Basic principles of cognitive algorithms design

Kristina Machova
Department of Cybernetics and Artificial Intelligence
Technical University of Kosice
Letná 9, 042 00 Kosice
Slovakia
Kristina.Machova@tuke.sk

Jan Paralic
Department of Cybernetics and Artificial Intelligence
Technical University of Kosice
Letná 9, 042 00 Kosice
Slovakia
Jan.Paralic@tuke.sk

Abstract – This paper focuses on basic principles of cognitive algorithm design. It presents various principles employed in known cognitive algorithms dealing with different representations of concept definitions. It discusses pros and cons of application of the presented basic principles in relation with classification task, which is one of the most often used data mining approaches in various application domains. This paper focuses also on systematization of algorithm design. A problem (cognitive task) is analyzed and those of basic principles are selected, witch match the given cognitive task requirements. Finally, a generic procedure for composition of a suitable cognitive algorithm for particular classification task is defined.

I. INTRODUCTION

We can see at least too main business motivations for research of cognitive algorithms, which are able to learn some kind of knowledge from data. Firstly, quality of a knowledge-based system [6] used by a company is dependent on the quality of knowledge, which is used by the system. Secondly, knowledge discovered at the right time may yield a high competitive advantage. Therefore, a high degree of attention is devoted to research of knowledge acquisition approaches.

A rich set of knowledge acquisition techniques comes from machine learning employing different cognitive algorithms. Cognitive algorithms can be of various kinds (e.g. inductive, deductive, incremental, non-incremental), they can perform learning with teacher or without teacher, with attention to solve cognitive tasks of various kinds. Moreover, they can use various representations of learned knowledge. Therefore, a large number of various machine learning techniques has been discovered and is being further researched and improved. But, there is much less information available with respect to their common basic principles and methodology how to select a suitable of learning algorithm for particular task. This paper is a contribution in this direction.

We will assume, that the input of a cognitive algorithm has the form of a set of training examples. The algorithm produces a concept as its output – we expect to obtain a definition of that concept as well. This definition can be of different sorts. In case of a classification task, the definition can often be in the form of a classification rule. The concept definition will contain attributes with appropriate values – conditions whose meeting will be sufficient for the classification of a new example into a class represented by the given concept.

The rest of this paper is organized as follows. Next section provides description of the process of knowledge discovery in databases. The third section is focused on description of identified basic principles of classification cognitive tasks. Fourth section proposes a generic procedure for composition of a suitable cognitive algorithm for particular classification task.

II. DATA MINING

Knowledge discovery in databases (KDD) can be defined as nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. According to [2] it is an interactive and iterative process with several steps. It means that at any stage the user should have possibility to make changes (for instance to choose different task or technique) and repeat the following steps to achieve better results. Data mining is a part of this process.

In most of sources, the term Data Mining (DM) is often used to name the field of knowledge discovery. This confusing use of terms KDD and DM is due to historical reasons and due to fact that the most of the work is focused on refinement and applicability experiments of machine learning algorithms from artificial intelligence for the data-mining step. Pre-processing is often included in this step as a part of mining algorithm.

Within the KDD process following steps can be recognized [3].

1. *Data cleaning* - removal of noisy and inconsistent data
2. *Data integration* - data from multiple data sources may be combined
3. *Data selection*, where data relevant to the analysis task are retrieved from database (data warehouse)
4. *Data transformation* - data are transformed or consolidated into forms appropriate for mining
5. *Data mining* –core of the KDD process, where intelligent methods are applied in order to extract data patterns (explicit form of knowledge)
6. *Pattern evaluation* – to identify interesting patterns
7. *Knowledge representation* - visualization of mined knowledge.

There are various types of data mining, e.g. descriptive data mining (trying to find a compact description of a specific subgroup of data), or predictive data mining (building a classification or prediction model from a training data set and using it for predicting class or target attribute for new instances of data).

One of the most often used data mining tasks is classification with plenty of possible algorithms to be applied.

III. BASIC PRINCIPLES

The basic principles, which can be found in different cognitive algorithms, can be divided into characteristic and aided principles. **Characteristic principles** are: ordered version space, hill-climbing principle, division of example space into subspaces, control with exceptions and competitive principle. **Aided principles** are: score function and reduction of the number of versions. Aided principles obviously are combined with characteristic principles. Combinations between various characteristic principles are also possible. Possible combinations of all basic principles are presented (dash line) in Figure 1, which presents a kind of ontology of basic principles.

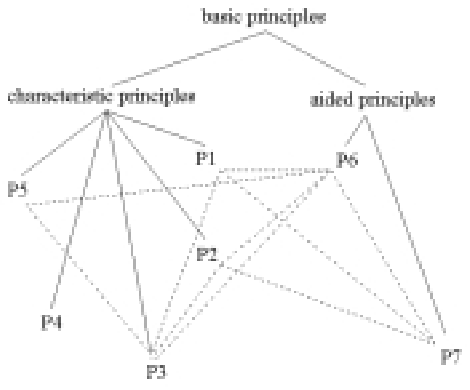

Figure 1: Basic principles ontology. (P1 - ordered version space, P2 - hill climbing principle, P3 - division of example space into subspaces, P4 - control with exceptions, P5 - competitive principle, P6 - score function, P6 - reduction of the number of versions).

Algorithms, in which these principles are used, will be presented. Detailed descriptions of these algorithms can be found in [5] and [8].

A. Ordered version space

One of the possibilities how to find a concept definition is to create a space of all concept versions and to search for a solution in this space. The search will be more effective when concept versions will be ordered according to some criterion. Generality is the most used criterion for this ordering. We can use three kinds of search through the ordered version space: *from general to specific, from specific to general, and parallel search combining both directions.* This principle is used e.g. by the following algorithms: VSS (Version Space Search) [7], EGS (Exhaustive General to Specific), and ESG (Exhaustive Specific to General).

B. Hill-climbing principle

This principle is based on investigation of the surroundings of an actual solution and the solution is moved into the best point found in the neighborhood of the actual solution. The process is repeated until the sub-optimal solution is found. Examples of algorithms using this principle are e.g. IWP (Iterative Weight Perturbation), SOMA [7].

C. Division of example space into subspaces

The principle of division of the example space into subspaces was one of the first principles used in the field of machine learning. This principle is known as „divide and conquer“. It is based on division of the example space into subspaces until the finishing condition is met, for example till each subspace contains only examples of the same class. The division into subspaces is based on conditions defined by means of information theory. The principle is used e.g. in the following algorithms: NSC (Non-incremental Separate and Conquer), AQ11, ID3, ID5R, C4.5, and MDPL.

D. Control with exceptions

Control with exception means that training examples are classified into a set of defined classes within each algorithm iteration. New classes are created for incorrectly classified examples – exceptions. This process is repeated until all exceptions are correctly classified. The principle of control with exceptions is used in NCD (Non-incremental Induction of Competitive Disjunctions), ICD (Incremental Induction of Competitive Disjunctions), and NEX (Non-incremental Induction of Decision List with Exceptions).

E. Competitive principle

Competitive principle is based on concepts candidates' evaluation according to some score function. The concept candidate evaluated as the best one is selected. For example such score function can be probability of class (concept candidate), or distance of a classified example from prototypes of different classes.

This principle is used by Bayes classifier and by cognitive algorithms for prototypes generation: NCD and ICD [5].

F. Score function

If we try to find the optimal solution, then we have to search through the whole version space. We must do exhaustive search, which is slow and demanding on time and memory resources. That is why we try to find out those concept versions that enable us to solve the given problem in the shortest way. We will use search bias – we will investigate some versions before the others. We will have to define a score function, which assigns the highest value to the best version. Generally, the value of the score function increases with P_c (number of covered positive training examples) and N_{nc} (number of uncovered negative training examples). A good score function considers the whole number of positive and negative examples. More sophisticated functions use statistical or information measures, for examples entropy. The principle of score functions is used in the algorithms: HGS (Heuristic General to Specific), HSG (Heuristic Specific to General), and HCT (Heuristic Criteria Tables), and also in some algorithms for decision trees generating (e.g. ID3 [9], ID5R and C4.5 [10]) and decision lists (e.g. CN2 [1]).

G. Reduction of the number of concept versions

When we give up the optimal solution requirement, we can limit the number of considered concepts in each iteration of a learning algorithm. So we reduce the number of concept versions considered during version space search. We can define this number before the learning algorithm starts as its parameter BS - Beam Size. As a rule the Beam Size is combined with a score function and they are presented as heuristic approach to learning. Concept versions are ordered according to values of the score function and only BS best versions are selected in each iteration. This heuristic approach is a compromise between precision and effectiveness – it does not guarantee to find the optimal solution, but always is able to find some quite well solution in real time. That principle is used in the following algorithms: HGS, HSG, and HCT.

The various combinations of basic principles in known cognitive algorithms are presented in Table 1. (P1 is ordered version space, P2 is hill-climbing principle, P3 is division of example space into subspaces, P4 is control with exceptions, P5 is competitive principle, P6 is score function, P7 is reduction of the number of versions.)

TABLE I

COMBINATION OF BASIC PRINCIPLES IN COGNITIVE ALGORITHMS

	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**
VSS	*						
EGS	*						
ESG	*						
HGS	*					*	*
HSG	*					*	*
HCT						*	*
IWP		*					
SOMA		*					
NSC			*				
AQ11			*				
ID3			*			*	
ID5.R			*			*	
C4.5			*			*	
MDPL			*				
NCD				*	*		
ICD				*	*		
NEX				*			
CN2						*	

IV. COGNITIVE ALGORITHMS DESIGN

The quality of learning algorithm design depends on the suitable selection of one or more basic principles. These basic principles are selected according to the kind of learning task, which should be solved by the algorithm.

Complicated learning task with large number of training examples can be solved by using ordered version space search with the aid of score function and reduction of the number of concept versions. This basic principles combination is suitable for learning tasks with short time available for solution and if optimal solution is not necessary.

Principle of control with exception can by used for task represented by noisy training data.

A learning task characterized by very unbalanced distribution of examples into classes can be solved using principle of the division of example space into subspaces.

A hill-climbing principle is suitable for learning task with linear separability.

Untraditional combinations of basic characteristic principles are also possible. For example combination of using division of example space into subspaces with using ordered version space search in all subspaces.

We propose the following generic procedure for composition of suitable cognitive algorithm:

- Analysis of the given data set and DM task.
- Selection of a set of suitable basic principles for the given DM task.
- General algorithm design, based on selected basic principles.
- Selection of a suitable programming language.
- Algorithm implementation in selected language.

Proposed generic procedure can effectively be supported by KDD software tool that integrates also various algorithms for classification (e.g. [4]).

V. ACKNOWLEDGMENT

This work has been done within the VEGA project 1/8131/01 "Knowledge Technologies for Information Acquisition and Retrieval" of Scientific Grant Agency of Ministry of Education of the Slovak Republic.

VI. REFERENCES

[1] Clark, P., Nibblet, T.: The CN2 Induction Algorithm. Machine Learning, Vol.3, No.4, 1989, 261-284.

[2] Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: The KDD Process for Extracting Useful Knowledge from Volumes of Data. Comm. of the ACM, Vol.39, No.11, 1996, 27-34.

[3] Han, J. and Kamber, M. (2000). Data Mining – Concepts and Techniques. Morgan Kaufmann Publishers, 2000.

[4] KDD Package with documentation: http://neuron.tuke.sk/~paralic/KDD/

[5] Langley, P.: Elements of Machine Learning. Morgan Kaufmann Publishers, Inc. San Francisco, California, 1996, 419.

[6] Mach, M.: Knowledge acquisition for knowledge-based systems. Elfa, Košice, 1997, 104.

[7] Mitchell, T.M.: Version spaces: A candidate elimination approach to rule learning. Proc. of the 5th Int. Joint Conference on Artificial Intelligence, Cambridge, MA: Morgan Kaufmann, 1977, 305-310.

[8] Mitchell, T.M.: Machine Learning. The McGraw-Hill Companies, Inc. New York, 1997b, 414.

[9] Quinlan, J.R.: Induction of decision trees. Machine Learning 1, 1, 1986, 81-106.

[10] Quinlan, J.R.: C4.5 Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, California, 1993, ISBN 1-55860-238-0.