Regression-Based Classification Methods and Their Comparison with Decision Tree Algorithms

Mikhail V. Kiselev
Megaputer Intelligence Ltd., 38 B.Tatarskaya, Moscow 113184, Russia
megaputer@glas.apc.org

Sergei M. Ananyan
Department of Physics, College of William and Mary, Williamsburg, VA 23187 USA
sxanan@physics.wm.edu

Sergei B. Arseniev
Megaputer Intelligence Ltd., 38 B.Tatarskaya, Moscow 113184, Russia
megaputer@glas.apc.org

ABSTRACT

Classification learning can be considered as a regression problem with dependent variable consisting of 0s and 1s. Reducing classification to the problem of finding numerical dependencies we gain an opportunity to utilize powerful regression methods implemented in the PolyAnalyst data mining system. Resulting regression functions can be considered as fuzzy membership indicators for a recognized class. In order to obtain classifying rules, the optimum threshold values which minimize the number of misclassified cases can be found for these functions. We show that this approach allows one to solve the over-fit problem satisfactorily and provides results that are at least not worse than results obtained by the most popular decision tree algorithms.

KEYWORDS: classification learning, non-linear regression, decision trees

1 Introduction.

It probably would not be a mistake to say that classification learning is the most explored class of problems in KDD and machine learning. An exceptionally large number of proposed techniques and approaches, dramatically different from each other, distinguishes it from other important problems such as deviation detection or clustering. Nevertheless, majority of existing methods, no matter how diverse they are, have the same weak sides or at least the same directions for further perfection. In our opinion, problems limiting performance of symbolic methods for classification learning can be grouped in the following three categories.

1. Over-fit. A model created on the basis of training examples often includes many components each supported only by a minor part of examples. In case of presence of numerous insignificant submodels the whole developed model becomes

insignificant. When being validated on test examples, such a model may give a much greater number of classification errors than simpler models.

2. Difficulty of model interpretation by the user. In contrast with neural networks, goals of KDD include not only creation of a model that works well but also a possibility for the user to understand how it does it. Over-complicated models may require for their comprehension time of order of that necessary for a manual analysis of raw data.

3. Little expressive power of a knowledge representation language. Now it is obvious that a suitable choice of a hypotheses space can drastically increase performance of KDD systems [Bloedorn, Michalski 96]. Rules that have simple expression in some language may be approximated by very complex models when formalized in another language, thus masking the true nature of influencing factors and even leading to a practical impossibility to construct a model due to pruning or other mechanisms preventing over-fit. At the same time, having an efficient strategy of search in a sufficiently wide hypotheses space, we could in principle avoid this danger.

There exists probably no method which could eliminate all these problems completely so that the most wise strategy should include a combination of different approaches integrating their strong sides. From this point of view we deem it important to use one more classification learning technique which is able to overcome efficiently the difficulties listed above. This technique is based on methods implemented in the PolyAnalyst data mining system [Kiselev, Arseniev 96; Kiselev 95; Kiselev 94]. PolyAnalyst belongs to a class of KDD systems whose main purpose is discovery of numerical dependencies in data. This class of systems also faces problems similar to the above mentioned problems, and the fact that PolyAnalyst can resolve them successfully allows us to think that its application to classification learning could be fruitful as well.

Further discussion is organized in the following way. Section 2 contains description of proposed technique. Section 3 is devoted to comparison of our technique with popular decision tree algorithms on publicly available benchmark data accompanying one of the most well known shareware decision tree system SIPINA-W [Zighed 92] and to discussion of these results. And last Section 4 contains a summary of results of the research.

2 Regression-based methods for solution of classification learning problems.

Let us assume that we have two groups A and B of training examples - records having the same format (the same set of numerical, boolean and symbolic fields). We should build some rule which relates any possible record of the same format to group A or B depending on values of its fields. It is one of general formulations of the classification learning problem. The essence of the proposed method of solution of this problem is quite simple. We complement records from both groups by one more numerical field **IND** containing zeros in records from the group A and ones in

records from the group B. Then we use some method of searching for a numerical dependence between **IND** and other fields. As in majority of regression methods we use the "least sum of squares" criterion for the evaluation of a found dependence. An obtained continuous function can be considered as a fuzzy membership indicator for a set defined by the group B. This thesis can be substantiated by the following reasoning.

Suppose that both groups A and B include a large number of records and we have found some function $f_{IND}(x_i)$ as a result of solving our regression problem. Here x_i is the set of fields entering our regression model ($x_i \in X \subset \Re^n$). Let us break X into a number of small regions such that variation of function $f_{IND}(x_i)$ in each region is negligibly small. It is evident that if the value of this function in each region were equal to $n_B /(n_A+n_B)$, where n_A and n_B are numbers of records from groups A and B in this region, then this function would have the least possible sum of squares of deviations and therefore would be an ideal solution for our regression problem. At the same time this function would express the probability that a record with certain values of its fields x_i belongs to the group B and therefore would correspond to definition of a fuzzy set membership indicator.

Of course, in practice, the found regression function cannot serve as an exact measure of probability that a given record belongs to the group B. For example, its value does not lie necessarily in [0,1] interval. Nevertheless, in many cases it can provide an acceptable picture of distribution of data points from sets A and B in different regions of X. The found function $f_{IND}(x_i)$ can be used in a classification rule of the form $f_{IND}(x_i) > a$, where a threshold value a is selected so that this rule would produce a minimum number of misclassified training examples.

This method can be used also when classification rules for more than two sets should be obtained. If training examples include representatives of N classes where $N > 2$ we should solve N regression problems so that in a j-th regression problem the value of the dependent variable is equal to 1 for the records belonging to the j-th class. As a result we obtain N functions f_j expressing the measure of affinity to the j-th class. In order to classify a new record we should calculate for it functions f_j for all values of j and if the value of a function f_k is the largest one, we should relate the record to the k-th class.

Thus, for every algorithm of discovery of numerical dependencies we have a respective algorithm of classification learning. One can see that if some method for discovery of numerical dependencies solves efficiently problems mentioned in Introduction, we can hope that the respective classification learning method will also have good characteristics. It is obvious for problems 2 and 3 and it can be proven formally for problem 1 (utilizing the fact that a high value of the standard error shown for test examples by a classifying function f_{IND} is equivalent to a high number of classification errors under very general assumptions about the distribution of residuals).

In a practical realization of our approach we used two methods for discovery of numerical dependencies implemented in the data mining system PolyAnalyst [Kiselev, Arseniev 96], namely our version of multi-linear regression with an automated selection of independent variables and creation of general non-linear regression models in the form of functional programs (so called Core PolyAnalyst algorithm). The first method will be referred to as LIN, the second - as PA.

The LIN method uses the value of F-statistics as a criterion for inclusion in the regression model or rejection of independent variables. In order to evaluate significance of the result under condition of multiple regression model trials a randomized testing is performed. As it was shown (see, for example, [BKW 80; Jensen 91]) these measures allow one to build reliable regression models avoiding over-fit. Linear models are usually interpreted by the user easily because effect of each independent variable can be considered independently. At the same time the LIN method has two weak sides related to problem 3 of Introduction. Firstly, it does not allow to express nonlinear effects of interaction between independent variables. Secondly, it does not allow one to include into a model variables which are not numerical. However, as will be shown below, despite these drawbacks the LIN method shows very good results on the benchmark data used for comparison of different classification methods (this was a surprise for the authors of this paper).

We will furnish here only a very brief sketch of algorithms underlying the PA method - their detailed description would require too much space (see [Kiselev 94]). The kernel of the data mining system PolyAnalyst is a mechanism that builds new functional programs from existing functional programs. Ignoring semantics of these programs, they can be understood as abstract objects with some number of inputs (or without inputs) and one output. Inputs (also called arguments) and outputs are marked by their ***type*** and some other attributes. The simplest atomic functional procedures are called primitives. The set of primitives is determined by the structure and properties of data to be analyzed. The set of primitives includes standard and user-defined primitives. Information in databases is accessed via special data access primitives which also can be standard or user-defined. To build new functional programs from existing ones PolyAnalyst uses four production schemes. The simplest scheme is ***functional composition***. PolyAnalyst takes one functional program (it is called a producing function) and connects some its inputs with outputs of some other existing programs. This process is controlled by the ***compatibility rules*** which prohibit certain combinations of attributes of connected inputs and outputs. The second scheme serves to create iterative or recursive functional programs. It realizes constructions similar to `for` and `while` blocks of C language. The scheme called ***difference*** is used when structured data such as time series or vectors are explored. It produces programs calculating the difference of some numerical values for neighboring "data points" in these series or vectors. And last is a scheme most important for discovery of numerical laws. It produces functional programs realizing rational expressions (a polynom divided by a polynom) formed from numerical constants and some existing programs (naturally, they should return numerical values).

In fact, this mechanism realizes a simple universal programming language suitable for formalization of a wide range of laws and rules which can be discovered in data. PolyAnalyst has an explanation generator module which translates rules expressed in the form of functional programs into a clear verbal form including standard mathematical notation. Generator of functional programs is controlled by the search strategy module. The search direction is determined in accordance with evaluation of each individual functional program carried out by the search strategy module. The search process is a combination of the full search (low priority component) and so-called 'generalizing transformations' - GT (high priority component). The GT process takes one of the best found programs (called the root program) and uses it for creation of new programs with the help of one of the above mentioned production schemes in all possible ways satisfying the following condition. Each derivative program should have some set of arguments such that when these arguments have certain constant values, the derivative program becomes identical to the root program. This condition guarantees that derivative programs fit data not worse (in terms of the standard error) than the root program. In normal situations the GT process takes the most part of the PolyAnalyst computation time.

Methods for avoiding over-fit used in PA are similar to ones used in LIN. They are: randomized testing and analysis of significance of regression model components on the basis of calculation of the respective regression coefficient standard errors. We calculate these errors using linearization of considered non-linear model around found values of regression coefficients and applying standard technique of linear regression methods. Our version of randomized testing includes the following steps.

1. The randomized vectors of dependent variables are created using random transpositions of the original vector. The vectors of independent variables in all randomized tests remain the same.

2. The classification problem is solved for original vector of dependent variables and for every randomized vector.

3. Standard errors obtained for original and randomized vectors are compared.

These methods help building only reliable and significant regression models. Since discovered dependencies are expressed in the language which represents some natural extension of the usual mathematical notation, all obtained results are easily understood by a human. A prodigious expressive power of the PolyAnalyst knowledge representation language allows one to formalize a great variety of possible dependencies. Thus, the PA method seems to satisfy all the requirements stated in Introduction, and we can hope that it will show good benchmark results as will be demonstrated in the next section.

3 Comparison of methods LIN and PA with decision tree algorithms on benchmark data.

Naturally, the most reliable way of evaluation of relative efficiency of different classification learning methods is a comparison of results obtained by them on data corresponding to real data analysis problems. Fortunately, at present several sets of

classification learning benchmark data are publicly available, and results demonstrated by various systems in these benchmark tests are known. For purposes of our research we have chosen the set of benchmark data accompanying a shareware decision tree system SIPINA-W. This system is created in the University of Lyon, France by a research group under direction of professor D.A. Zighed [Zighed 92]. The system together with the benchmark data, their description and table of results obtained by various decision tree systems can be downloaded from the University of Lyon FTP site `ftp://eric.univ-lyon2.fr/pub/sipina`. We chose this benchmark data set because it represents real-world classification problems from various fields, results shown by other tested systems are documented, and, moreover, they can be reproduced partially by the SIPINA-W system. The main drawback of this benchmark data set is a small number of test problems. The benchmark set consists of 6 test problems but we had to use only 5 of them because one dataset includes only boolean variables. The LIN method cannot be applied to data without numerical fields, while application of the PA method also can hardly give good results since both methods are based on numerical regression algorithms. Nature of these methods determines their weakness in domains where non-numerical data prevail.

In all the problems data are divided into two groups - learning data and test data. The learning data are used for construction of a classifying rule, the test data are used for the evaluation of its accuracy. We used the following five problems in our research (see the full description of problems in the SIPINA-W distribution package):

1. Classification of sorts of irises. Input data describe parameters of iris flowers. This problem is the least realistic because it is quite simple. The data include only 4 independent numerical variables, 98 learning examples, 52 test examples.

2. Ischemic heart disease diagnosis. Input data contain physiological parameters of patients and results of specialized medical tests - 7 numerical, 3 boolean, 3 categorical variables, 180 learning examples, 117 test examples.

3. Contents of the non-ethyl alcohols in the spirits made from different kinds of plants. Independent variables describe percent contents of various alcohols. All examples are divided into three classes corresponding to plants used for production of a spirit - cherry, mirabelle, or pear. 9 numerical variables, 102 learning examples, 56 test examples.

4. Breast cancer diagnostics. Independent variables are citological test data - 9 numerical variables, 444 learning examples, 239 test examples.

5. Impulse form recognition. Independent variables represent 21 consequent measurements of some signal. A rule for recognition of 3 basic impulse forms should be inferred. 21 numerical variables, 300 learning examples, 5000 test examples.

Results obtained by the following systems and algorithms were compared:

1. SIPINA. A shareware version of SIPINA-W has been applied by the authors of this paper to all 5 benchmark problems. The "automatic" regime of analysis was used. No default algorithm parameters were changed.

Table 1. Benchmark results.

	Pred%(L)	*Pred%(T)*	*Unclass.(T)*	*Depth*	*Nodes*	*Complexity*
IRIS[Learning:98; Test:52] - 4N						
SIPINA	97	98	0	5	8	16
C4.5	96	90	0	3	5	10
Chi2-Link	97	98	0	4	7	14
CART	93	94	0	3	5	10
PolyAnalyst(LIN)	99	94	0	2	2	9
PolyAnalyst(PA)	98	87	0	2	2	5
HEART DISEASES[Learning:180, Test: 117] - 7N3B3C						
SIPINA	87	66	22	16	39	78
C4.5	93	69	8	9	50	100
Chi2-Link	80	71	0	4	14	28
CART	73	65	7	2	3	6
PolyAnalyst(LIN)	78	79	0	2	1	7
PolyAnalyst(PA)	78	73	0	2	1	7
ALCOHOL[Learning:102, Test:56] - 9N						
SIPINA	94	84	6	15	28	56
C4.5	98	78	0	9	21	42
Chi2-Link	82	85	0	3	5	10
CART	82	85	0	3	5	10
PolyAnalyst(LIN)	90	91	0	2	2	13
PolyAnalyst(PA)	93	87.5	0	2	2	8
BREAST CANCER[Learning:444, Test:239] - 9N						
SIPINA	98	93	6	9	20	40
C4.5	99	94	0	10	27	54
Chi2-Link	95	92	4	5	19	38
CART	97	94	0	3	5	10
PolyAnalyst(LIN)	98	95	0	1	1	8
PolyAnalyst(PA)	98	95	0	1	1	25
WAVES[Learning:300, Test:5000] - 21N						
SIPINA	90	69.5	650	18	54	108
C4.5	89	70	0	14	47	94
Chi2-Link	74	56	529	6	23	46
CART	82	68	0	5	11	22
PolyAnalyst(LIN)	83	79	0	3	5	74
PolyAnalyst(PA)	82	82	0	3	5	46

2. C4.5 [Quinlan 93]. This and two other decision tree algorithms were not tested by the authors themselves - the respective results were taken from a table in the SIPINA-W distribution package.

3. Chi2-Link - a modification of the method proposed in [Kass 75].

4. CART [BFOS 84].

5. LIN - using PolyAnalyst DM system.

6. PA - using PolyAnalyst DM system.

Results of comparison are presented in Table 1. It contains two groups of parameters. The first group characterizes predictive power of an obtained rule, the second one characterizes its simplicity and clarity for comprehension. The most important parameter in the first group is percent of correctly classified test examples (denoted as `Pred%(T)`). It is the main difference from a table in the SIPINA distribution package which contains in respective columns the sum of percents of correctly classified and unclassified examples. In order to find some common measure of complexity for decision trees and rules obtained by PolyAnalyst the number of "elementary operations" contained in these constructions is counted. It seems natural to consider one arithmetic operation, relational operator (such as ">"), or selection operation as an elementary operation. Since each decision tree node contains one relational operator and one selection operation, the decision tree with N non-terminal nodes should be characterized by complexity 2N. Rules obtained by LIN and PA methods can be found in Appendix.

Despite the fact that rules obtained by LIN and PA methods classify wrongly a relatively large number of learning examples they provide the best results for test examples in all benchmark problems except "***IRIS***" problem. It allows us to think that these methods can not only generate more accurate classification rules but also prevent over-fit efficiently (because they give a smaller value of the `Pred%(L)-Pred%(T)` difference). At the same time they usually produce simpler rules, which is important for their interpretation.

Results of this research contain a fact unexpected for the authors that LIN method often gives better results than PA method. At present we have no satisfactoryexplanation of this fact. Possibly it can be explained by the nature of the considered data, which might be characterized by independent and linear contribution of different input variables. In any case we plan to carry out further benchmark tests, for example, using public domain databases from the University of California in Irvine machine learning repository.

4. Conclusion.

In the present paper we proposed a classification learning technique based on regression methods for discovery of numerical dependencies implemented in PolyAnalyst data mining system. An ability of PolyAnalyst to discover a great variety of forms of functional dependencies and its efficient mechanisms for evaluation of model significance leads us to believe that its application to classification learning problems would allow one to overcome the difficulties listed in Introduction. In order

to test characteristics of proposed classification learning methods we applied them to 5 benchmark problems. Comparison of obtained results with results demonstrated by the most popular decision tree systems shows that our methods give more exact classifying rules for 4 problems out of 5, while at the same time the rules obtained by PolyAnalyst are the simplest ones for 3 problems. These results confirm our assumption about high efficiency of proposed classification methods. It should be noted however that methods described in this paper can be used efficiently only in the problems where numerical data and relations play an important role since they are based on numerical algorithms.

It is interesting that the method based on multi-linear regression produced more accurate classification rules than the method based on general non-linear regression models. This and other features of discussed methods will be explored further in the process of testing the developed technique on a larger number of more diverse data sets.

ACKNOWLEDGEMENTS.

The authors would like to express their gratitude to Professor D.A. Zighed, members of his research group and other people taking part in creation of SIPINA-W: S. Alussi, R. Bac, L. Jouvray, V. Placer, L. Ponsard, S. Rabaseda, R. Rakotomalala.

This work was supported in part by Russian Foundation for Fundamental Research grant 95-01-00775.

REFERENCES

Belsley, D.A., Kuh, E., Welsch, R.E. Regression diagnostics: identifying influential data and sources of collinearity, New York, John Wiley & Sons, 1980.

Bloedorn, E., Michalski, R.S. The AQ17-DCI system for data-driven constructive induction and its application to the analysis of world economics, in: Proceeding of ISMIS'96 (Ninth International Symposium on Methodologies for Intelligent Systems), Zakopane, Poland, Springer, 1996, pp 108-117.

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J. Classification and Regression Trees, Belmont, CA : Wardsworth, 1984.

Jensen, D. Knowledge discovery through induction with radomization testing, in: Proceedings of the AAAI KDD-91 Workshop, Anaheim, CA, 1991, pp 148-159.

Kass, G.V. An exploratory technique for investigating large quantities of categorical data, Applied Statistics, 24(2), 1974

Kiselev, M.V. PolyAnalyst - a machine discovery system inferring functional programs, in Proceedings of AAAI Workshop on Knowledge Discovery in Databases'94, Seattle, 1994, pp. 237-249.

Kiselev, M.V. PolyAnalyst 2.0: combination of statistical data preprocessing and symbolic KDD technique, in: Proceedings of ECML-95 Workshop on Statistics,

Machine Learning and Knowledge Discovery in Databases, Heraklion, Greece, 1995, pp. 187-192.

Kiselev, M.V., Arseniev, S.B. Discovery of numerical dependencies in form of rational expressions, in; Proceedings of ISMIS'96 (Ninth International Symposium on Methodologies for Intelligent Systems), Zakopane, Poland, Springer, 1996, pp. 134-145.

Quinlan, J.R. C4.5 Programs for machine learning. Morgan Kaufmann, 1993.

Zighed, D.A., Auray, J.P., Duru, G. SIPINA: Méthode et logiciel. Lyon Lacassagne, 1992.

Appendix

Here we represent the rules found by LIN and PA methods for 5 test problems mentioned above.

1. IRIS.

LIN:
```
1.49144 + 0.489025 * SepalWidth - 0.179836 *
PetalLength - 0.91428 * PetalWidth > 0.457002
```

PA:
```
PetalWidth>1.67262 and PetalLength<5.427524
```

2. HEART DISEASES

LIN:
```
0.0888805 * v10 + 0.138755 * v11 + 0.199631 * v12 >
0.495267
```
PA:
```
0.583921 * if(v3 = 4, 1, 0.223904) + 0.167872 * v12 >
0.525204
```

3. ALCOHOL

LIN:
```
if(bu1<4.357) return(kirsch)
else    if(0.999208 - 0.00703486 * bu2 +
          0.00236985 * isop - 0.0137976 *
          mepr + 0.000859275 * pro1 -
          0.0147635 * acal > 0.487445)
          return(mirab)
        else return(poire)
```

PA:
```
if(bu1<4.357) return(kirsch)
else    if(0.0492531 * pro1 / (bu2 + 0.163046
           * mepr) > 0.414355) return(mirab)
        else return(poire)
```

4. BREAST CANCER

LIN: -0.195398 + 0.0334729 * v1 + 0.0617623 * v2 + 0.0568881 * v6 > 0.268438

PA: (0.0967923 * v6 * v2 + 0.170927 * v2 * v2 - 0.831194 * v2 + 0.0228706 * v6 * v2 * v2 * v2 - 0.119172 * v6 + 0.659766) / (v2 + 0.0243011 * v6 * v2 * v2 * v2 - 0.220651 * v1) > 0.508278

5. **WAVES** The formulae below evaluate likeliness that a given example is related to categories w1, w2, or w3, respectively. Each example is related to a category for which likeliness is the greatest.

LIN:

```
eva(w1) = 0.512251 + 0.0230114 * v1 + 0.0275742 * v3 +
0.0648057 * v4 + 0.0382573 * v5 - 0.0453667 * v9 -
0.0225616 * v10 - 0.035247 * v11 - 0.0349169 * v12 -
0.0317344 * v14 + 0.021806 * v16 + 0.0596144 * v17 +
0.0515807 * v18 + 0.0237215 * v20 - 0.0255654 * v21
eva(w2) = 0.304898 - 0.0303732 * v1 + 0.0305195 * v2 +
0.0313452 * v8 + 0.0230719 * v9 + 0.0303204 * v10 +
0.0286657 * v11 - 0.0473285 * v13 - 0.0333897 * v15 -
0.0221375 * v16 - 0.0415234 * v17 + 0.0291991 * v21
eva(w3) = -0.0278807 * v2 - 0.0230603 * v3 - 0.0602541 *
v4 - 0.0431232 * v5 + 0.0283923 * v9 + 0.0335904 * v12 +
0.0551694 * v13 + 0.0428301 * v14 + 0.027215 * v15 -
0.0359495 * v18
```

PA:

```
eva(w1) = (29.1516 - 0.168725 * v12 * v12 * v11 - 1.33135
* v9 * v10 + 4.78966 * v4) / (v10 * v11 * v11 + 36.7588)
eva(w2) = (0.752802 * v9 + 0.273963 * v11 * v6 - 0.619969
* v16) / (v15 * v15 + 3.17794 + 0.928589 * v6)
eva(w3) = 0.00597957 * v14 * v13 * v12 + 0.0210095 * v5 *
v5 - 0.118357 * v5 + 0.0884176 + 0.0168819 * v15 * v11 -
0.00651304 * v4 * v15 * v12
```