Metodološki zvezki, Vol. 1, No. 1, 2004, 131-142

Discovery of Polynomial Equations for Regression

Ljupčo Todorovski, Sašo Džeroski, and Peter Ljubič[1]

Abstract

Both equation discovery and regression methods aim at inducing models of numerical data. While the equation discovery methods are usually evaluated in terms of comprehensibility of the induced model, the emphasis of the regression methods evaluation is on their predictive accuracy. In this paper, we present CIPER, an efficient method for discovery of polynomial equations and empirically evaluate its predictive performance on standard regression tasks. The evaluation shows that polynomials compare favorably to linear and piecewise regression models, induced by the existing state-of-the-art regression methods, in terms of degree of fit and complexity.

1 Introduction

Equation discovery (Langley et al., 1987) aims at developing methods for computational discovery of quantitative laws or models, expressed in the form of equations, in collections of measured numerical data. Equation discovery methods are mainly used for automated modeling of real-world systems from measurements and observations. Since they operate on numerical data, they are strongly related to the regression methods used in statistics and data mining for inducing predictive models of an observed numerical variable (Hastie et al., 2001).

Although the methods for equation discovery and regression methods are highly related, they differ mainly in the application focus. The focus of equation discovery is on inducing *comprehensible* and *general* quantitative laws or models of the observed real-world system or phenomena. The emphasis is on the comprehensibility of the induced model and especially its ability to reveal the structure of the observed real-world system. On the other hand, regression methods focus on the problem of inducing *accurate* predictive models of the observed system variable. While the comprehensibility, explanatory power, generality, and predictive accuracy of the induced models are equally important for measuring the performance of equation discovery methods, the predictive accuracy is the most important (or the only) criterion of success for the regression methods. Due to the difference between these two kinds of methods, outlined above, they are usually evaluated on different kinds of problems. While equation discovery methods are evaluated on the tasks of rediscovering

[1] Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia;
ljupco.todorovski@ijs.si, saso.dzeroski@ijs.si, peter.ljubic@ijs.si

existing or building new models of real-world systems (Todorovski and Džeroski, 1997; Washio and Motoda, 1997), regression methods are evaluated on the tasks of predicting a value of a numeric variable (Blake and Merz, 1998).

However, existing equation discovery methods would not be directly applicable on standard regression tasks. A main reason is their computational complexity, since they typically explore large space of potentially very complex candidate equation structures. In order to approach the regression task efficiently, we develop an equation discovery method CIPER that heuristically search through a space of polynomial equations. CIPER can be seen as a stepwise regression method for inducing polynomial equations — the differences to standard stepwise regression methods will be discussed further in Section 3. We evaluate the of CIPER on fourteen standard regression tasks from the UCI Repository of Machine Learning Databases and Domain Theories (Blake and Merz, 1998) and compare its performance to the performance of standard state-of-the-art regression methods built in the WEKA data mining suite (Frank and Witten, 1999).

The paper is organized as follows. First, equation discovery methods and their relation to the methods for inducing predictive regression models is introduced in Section 2. In Section 3 we introduce CIPER, an efficient search-based method for discovery of polynomial equations. Section 4 presents the results of the empirical evaluation of the proposed method as well as the results of its comparison with standard regression methods. Finally, Section 5 concludes the paper with a brief summary and directions for further work.

2 Equation discovery and regression

The task of equation discovery can be defined as follows: Given measurements of a set V of observed system variables $\{v_1, v_2, \ldots, v_n\}$, find a set of equations of the form $f(v_1, v_2, \ldots v_n) = 0$ that summarize the observations.

Early approaches to equation discovery dealt with rediscovering empirical laws from the history of science. Experiments with early equation discovery methods showed that a small set of data-driven heuristics can be used to rediscover many apparently complex physical laws including the ideal gas law, the law of gravitation, the law of refraction and Black's specific heat law (Langley et al., 1987). The performance of the early equation discovery methods was assessed as the success of rediscovery, i.e., by comparison of the discovered equations with the original ones.

In the process of development from early approaches to the present, the focus of the equation discovery methods shifted from rediscovering known quantitative laws to discovering new laws and inducing mathematical models of complex real-world systems (Džeroski, 1995; Washio and Motoda, 1997; Todorovski and Džeroski, 1997). However, the main emphasis remained on inducing *comprehensible* and *general* models of the observed systems, rather than models that have high predictive power. Following this, the performance of the equation discovery methods is assessed by means of the comprehensibility of the model and especially its ability to explain (or to reveal the structure of) the observed real-world system.

Since they deal with modeling numerical variables, equation discovery methods are strongly related to the regression methods from the area of statistics and data mining. The regression methods are used to induce predictive models of an observed numerical variable. More formally, the regression task can be defined as follows: Given measurements of a set V of system variables $\{v_1, v_2, \ldots, v_n\}$, find a model that predicts the value of a designated system variable $v_d \in V$ on the basis of the values of the independent variables in $V \setminus \{v_d\}$.

Unlike equation discovery, which does not focus on any particular variable, regression focuses on the dependent variable v_d. Explicit models that express v_d as a function of the independent variables from $V \setminus \{v_d\}$ are considered. Classical regression approaches from statistics consider mostly linear regression, where a linear function of the independent variables is used. The approaches based on regression trees employ piece-wise linear functions of the independent variables. The success of induced model is measured in terms of its accuracy.

Note that regression can be formulated as an equation discovery task. In that case the predictive regression model is an equation of the form $y = f(v_1, v_2, \ldots, v_n)$, where f comes from the class of functions considered by the equation discovery method. In this paper, we will consider the simple and yet general class of multivariate polynomial functions. An efficient method for discovery of polynomial equations is presented in the next section.

3 Discovery of polynomial equations

In this section, we present a heuristic search algorithm Ciper that searches through the space of polynomial equations and finds the one that has an optimal value of the heuristic function. First, we introduce a refinement operator that orders the space of polynomial equations. Then, we present the heuristic function used to measure the quality of each equation considered during the search along with the stopping criterion. After presenting the search algorithm based on beam search strategy, we discuss the relation of Ciper to stepwise regression methods.

3.1 The language of polynomial equations

We focus here on discovery of polynomial equations that can be used to predict the value of a dependent variable v_d. Given a set of variables V, and a dependent variable $v_d \in V$, a polynomial equation has the form $v_d = P$, where P is a polynomial over $V \setminus \{v_d\}$, i.e.,

$$P = \sum_{i=1}^{r} \mathrm{const}_i \cdot T_i,$$

where each T_i is a multiplicative term, r is the number of such terms, and const_i are real-valued constants. Each term is a finite product of variables from $V \setminus \{v_d\}$:

$$T_i = \prod_{v \in V \setminus \{v_d\}} v^{d_{v,i}},$$

where $d_{v,i}$ is (a non-negative integer) degree of the variable in the term. The degree of 0 denotes that the variable does not appear in the term. The sum of degrees of

Table 1: The refinement operator for ordering the space of polynomial equations.

original (current) equation
$v_d = \sum_{i=1}^{r} \text{const}_i \cdot T_i$
refined equations that increase r (one for each $v \in V \setminus v_d$)
$v_d = \sum_{i=1}^{r} \text{const}_i \cdot T_i + \text{const}_{r+1} * v$, where $\forall i : v \neq T_i$
refined equations that increase d (one for each T_j and $v \in V \setminus v_d$)
$v_d = \sum_{i=1, i \neq j}^{r} \text{const}_i \cdot T_i + T_j * v$, where $\forall i \neq j : T_j * v \neq T_i$

all variables in a term is called the degree of the term, i.e., $\deg(T_i) = \sum_{v \in V \setminus \{v_d\}} d_{v,i}$. The degree of a polynomial is the maximum degree of a term in that polynomial, i.e., $\deg(P) = \max_{i=1}^{r} \deg(T_i)$. The length of a polynomial is the sum of the degrees of all terms in that polynomial, i.e., $\text{len}(P) = \sum_{i=1}^{r} \deg(T_i)$.

For example, consider a set of variables $V = \{x, y, z\}$, where z is chosen to be a dependent variable. The term x (that is equivalent to $x^1 y^0$) has degree 1, the term $x^2 y$ has degree 3, while $x^2 y^3$ is a term of degree 5. An example polynomial equation is $z = 1.2x^2 y + 3.5xy^3$. It has degree 4 and length 7.

3.2 The refinement operator

In order to apply heuristic search methods to the task of inducing polynomial equations, we first have to order the search space of candidate equations. We introduce a refinement operator that orders this space according to equation complexity. Starting with the simplest possible equation and iteratively applying the refinement operator, all candidate polynomial equations can be generated.

Assume we measure the complexity of the polynomial equation $v_d = P$ as $\text{len}(P)$. The refinement operator increases the complexity of the equation by 1, either by adding a new linear term or by adding a variable to an existing term. First, we can add an arbitrary linear (first degree) term (that is a single variable from $V \setminus \{v_d\}$) to the current equation as presented in the first (upper) part of Table 1. Special care is taken that the newly introduced term is different from all the terms in the current equation. Second, we can increase the complexity $\text{len}(P)$ by adding a variable to one of the terms T_j in the current polynomial equation. Again, care should be taken that the changed term is different from all the other terms in the current equation. Note that the refinements of a given polynomial P are super-polynomials of P. They are minimal refinements in the sense that they increase the complexity of P by one unit.

The branching factor of the presented refinement operator depends on the number of variables $|V|$ and number of terms in the current equation r. The upper bound of the branching factor is $\mathcal{O}((|V| - 1)(r + 1)) = \mathcal{O}(|V| r)$, since there are at most $|V| - 1$ possible refinements that increase r and at most $(|V| - 1)r$ possible refinements that increase d.

The ordering of the search space of polynomial equations, defined on the set of variables $V = \{x, y, z\}$, where z is the dependent variable, is presented in Figure 1.

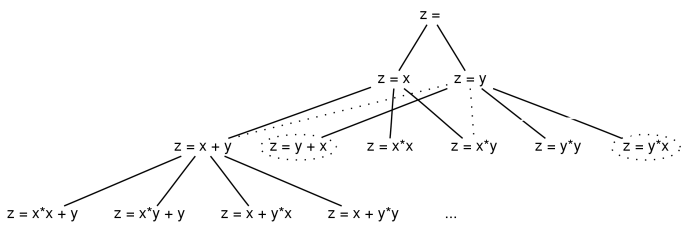

Figure 1: The search space of polynomial equations over the set of variables $V = \{x, y, z\}$, where z is the dependent variable, as ordered by the refinement operator from Table 1. Note that for simplicity, real-valued constants are omitted from the equations.

It shows that the defined refinement operator is not optimal, in sense that each polynomial equation can be derived more than once. This is due to the commutativity of the addition and multiplication operators. An optimal refinement operator can be easily obtained by taking into account the lexical ordering of the variables in V. Then, only variables (and/or terms) with higher lexical rank should be added to the terms and/or equations. The dotted nodes in the graph in Figure 1 denote equations that would not be generated by the refinement operator that takes into account lexical order. However, the redundancy due to the sub-optimality of the refinement operator can be avoided during the search procedure, as we will point out in the following section.

While an optimal refinement operator is desired for complete/exhaustive search, it may prevent the generation of good equations in greedy heuristic search. Suppose the polynomials x and z have low heuristic value, while y has a high heuristic value and $x + y$ is actually the best. Greedy search would choose y and the optimal refinement operator that takes into account lexicographic order would not generate $x + y$.

3.3 The search heuristic

Each polynomial equation structure considered during the search contains a number of generic constant parameters (denoted by const_i). In order to evaluate the quality of an equation, the values of these generic constants has to be fitted against training data consisting of the observed values of the variables in V. Since the polynomial equations are linear in the constant parameters, the standard linear regression method can be used for this purpose.

The quality of the obtained equation is evaluated using a degree of fit measure that measures the discrepancy between the observed values of v_d and the values predicted using the equation. One such measure is mean squared error (MSE), calculated as: $\text{MSE}(v_d = P) = \frac{1}{m}\sum_{i=1}^{m}(v_d(i) - \hat{v}_d(i))^2$, where $v_d(i)$ is the value of v_d for the i-th training example, $\hat{v}_d(i)$ is the value of v_d for the same example, but predicted using equation $v_d = P$, and m is the number of training examples.

Ciper uses an MDL (minimal description length) based heuristic function for evaluating the quality of equations that combines the degree of fit with the complex-

Table 2: A top-level outline of Ciper's beam search procedure.

```
    procedure Ciper($D$, $v_d$, $b$)
1   $E_0$ = simplest polynomial equation ($v_d$ = const)
2   $E_0$.MDL = FitParameters($E_0$, $D$)
3   $Q = \{E_0\}$
4   repeat
5     $Q_r$ = {refinements of equation structures in $Q$}
6     foreach equation structure $E \in Q_r$ do
7     $E$.MDL = FitParameters($E$, $D$)
8     endfor
9     $Q$ = {best $b$ equations from $Q \cup Q_r$}
10  until $Q$ unchanged during the last iteration
11  print $Q$
```

ity of the equation. In the literature, the following combination has been considered: based on Akaike and Bayesian information criteria for regression model selection (Hastie et al., 2001):

$$\mathrm{MDL}(v_d = P) = \mathrm{len}(P) \log m + m \log \mathrm{MSE}(v_d = P).$$

where len(P) is the length of the P, and m number of training examples. While the second term the MDL heuristic function measures the degree of fit of a given equation, the first term introduces a penalty for complexity of the equation. Through this penalty the MDL heuristic function introduces preference toward simpler equations.

3.4 The search algorithm

Ciper employs beam search through the space of possible equations using the search algorithm presented in Table 2. The algorithm takes as input a training data set D containing the values of independent variables and the dependent variable v_d. The output of Ciper consists of the b best polynomial equations according to the MDL heuristic function defined in the previous section.

Before the search procedure starts, the beam Q is initialized with the simplest possible polynomial equations of the form v_d = const. The value of the constant parameter const is fitted against the training data D using linear regression. In each search iteration, the refinements of the equations in the current beam are generated (using the refinement operator from Table 1) and collected in Q_r (line 5). In case when redundant equations are generated due to the sub-optimality of the refinement operator, the duplicate equations are filtered out from the set Q_r (each refined equation structure is compared to the equations from the current version of Q_r: if it is already included in Q_r, we skip it and proceed with the next refinement). Again, linear regression is used to fit the constant parameters of the refinements against the training data D (lines 6-8). Finally, at the end of each search iteration, only the best b equations, according to the MDL heuristic function, are kept in the beam (line 9). The search stops when the performed iteration does not change the beam.

3.5 Stepwise regression and MARS

The CIPER search algorithm is similar in spirit to the forward stepwise method for linear regression (Hastie et al., 2001). As CIPER, the stepwise regression method also starts with the simplest model $v_d = \text{const}$ and sequentially adds those independent variables to the model that most significantly improve its fit to the training data. To avoid overfitting, stepwise regression methods test the significance of the MSE improvement gained by refining the current equation and do not take into account those refinements that do not lead to significant improvements. The significance of the MSE improvement is based on F statistic:

$$F = \frac{\text{MSE}(v_d = P) - \text{MSE}(v_d = P')}{MSE(v_d = P')} \cdot (m - r - 2),$$

where $v_d = P$ is the current equation, $v_d = P'$ is the candidate equation with the newly added term, r is the number of terms in the current equation, and m is the number of training examples. The improvement is significant, if the obtained F value is greater than the 95th percentile of the $F(1, m - r - 2)$ distribution (Hastie et al., 2001). Stepwise regression method proceed with greedy search by choosing the best significant improvement and stops, if no significant improvement is available.

CIPER can be viewed as a stepwise method for polynomial regression with MDL heuristic function. However, there are several other important differences between CIPER and the stepwise regression method.

The refinement operator used in CIPER is better suited for polynomial regression. While the stepwise regression method can only refine the current equation by adding a new term to it, CIPER can also add a variable to an existing term in the current equation. Using this second kind of refinement, CIPER can generate polynomials of arbitrary degree. On the other hand, to use forward stepwise method for polynomial regression, terms of degree two and more have to be precomputed and introduced as new independent variables. However, this is a serious limitation of the stepwise method, since precomputation of higher degree terms requires user to specify their maximal degree of the introduced terms and it introduces potentially huge number of independent variables. The number of independent variables is of order $\mathcal{O}(|V|^d)$, where d is the maximal degree of precomputed terms.

The huge number of precomputed higher degree terms is reflected in the high branching factor of the stepwise refinement operator. Since it adds a new term to the current equation, its branching factor equals the number of independent variables, i.e., $\mathcal{O}(|V|^d)$. Note that the branching factor of CIPER's refinement operator ($\mathcal{O}(|V|r)$) is linear with regards to the number of independent variables. The lower branching factor of the refinement operator permits the use of higher beam widths in CIPER, which is in contrast with beam width of one used for stepwise regression methods.

Similar refinement operator has been also used in the MARS (multivariate adaptive regression splines) method (Friedman, 1991). The difference is however that the MARS refinement operator adds all possible *piecewise* linear terms of a form $\max(v - t, 0)$ or $\max(t - v, 0)$, where v is an independent variable $v \in V \setminus \{v_d\}$ and t is one of its values, to a current equation. Since each example in the training set

Table 3: Properties (number of variables n, number of examples m, and class variance VAR(v_d)) of the thirteen regression data sets used in the experiments.

Dataset	m	n	VAR(v_d)
autoprice	159	15	$3.433 \cdot 10^7$
baskball	96	4	0.01173
bodyfat	252	14	69.76
elusage	55	3	565.96
fruitfly	125	5	250.12
housing	506	14	84.42
mbagrade	61	3	0.1063
pollution	60	16	3805.13
pwlinear	200	11	19.92
quake	2178	4	0.03587
sensory	576	12	0.6758
strike	625	7	313837
veteran	137	8	24724.3
vineyard	52	4	18.94

defines a potential break point (knot) t in the piecewise linear term, the branching factor of MARS refinement operator is much higher since it also depends on the number of examples in the training set m. The branching factor of MARS refinement operator is of order $\mathcal{O}(|V|rm)$, which can be quite prohibitive for large data sets.

4 Experimental evaluation

The main goal of the performed experiments is to evaluate the predictive performance of equation discovery method Ciper especially in comparison with the standard regression methods for inducing linear and piecewise models, implemented in the data mining suite Weka (Frank and Witten, 1999). The performance of the methods is evaluated on thirteen data sets from the UCI Repository (Blake and Merz, 1998). These data sets have been widely used in other comparative studies. Table 3 presents the basic properties the data sets.

4.1 Experimental methodology and settings

In all the experiments presented here, regression performance is estimated using 10-fold cross validation. The regression performance is measured in terms of RE defined as:

$$RE = \frac{\sum_{i=1}^{m}(v_d(i) - \hat{v_d}(i))^2}{\sum_{i=1}^{m}(v_d(i) - \overline{v}_d)^2},$$

where $v_d(i)$ and $\hat{v_d}(i)$ are the observed and predicted values of the dependent variable for the i-th training example, m is the number of examples, and $\overline{v}_d$ is the average

Table 4: Predictive performance of CIPER in terms of relative mean squared error (RE), as compared to three regression methods implemented in WEKA: linear regression LR, regression trees RT, and model trees MT.

Dataset	CIPER	LR	RT	MT
autoprice	0.1503	0.2345	0.3242	0.1463
baskball	0.6116	0.6651	0.7845	0.6310
bodyfat	0.0282	0.0273	0.1097	0.0252
elusage	0.1751	0.2254	0.4365	0.2780
fruitfly	1.0074	1.1025	1.0132	1.0258
housing	0.1912	0.2891	0.2750	0.1666
mbagrade	0.7760	0.8342	1.0505	0.8342
pollution	0.5456	0.5513	0.7741	0.4187
pwlinear	0.1484	0.2479	0.3266	0.1062
quake	0.9959	0.9982	1.0010	0.9926
sensory	0.8879	0.8688	0.8463	0.7548
strike	0.9489	0.8386	0.8665	0.8295
veteran	0.8970	0.9226	0.9077	0.8789
vineyard	0.2869	0.4336	0.7281	0.4776
Average	0.5465	0.5885	0.6746	0.5404

value of the dependent variable. Note that RE gives a normalized value of the mean squared error, that is independent on the magnitude of the dependent variable v_d. The normalization allows for comparison and aggregation of the performance measure across different data sets.

We compare the performance of our approach based on polynomial equations to the performance of three standard regression methods implemented in WEKA (Frank and Witten, 1999): linear regression, regression trees, and model trees. The tree-based models are induced with the M5' algorithm (Wang and Witten, 1997). All algorithms have been used with their default parameters' settings. The default beam width in CIPER is 16.

4.2 Experimental results

The results of the experiments are presented in Tables 4 and 5. The first table compares the regression methods in terms of their predictive error and the second one compares the complexity of the induced models.

From the results on predictive accuracy of the models induced with different regression methods in Table 4, we can see that our approach based on polynomial equations performs better than linear regression and regression trees. CIPER performs much better then regression trees on smaller data sets. A possible explanation for this is that CIPER induces a single equation/model over the entire data set, as opposed to a number of partial models induced for data subsets in regression trees. Finally, to our surprise, the overall accuracy of CIPER is comparable to the accuracy of model trees. Note that we also compared the predictive error of models induced using CIPER with MDL heuristic to the error of models induced using CIPER with

Table 5: Complexities of the models induced with CIPER as compared to the linear and tree-based models in terms of number of constant parameters in the equation #P, polynomial length LEN and degree DEG, as well as number of decision nodes #DN for tree-based piecewise models.

Dataset	CIPER			LR	RT		MT	
	DEG	#P (r)	LEN	#P	#DN	#P	#DN	#P
autoprice	2	5	5	16	7	8	6	19
baskball	1	3	2	5	1	2	0	3
bodyfat	3	8	11	15	15	16	5	12
elusage	2	3	3	13	2	3	1	6
fruitfly	0	1	0	7	0	1	0	1
housing	4	15	32	14	25	26	18	56
mbagrade	1	3	2	3	0	1	0	3
pollution	1	5	4	16	6	7	0	10
pwlinear	2	10	12	11	13	14	1	12
quake	1	2	1	4	0	1	5	10
sensory	2	4	4	26	7	8	3	27
strike	1	4	3	23	9	10	7	22
veteran	1	2	1	10	1	2	0	3
vineyard	2	4	4	4	3	4	1	6
Average	1.64	4.93	6.00	11.93	6.36	7.36	3.36	13.57

F statistics as heuristic (used in stepwise regression methods, see Section 2.5). The results, which due to lack of space are not included in the paper, show that CIPER with MDL outperforms CIPER with F.

Furthermore, Table 5 presents a comparison of the complexities of different models. The complexity of polynomial equations is assessed in terms of polynomial degree DEG, number of terms r (which equals the number of constant parameters in the equations #P), and polynomial length LEN, defined in Section 3.1. The complexity of linear regression is assessed in terms of number of constant parameters. Finally, the complexity of tree models is measured in number of decision nodes and number of constant parameters used in the leaf nodes. While regression trees use one constant parameter in each leaf node, model trees use linear regression model in each leaf node. Thus, the number of the constant parameters in the model trees is higher compared to the number of constant parameters in the regression trees.

The results show that the average complexity of the polynomial models compares favorably with complexities of the other regression methods. The average model tree consists of 3 decision nodes and includes 14 constant parameters in the leaf nodes. This means that the average number of constant parameters in the model tree is 17, since each decision node include one constant parameter (the splitting boundary). Similarly, the average number of constant parameters in regression tree models is 13. Both numbers are significantly higher than the average of 6 parameters in the polynomial models, even if the average length of the equations (5) is taken into account.

5 Summary and further work

This paper presents CIPER, a method for efficient induction of polynomial equations that can be used as predictive regression models. CIPER employs heuristic beam search through the space of candidate polynomial equations. The search is based on a refinement operator with low branching factor that makes it much more suitable for polynomial regression compared to much complex refinement operators used in stepwise regression methods and MARS. Evaluation of CIPER on a number of standard predictive regression tasks shows that it is superior to linear regression and stepwise regression methods as well as regression trees. CIPER appears to be competitive to model trees too. The complexity of the induced polynomials, in terms of number of parameters, is much lower than the complexity of piecewise models.

First direction for further and ongoing work is an extensive empirical evaluation of CIPER on which involves larger sample of data sets, its comparison with other methods, such as MARS (Friedman, 1991) and kernel methods for regression (Hastie et al., 2001), and bias-variance analysis of the predictive error (Geman et al., 1992). Other directions for further research include integration of efficient methods for partitioning the data set in CIPER and use them to induce piecewise polynomial models (one piece for each partition). The partitioning of the data set can be based on Euclidean proximity of training examples: clustering methods can be used for this purpose as in Torgo and da Costa (2000) and Falkenhainer and Michalski (1990). Finally, since linear regression method is used for fitting the model parameter in CIPER, it is fairly straightforward to develop a version of CIPER that is capable of incremental induction of regression models from numeric data streams. The development can be based on the incremental linear regression method presented in Chen et al. (2002).

Acknowledgments

This work was supported in part by the project cInQ (Consortium on discovering knowledge with Inductive Queries), funded by the European Commission under the FET arm of the IST programme.

References

[1] Blake, C.L. and Merz, C.J. (1998): UCI repository of machine learning databases. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[2] Chen, Y., Dong, G., Han, J., Wah, B., and Wang, J. (2002): Multidimensional regression analysis of time-series data streams. In *Proceedings of the Twentyeighth International Conference on Very Large Data Bases*, 323-334, San Mateo: Morgan Kaufmann.

[3] Džeroski, S. and Todorovski, L. (1995): Discovering dynamics: from inductive logic programming to machine discovery. *Journal of Intelligent Information Systems*, **4**, 89-108.

[4] Falkenhainer, B. and Michalski, R. (1990): Integrating quantitative and qualitative discovery in the abacus system. In Y. Kodratoff and R. Michalski (Eds.): *Machine Learning: An Artificial Intelligence Approach.* San Mateo: Morgan Kaufmann.

[5] Frank, E. and Witten, I.H. (1999): *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* San Mateo: Morgan Kaufmann.

[6] Friedman, J. (1991): Multivariate adaptive regression splines (with discussion). *Annals of Statistics*, **19**, 1-141.

[7] Geman, S., Bienenstock, E., and Doursat, R. (1992): Neural networks and the bias/variance dilemma. *Neural Computation*, **4**, 1-58.

[8] Hastie, T., Tibshirani, R., and Friedman, J. (2001): *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Berlin: Springer.

[9] Langley, P., Simon, H.A., Bradshaw, G.L., and Żythow, J. M. (1987): *Scientific Discovery.* Cambridge: MIT Press, Cambridge.

[10] Todorovski, L. and Džeroski, S. (1997): Declarative bias in equation discovery. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 376-384, San Mateo: Morgan Kaufmann.

[11] Torgo, L. and da Costa, J.P. (2000): Clustered partial linear regression. In *Proceedings of the Eleventh European Conference on Machine Learning*, 426-436. Berlin: Springer.

[12] Wang, Y. and Witten, I.H. (1997): Induction of model trees for predicting continuous classes. In *The Proceedings of the Poster Papers of the Eighth European Conference on Machine Learning*, 128-137, University of Economics, Faculty of Informatics and Statistics, Prague.

[13] Washio, T. and Motoda, H. (1997): Discovering admissible models of complex systems based on scale-types and identity constraints. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, **2**, 810-817, San Mateo: Morgan Kaufmann.