Proceedings of AAAI Workshop on Knowledge Discovery in Databases'94, Seattle, pp 237-249

# PolyAnalyst - A MACHINE DISCOVERY SYSTEM INFERRING FUNCTIONAL PROGRAMS.

*Mikhail V.Kiselev*
*Computer Patient Monitoring Laboratory at National Research Center of Surgery, per.Abrikosovski, 2, Moscow 119874, RUSSIA*
*e-mail: geography@glas.apc.org*

A B S T R A C T

The present paper describes the learning technique used in PolyAnalyst - the system of machine discovery and intelligent analysis of the experimental/ observational data which has been created in the Computer Patient Monitoring Laboratory at the National Research Center of Surgery. PolyAnalyst is a multi-purpose system designed to solve the following classes of problems: 1) construction of a procedure realizing the mapping from the set of descriptions to the set of parameters given by the pairs <description, parameter>; 2) search for the interdependences between components of the description; 3) search for characteristic features of a given set of descriptions. Here the description means a single experimental/ observational data record and it is assumed that all the records in the same set of observations have the same structure. The paper is devoted mainly to PolyAnalyst's application to the type 1 problems. This type includes classification, empirical law inference, choice of the best decision from a fixed set of possible decisions, and other tasks. To solve a problem PolyAnalyst constructs and tests programs on a simple functional programming language whose inputs are the descriptions and outputs are the corresponding parameter values. While searching for the solution PolyAnalyst combines full search, heuristical search, and direct construction of the programs. Since the first version of PolyAnalyst was created it has solved a number of real problems from chemistry, medicine, geophysics, and agricultural science. One example is given in the paper - the prediction of the elasticity of the polyethylene samples from their infra-red spectrums.

KEYWORDS: machine discovery, automated knowledge acquisition.

## 1. Introduction

Since 60s when machine learning began existing as a science the great diversity of methods and systems has been proposed varying in the form of training examples, the degree of autonomy, the use of existing formalized knowledge about application domain, the form of learned information, and other features. However the "density" of research is not the same over all the range of the machine learning tasks. In particular this density is still low in the

fields where the problems of recognition of the complex multi-factor interdependences in the bodies of the experimental/observational data should be solved under condition of sparse or absent model knowledge and under minimum guidance from a human. It is especially true for the cases, when the structural properties are important and the possibility to perform desired experiments is limited. The main difficulty of these problems is determined by the fact that since the type of sought dependence is not known a priori the solution has to be sought in the space of procedures realizing various dependences. In this situation the possibility to construct the solution directly is limited, therefore the search in very wide spaces should be used. Although on the other hand the search in the sets of procedures can be combined with the use of more effective methods in some their wide subsets. For example, they may be the sets of procedures equivalent to the predicate calculus formulae or to the rational functions.

The present research is devoted to use of methods based on the search in the space of procedures for solution of real problems of the empirical law inference and machine learning. Its aims include the effective search strategies, the principles of selection of the best search directions, and the methods of the direct synthesis of desired procedures.

The last years are marked by significant increase of research activity in the field of automated knowledge acquisition, machine discovery, and automated inference of models. The present paper does not include a thorough literature review - it would require a lot of space. Good review of research before 1989 can be found in (MacDonald, Witten, 1989). There exists a number of systems which infer the dependences between variables constituting training examples (as a rule in the form of rational functions). The most famous ones are BACON (Langley, Zitkov, Bradshaw, Simon, 1986), ABACUS (Falkenhainer, Michalski, 1990), FAHRENHEIT (Zitkov, Zhu, 1991). The description of another interesting machine discovery technique can be found in (Phan, Witten, 1990). There are various approaches which use the extensions of the classic declarative formalisms to achieve more expressive power (see, for example, extension of the predicate calculus in (Michalski, 1993)). The system ARE which synthesizes functional programs in the process of search for the solution is described in (Shen, 1990). The problem of discovery of structural regularities in data is close to the grammar inference problem - see, for example, (Wolff, 1987). At last many works are devoted to the theoretical aspects of problem of the recursive functions inference (Jantke, 1987). Summarizing it should be noted that all these works touch only different instances of the general problem of the program synthesis as the universal approach to learning from examples

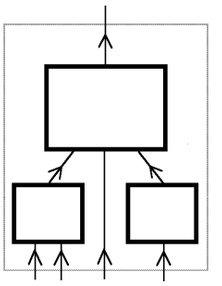
and machine discovery. Besides that only few systems (for example, FAHRENHEIT) are reported to have the real-world applications. The present paper is an attempt to show that the search in the space of procedures built from the domain-specific sets of primitives with the help of the universal production schemes can serve as a basis for a universal data analysis system capable of solving the real problems from the various fields.

The paper is organized in the following way. Section 2 contains the description of the functional programming language in which PolyAnalyst formalizes constructed procedures. Section 3 explains how the search in the space of procedures is organised. Methods used to avoid the tautologies and the families of equivalent procedures are enumerated. In Section 4 the heuristics which determine the search priorities are described. Section 5 is devoted to the principles of direct construction of procedures by subdividing the initial task to a number of the simpler subtasks. Since almost every constructed program contains constants it in general represents a family of procedures. This fact makes the evalutaion of a new program the non-trivial task. The evaluation procedure is described in Section 6. At last Section 7 is devoted to one of PolyAnalyst's applications.

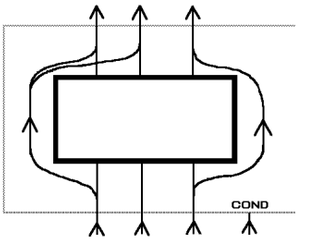
## 2. The internal functional programming language.

The PolyAnalyst system synthesizes procedures. A procedure can be described as an object with some number of inputs and outputs (number of outputs should be not less than 1). The simplest procedures are called primitives. Sets of primitives can be different for different tasks and are determined by the structure of analysed data and features of application field which are specified in the domain definition (DD) file (see Section 7). Every input and output of procedures has a data type associated with it. It should be one of the data types defined for the given task in the DD file. New procedures are constructed from existing ones with the help of several (4 in the present version) production schemes (Kiselev, Kiseleva, 1990). These production schemes and other features of the language resemble John Backus' FP formalism (Backus, 1978). The production schemes are illustrated by the diagrams below in which the solid rectangles denote procedures used to build the new procedure and the lines with the arrows denote the data streams.

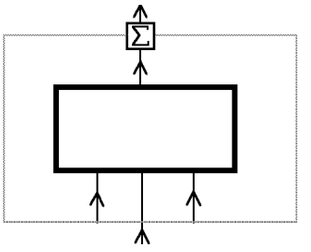
**a. Functional composition.** Input and output data types should match. It is also true for the other production schemes below.



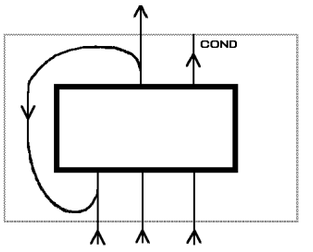
**b. Conditional execution.** If the value of the COND input is **true** then execute the procedure else copy some input values to outputs.




**c. Unconditional loop.** Put on some inputs of the procedure all possible combinations of values admissible for their data types subsequently and sum all values returned by the procedure. Naturally, the data types associated with these inputs should be discrete and finite. This construction is suitable to realize the predicate calculus quantors, integration, and other operations.




**d. Conditional loop.** Execute the procedure and test the value of its output marked as COND. If it is **false** then stop else copy values from outputs to inputs and begin new iteration. The outputs of the new procedure are all the outputs of the old procedure except the COND output (which is always **false** at the end of execution).




## 3. The search.

For every procedure built by the PolyAnalyst system a measure of its complexity is defined. The procedures are built in the order of increasing complexity. The heuristical search subsystem of the PolyAnalyst can reevaluate this complexity measure in accordance with certain heuristics (see the next Section) and due to this mechanism can change the search order. Thus, this

value called dynamical complexity is a sum of the true complexity of procedure and some heuristical corrections. The true complexity of a procedure is a sum of the complexities of the primitives entering it and the complexities of the production schemes which have been used for its creation.

It is obvious that to be operable the system should have the mechanisms which would recognize tautologies and equivalences of procedures and exclude tautological or equivalent procedures from the search. Further the tautologial procedures and procedures which are equivalent to ever found ones will be denoted by the term ’reducible’. It should be noted that these mechanisms are especially important for the PolyAnalyst because of the strong combinatorial growth of synthesized procedures. During the period of creation of the PolyAnalyst the great efforts were made to provide it with the effective multi-level system sifting away the reducible procedures. The performed tests show that the existing mechanism recognizes and eliminates practically 100% of the reducible procedures in the first 7 generations.

It includes the following components.

**a.** The production schemes are applied in the order which excludes the explicit "syntactic" equivalences. For example, if we have the three procedures *A*, *B*, and *C*, all with one input and one output then the procedure corresponding to their composition *A(B(C(.)))* could be built by the two ways - as a composition of *A(B(.))* and *C(.)* , and as a composition of *A(.)* and *B(C(.))* . The system recognizes such situation and uses the second way only.

**b.** Every input and output of procedures is marked not only by its data type but also by the other attribute called the ’algebra type’. This attribute helps to avoid the "local" equivalences based on the identities which include several primitives. For example, the primitives "+", "=", and "-" are bound by the identity *(a+b)=c <=> b=(c-a)*. The program which realizes the "functional composition" production scheme contains the rules which prohibit certain combinations of input and output algebra types. In the example mentioned above the system builds only the left hand side procedure because the output algebra type of the "-" primitive and the input algebra type of the "=" primitive are incompatible.

**c.** Symmetry of the primitive inputs can be specified. The mechanism which builds new procedures determines the symmetries of constructed procedure knowing the symmetries of its components. It allows to eliminate the "symmetry-based" equivalences such as *a&b<=>b&a*. Since the primitive "&" is declared symmetric only one variant is built.

**d.** It is possible to declare that connection of two inputs of the certain primitive with the same variable (an output of some procedure or a loop variable)

leads to tautology. The system will not build the procedures contaning such fragments. For example, $a\&a <=> a$.

**e.** It is the consequence of the general results of the theory of algorithms that in general the equivalnce of two procedures cannot be proven by recursive use of a finite set of rules like in **a.-d.** For this reason these exact "syntactic" methods are supplemented by the inexact methods based on the selective execution of the procedure for various combinations of the input values. These methods are called inexact in the sense that while they are able to recognize all the reducible procedures some not reducible procedures may be also erroneously determined as reducible. However in practice the risk to reject a "good" procedure can be minimized to reasonable values by increasing the number of performed random tests. One of these methods determines if some input of a procedure does not influence its output. If the test executions show that the output value of a procedure does not depend on some of its input for all tried combinations of values on the other inputs the procedure is declared tautological and discarded.

**f.** For all new procedures the returned values are calculated for definite set of the standard input value combinations. The returned values are used to calculate some procedure's characteristic value called procedure's signature. If two procedures have equal signatures they may be equivalent. In this case a number of tests is carried out and if in each test the values returned by the both procedures are equal the procedures are considered equivalent and the more complex one is discarded.

## 4. Heuristics determining the search priorities.

The heuristical rules of the dynamical complexity re-evaluation used in the PolyAnalyst system are divided into two cathegories. The heuristics from the first cathegory re-evaluate the dynamical complexity of procedures on the basis of their certain structural features. Their importance is determined by the fact that not all synthesized procedures can be a solution even potentially. For example, the procedure a&(b=c) constructed from two primitives "&" and "=" can not be a solution because it has no access to the analyzed data. To become a candidate for a solution (such procedures are called the complete procedures) it should be linked with some procedures which have access to the data through the special data access primitives. It is natural that percentage of incomplete procedures increases with the growth of complexity of built procedures. If not to take appropriate measures the system wastes more and more time constructing the incomplete procedures which will never be completed. The heuristics

increase the dynamical complexity of the incomplete procedures proportionally to number of steps necessary for their completion.

The heuristics from the second cathegory evaluate a procedure applying it to the training examples. They can decrease as well as increase the dynamical complexity and can change radically the search priorities.

The three of them are most important:

- If a procedure has been found to be a solution of some task in the given application field (or it may be one of the subtasks - see the next Section) decrease its dynamical complexity and to a lesser degree compexity of its parents and children.

The last two heuristics use a parameter of procedure called the enthropy of returned values (ERV). To calculate this number a procedure is applied to all training examples subsequently. Then ERV is the enthropy of the distribution of these values.

- If a procedure has low ERV values for many combinations of constants entering it increase its dynamical complexity.

The third heuristics uses the ratio $ERV/ERV_{rand}$ where $ERV_{rand}$ is ERV calculated not for the real examples but for generated set of the random examples. The random examples have the same structure as the real examples but consist of random values distributed by the same law as respective values in the real examples.

- If a procedure has high $ERV/ERV_{rand}$ value decrease its dynamical complexity.

It should be noted that the case when $ERV/ERV_{rand}$ is close to zero may be also interesting. It means that the built procedure expresses some specific property of the analysed data. Finding such characteristic properties can be the main aim in some tasks.

The additional opportunities to recognize the valuable procedures appear in the case when a measure of closeness of found procedure to the solution can be defined. This case includes the problem of inference of empirical dependences in bodies of data belonging to the "continuous" domains where the real numbers play important role. In this class of problems the standard error of prediction is a natural measure of closeness to the solution. The following search strategy has proven to be effective under such conditions. Two search processes run concurrently. The first one is the search process described in the previous section. It can roughly be called the breadth-first search. The second one takes

as a starting point the best (in terms of standard error) procedure found by the first process. For any procedure a set of transformations can be determined which do not increase the standard error. These transformations are called not worsening transformations (nWT). The second search process applies all possible nWT's to the best procedure. The derived procedure with the least standard error and the least number of the additional parameters is considered the best obtained solution. When the first process finds new best procedure the second process takes it as a new starting point and performs new nWT-search. The operation continues until desired standard error level is reached.

## 5. Subtasks and direct construction of procedures.

Despite the heuristics and other mechanisms which help to cut off not perspective search branches it is impossible to reach the solution if it is very complex not using some effective direct methods. The following scheme has proven to be effective in many cases: search -> task subdividing -> attempts to solve the subtasks -> construction of the solution from the solutions of the subtasks.

At present we use the following task subdividing strategies.

**a.** Suppose that the system has found a number of procedures each of which solves the task not over the whole set of training examples but over some its subsets so that all these subsets together overlay the full set of examples completely. Then the system initiates the new task of classification of training examples to these subsets. If it finds the solution of this subtask then the solution of the initial task is composed from the subtasks solutions in the obvious way.

**b.** Suppose that the system has found a procedure which returns the correct value being applied to each training example but with different values of the constants entering it. In this case the system can try to find the procedures realizing the mapping from the set of training examples to the desired values of the constants. After successful solution of these subtasks the final solution is constructed through the "functional composition" production scheme (**a.** in Section 2).

**c.** Suppose that the same situation occurs as in **b.** The system can choose the other way to use it. It forms new set of training examples adding to the existing examples the respective values of procedure's constants. Assuming that all new examples belong to some class it tries to solve the respective classification task. If the system finds its solution the solution of the initial task is constructed through the "conditional loop" production scheme.

## 6. Evaluation procedure.

After a new procedure has been constructed it is compiled to effective machine code. When it evaluates a new procedure PolyAnalyst calls it like the C language function.

As was mentioned in Introduction since a constructed procedure can include the constants it realizes a set of different dependences parametrized by values of these constants. To determine if the set contains the solution is therefore non-trivial problem. To solve this problem the present version of the PolyAnalyst system scans over all combinations of the discrete constants and for each combination it performs hill-climbing in the space of the continuous constants.

## 7. Example of PolyAnalyst's application. Dependence of the mechanical properties of polymeres on their IR-spectrums

The PolyAnalyst system has solved a number of problems in chemistry of polymeres. For example, it is given with a number of the IR-spectrums of the polyethylene films and the values of the elasticity of the corresponing samples (see Fig.1). The system should find a formula expressing dependence of elasticity on some spectral parameters. This task was solved by the PolyAnalyst version which worked only with discrete data. Therefore the spectral curves had to be transformed to some sets of discrete parameters. The problem of discretization of continuous data appears in many AI

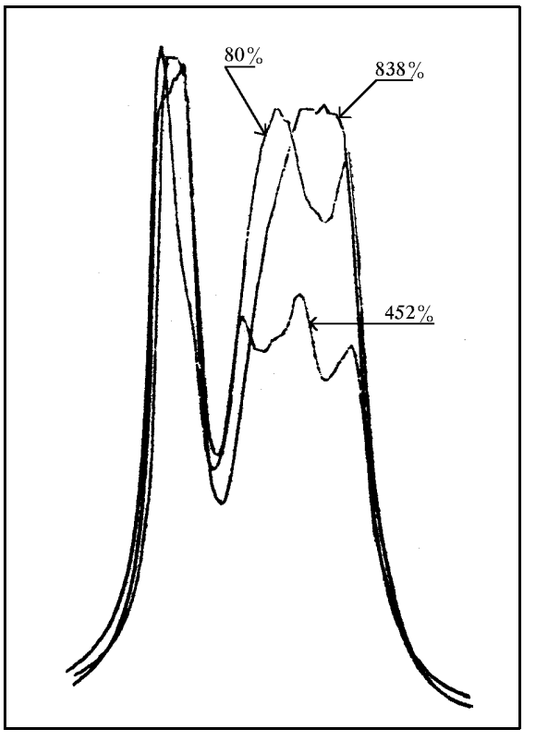



**Fig.1.** The examples of the IR-spectrums of the polyethylene films. The numbers are values of elasticity of the respective samples.

tasks - see an example of the discretization procedure in (Catlett, 1991). We used the following scheme. The investigated spectral band was broken to $N_X$ intervals. For each interval the average intensity, the intensity dispersion, and the number of peaks were calculated. Then the range of intensity was also broken to $N_Y$ intervals. Correspondingly the average intensities and the intensity dispersions were transformed to the integeres from 0 to $N_Y$-1. After this procedure every spectrum was represented by the three-sectional raw of the integers. The first section contained $N_X$ values of discretized average intensities, the second section contained $N_X$ discretized intensity dispersions, the third section contained $N_X$ values of numbers of peaks in the respective spectral interval. To determine the optimal values of $N_X$ and $N_Y$ the following consideration was used. To write the first two sections $2N_X\log_2 N_Y$ bits is necessary. We calculated the information gain (IG) for each spectrum and for different NX, NY values and selected those NX, NY values which gave the maximum information gain per bit ($IG/2N_X\log_2 N_Y$).

To start search for a solution the system should know the structure of the examples, the set of primitives, and the set of data types. This information is supplied in so called domain definition (DD) file. The DD file for our case is shown on the Fig.2. The section DATA_TYPES contains names of data types and their ranges. In our case $N_X$ = 4, $N_Y$ = 11. Here "band" is the spectral interval number. In the section ORDER the properties of the data types are specified. This information is necessary to generate correct set of primitives. For example, for the "intensity" data type the whole set of the arithmetical operations is generated while for the "band" data type only the operators "next", "previous", "equal to", and "less than" are defined. The section QUANT allows the loop variables of types "band" and "intensity". At last the ASS_INSTRUCTIONS section fixes the three-sectional structure of the examples. It also contains the verbal definition of meaning of the sections. The names of data types and example sections are used when the obtained dependence is translated by the PolyAnalyst to the "pseudo-

```
.DATA_TYPES
   N=0-10
   band=0-3
   intensity=0-10
.ORDER
   band=STRAIGHT
   intensity=ADDITIVE
.QUANT
   band,intensity
.ASS_INSTRUCTIONS
   band->intensity, of average level in $0
   band->L, there is high dispersion in $0
   band->N, of peaks in $0
.END
```

**Fig.2**. The DD file for the "IR spectrums" domain.

english" verbal form (see below). It should be added that after the discretization the dispersion values became 0s and 1s only so that they are treated as boolean (L).

```
*****   FB#381
MEANING:
#o0 is N $A - $B where
        $A is N of peaks in #0
        If not $D then $B is $C;
        else $B is N $C * #1
        where
                $C is N of peaks in #2
                $D is there is high dispersion in #3
THIS IS TARGET !!
inputs: 0 4 2 0

BEST RESULT reached by FB #381 operands: 0 4 2 0
res=-79.8*out+398.  ±  81.870
```

**Fig.3**. The "pseudo-verbal" form of solution

To solve this task (represented by spectrums of 47 polyethylene samples with known elasticity) the PolyAnalyst computed during several hours. The solution is shown on the Fig.3. At present the verbal representation produced by the PolyAnalyst is far from perfect - it is direct projection of the respective internal language structures. However if to make all recursive substitutions the following formula can obtained. Elasticity = -79.8 * out + 398. ± 81.870 where 'out' is the value returned by the following procedure.

$$\text{out} = (\text{N of peaks in band \#0}) - \begin{cases} \text{Npeaks in b.\#2 if low dispersion in b.\#0} \\ \text{else} \\ (\text{Npeaks in b.\#2}) * 4 \end{cases}$$

It can be expressed also as

out = (Npeaks in b.#0) - Npeaks in b.#2 * (1 + 3 * dispersion in b.#0)

The PolyAnalyst is able to determine if the set of training examples is too small. One of the simplest methods is following. It generates the pairs <spectral data, elasticity> in which the both components are from the real examples but are randomly mixed. The same task is solved for this randomized set also. If the found solution gives the close value of standard error the solution of the main task cannot be considered reliable. This test is performed for several different randomized training sets.

Beside the tasks from chemistry of polymeres the PolyAnalyst has solved a number of real tasks from medicine, geophysics, and agricultural science.

## 8. Conclusion.

Our research shows that it is possible to organize effective search in the space of procedures and use it for the real-world applications. The following factors are the most important for the systems based on this approach:

- effective mechanisms of avoidance of trivial and equivalent procedures;
- reliable principles of selection of the search priorities;
- combined use of search and direct inference of programs;
- combination of the universal search technique in the procedural spaces and special methods applicable to their subsets equivalent to some declarative formalisms.

The working mechanisms which realize the first three prnciples in the PolyAnalyst system were described in the paper. The mechanism realizing the effective search technique in the sub-spaces which are equivalent to the set of the predicate calculus formulae and the set of the rational functions is developed now.

## REFERENCES.


BACKUS J. (1978)
Can programming be liberated from the von Neumann style? Commun. ACM, v.21, pp 613-541.

CATLETT J. (1991)
On changing continuous attributes into ordered discrete attributes, Proceedings of Machine Learning - EWSL-91, pp 164-178.

FALKENHAINER B.C., MICHALSKI R.S. (1990)
Integrating Quantitative and Qualitative Discovery in the ABACUS System In: Y.Kodratoff, R.S.Michalski, (Eds.): Machine Learning: An Artificial Intelligence Approach (Volume III). San Mateo, CA: Kaufmann, pp 153-190.

JANTKE K.P. (Ed.) (1987)
Analogical and Inductive Inference, Springer-Verlag.

KISELEV M.V., KISELEVA M.N. (1990)
Procedural Approach to Learning from Examples, Proceedings of the second Conference "Artificial Intelligence -90" (USSR), vol.3, pp 197-200, - In Russian.

LANGLEY P., ZITKOV J.M., BRADSHAW G.L., SIMON H.A. (1986)
Rediscovering Chemistry with the Bacon system, Machine Learning, v.2, pp 425-469.

MACDONALD B.A., WITTEN I.H. (1989)
A Framework for Knowledge Acquisition through Techniques of Concept Learning, IEEE Trans. on Systems, Man, and Cyber., v.19, pp 499-512.

MICHALSKI R.S. (1993)
A Theory and Methodology of Inductive Learning In: B.G.Buchanan, D.C.Wilkins, (Eds.): Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems. San Mateo, CA: Kaufmann, pp 323-348.

PHAN T.H., WITTEN I.H. (1990)
Accelerating Search in Function Induction, J. Expt. Theor. Artif. Intell., v.2, pp 131-150.

SHEN Wei-Min (1990)
Functional Transformations in AI Discovery Systems, Artif. Intell., v.41, pp 257-272.

WOLFF J.G. (1987)
Cognitive development as optimization, I n: L.Bolk, (Ed.), Computational models of learning, Springer, Berlin, pp 161-205.

ZITKOV J.M., ZHU J. (1991)
Application of Empirical Discovery in Knowledge Acqusition, Proceedings of Machine Learning - EWSL-91, pp 101-117.