", in: T.M. Mitchel, J.G. Carbonell, R.S. Michalski (eds.), *Machine Learning: A guide to current research*, Kluwer, 1986

[27] L. Torgo, "Functional models for regression tree leaves", *ICML-97*, Morgan Kaufman, 1997

[28] L. Watanabe, L. Rendell," Feature construction in structural decision tree", *ICML-91*, 1991, 218-222

[29] B. Zupan et al., "Feature transformation by function decomposition", *IEEE Expert*, 1998