

# Multitarget Polynomial Regression with Constraints

Aleksandar Pečkov, Sašo Džeroski, and Ljupčo Todorovski

Jozef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia

**Abstract.** The paper addresses the task of multi-target polynomial regression, i.e., the task of inducing polynomials that can predict the value of more than one numeric variable. As in other learning tasks, we face the problem of finding an optimal trade-off between the complexity of the induced model and its predictive error. We propose a minimal description length scheme for multi-target polynomial regression, which includes coding schemes for polynomials and their predictive errors on training data. The proposed MDL scheme is implemented in an algorithm for polynomial induction that can also take into account language constraints, i.e., constraints on terms to be included in the induced polynomials. We empirically compare the multi-target model with the multiple single target models. The results of the experiments show that there is no loss in predictive performance when using multi-target models as compared to multiple target models and that fewer equation structures are considered in the former case.

## 1 Introduction

Regression models are used to predict the value of a dependent numeric variable from the values of independent (predictor) variables. Commonly used regression models include linear regression and regression trees [1]. While the linear regression method tries to find a global model of the data (a linear equation), regression trees are piecewise models that partition the data space into a number of sub-spaces and induce a simple constant or linear model in each of them. While linear models tend to be oversimplistic, regression trees can sometimes overfit the training data. In this paper we address the task of polynomial regression, i.e., the task of inducing polynomial equations from numeric data that can be used to predict the value of a numeric variable. Polynomials can also overfit the data. Namely, it is well known that a data set of  $n$  points can be perfectly interpolated (and often overfitted) with a polynomial of  $(n - 1)$ -th degree.

In order to address the problem of overfitting, different approaches to model selection have been proposed in the literature [3] (pages 193-222). Each approach tries to find an optimal trade-off between the complexity of the induced model and its predictive error and thus avoid overfitting. The minimal description length (MDL) principle is one of them. Following the MDL principle, the quality of a model is estimated by combining the estimate of the model complexity and the predictive error the model makes on the training data. The

complexity of the model and the error are measured in terms of the number of bits necessary for encoding them.

In this paper we first address the task of polynomial regression and present a MDL encoding scheme for single target polynomial model. We compare this encoding scheme with an Akaike like ad-hoc encoding scheme. Also we compare the better to liner regression, regression trees, and model trees. Then we extend our approach for multi-target regression by extending our encoding scheme for multi-target models. Finally, we empirically compare the multi-target approach with the single target approach.

## 2 Polynomial Regression

The task of polynomial regression, is to induce a polynomial equation from numeric data that can predict the value of a numeric variable.

Every polynomial over variables  $x_1, x_2, \dots, x_n$  can be written in the form:

$$P = c_0 + \sum_{i=1}^m c_i \cdot T_i$$

where  $T_i = \prod_{j=1}^n x_j^{a_{i,j}}$ ,  $c_i, i = 1..m$  and  $c_0$  are constants and  $c_i \neq 0$ . We say  $T_i$  is a *term* or *monomial* in  $P$ . The length of  $P$  is defined as  $Len(P) = \sum_{i=1}^m \sum_{j=1}^n a_{i,j}$ , while the size of  $P$  is  $Size(P) = m$ ; and the degree of  $P$  is  $Deg(P) = Max_{i=1}^m \sum_{j=1}^n a_{i,j}$ .

An example polynomial equation is  $P = 1.2x^2y + 3.5xy^3 + 5xz + 2$ . This equation has size 3 (it has three terms), degree 4 (the maximal term degree is 4) and length 9.

Ciper [7] (Constrained Induction of Polynomial Equations for Regression) is a beam search algorithm that heuristically searches through the space of candidate polynomial equations for the ones that fit the data best.

The top-level outline of Ciper algorithm is shown in Table 1. First, the beam is initialized either with the simplest polynomial equation  $P = c$ , or with a minimal polynomial that follows the given constraints (the constraints will be described below). In every search iteration, a set of polynomials is generated from the beam using a refinement operator. The coefficients before the terms are fitted using linear regression. For each of the polynomials, the value of the minimal description length (MDL) heuristics is calculated. At the end of the iteration, the equations with smallest MDL values are retained in the beam. The evaluation stops when the refinement operator can not generate new equations or when the content of the beam was unchanged in the last iteration. Such situation occurs when every polynomial that is generated in the last iteration has worse MDL estimate than the polynomials already in the beam.

The refinement operator increases the length of an equation by one, either by adding a first degree term or by multiplying an existing term by a variable (Figure 1). Starting with the simplest equation, and iteratively applying the refinement operator, all polynomial equations can be generated.

**Table 1.** A top-level outline of the CIPER algorithm.  $Q$  and  $Q_r$  are sets of equations (the beam).

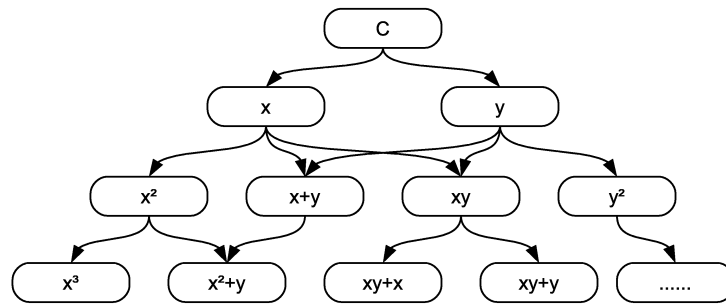
---

```

procedure CIPER(Data, InitialPol, Constraints)
  InitialPol = FITPARAMETERS(InitialPol, Data)
   $Q = \{InitialPol\}$ 
  repeat
     $Q_r =$  refinements of (some) equation structures in  $Q$ 
    foreach equation structure  $E \in Q_r$  do
       $E =$  FITPARAMETERS( $E$ , Data)
    endfor
     $Q = \{best\ b\ equations\ from\ Q \cup Q_r\}$ 
  until  $Q$  unchanged during the last iteration
  print  $Q$ 

```

---



**Fig. 1.** The Ciper refinement operator.

Ciper can take into account two types of constraints on the form of the induced polynomial equations:

- *Language constraints* specify structural bounds on the polynomial structures considered during search:  $P_L$  and/or  $P_U$ . These constraints specify that every candidate equation  $P$  should be a super-polynomial of  $P_L$ , while  $P_U$  should be super-polynomial of every  $P$ . A polynomial  $P$  is a subpolynomial from a polynomial  $P'$  if for every term  $T$  in  $P$  exists a term  $T'$  in  $P'$  such that the degree of every variable in  $T$  is larger than or equal to the degree of the same variable in  $T'$ .
- *Complexity constraints* constrain the complexity of a polynomial with specifying the upper bound for the polynomial length, degree, or size.

Our approach is based on the single target Ciper algorithm can take into account some constraints during induction, most noticeably, language constraints. This capability is preserved in the multi-target version of Ciper. In the next section, we describe the heuristic function used to evaluate every candidate equation.

### 3 Minimal Description Length

Following the minimal description length (MDL) principle, among the number of candidate models, we select the one that represents a good trade-off between model's predictive error its complexity. The MDL principle combines two ideas (or assumptions) about relation between learning and data compression:

- regularities in the data can be used to compress the data, i.e., the more regularities there are, the more the data can be compressed;
- the more we are able to compress the data, the more we have learned about the data.

Thus, complexity of the model can be estimated as its ability to compress data: larger the compression, smaller the complexity of the obtained model. More specifically, MDL estimate of the model quality is composed of two components:

$$MDL(H) = L(H) + L(D|H),$$

where the first component  $L(H)$  corresponds to the length of the encoding of model (hypothesis)  $H$ , while the second one  $L(D|H)$  is the length of the description of the data when encoded using the model  $H$ .

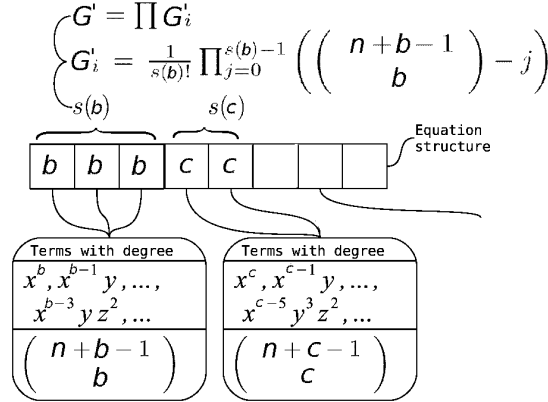
#### 3.1 Encoding polynomial structure

In order to encode the structure of polynomials, we follow the refined MDL approach [10]. We first partition the space of candidate models into subgroups  $\mathcal{H}_c$  of models with equal complexity  $c$ . A particular model  $H \in \mathcal{H}_c$  can be then encoded using  $N = \log|\mathcal{H}_c|$  (note that  $\log$  stands for binary logarithm) bits, where  $|\mathcal{H}_c|$  denotes the number of models in the class  $\mathcal{H}_c$ .

In case of polynomials, we are going to partition the space of candidate polynomial structures in classes at several levels. At the highest level, we'll group together the candidate polynomials with same length  $l$  and same number of terms (size)  $m$ . We'll refer to these classes as  $G(m, l)$ ; for example  $G(1, 1)$  contains polynomial structures with one linear term, while  $G(1, 2)$  contains polynomial structures with only one term of second degree. Note that  $m \leq l$ . On the second level, we partition each  $G(m, l)$  in subgroups with fixed term degrees  $G'(a_1, a_2, \dots, a_m)$ . Polynomials in this subgroup have  $m$  terms with degrees  $a_1 \geq a_2 \geq \dots \geq a_m$ . Note that  $\sum_{i=1}^m a_i = l$ . Now we have to calculate how many sub-groups  $G'$  there are in a single  $G(m, l)$  group and also calculate how many polynomial structures there are in each  $G(a_1, a_2, \dots, a_m)$  group.

The number  $|G'(a_1, a_2, \dots, a_m)|$  can be easily calculated using a procedure roughly depicted in Figure 2. Given the degree of the first term  $a_1$  we have to choose  $a_1$  variables from the set  $\{x_1, x_2, \dots, x_n\}$  where variables can appear in the selection more than once. Thus, the number of possibilities for the first term equals the number of combinations with repetition where we select  $a_1$  elements from a set of given  $n$ . This number equals  $\binom{n+a_1-1}{a_1}$ . Continuing the same reasoning for all  $m$  terms, gives us the number of possible structures in

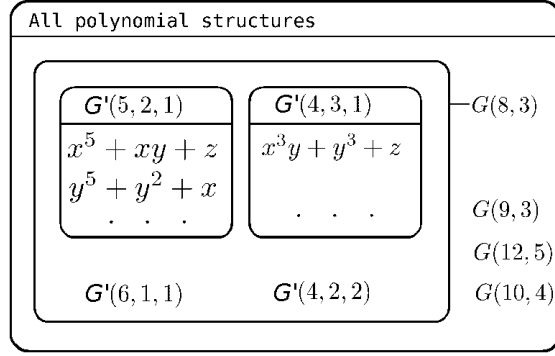
$G'(a_1, a_2, \dots, a_m)$  to be  $\prod_{i=1}^m \binom{n+a_i-1}{a_i}$ . However, if there are several  $a_i$  values that are equal, we will encounter the same term many times, which means that the above formula over-estimates the number of possible structures. The remedy is to divide the number with the factorial of repetitions observed in the tuple. For example, when dealing with case  $G'(5, 5, 3, 2, 2, 2)$ , we have to divide with  $2!3!$ , since value 5 is repeated twice and value 2 is repeated three times. Note also that each multiplicative term decreases by 1 for each degree value repetition (see Figure 2).



**Fig. 2.** Calculating the number of polynomial structures in  $G'(a_1, a_2, \dots, a_m)$ . At the bottom, we have the sets of terms (one with terms of degree  $b$  and one with terms of degree  $c$ ). In the middle layer, they are combined in equation structures, where  $s(b)$  and  $s(c)$  denote the number of repetitions of  $b$  and  $c$  values respectively.

Having the number of equation structures in each  $G'$  group, we now turn to a problem of calculating the number of  $G'$  groups within each  $G(m, l)$ . The size of  $G$  grows according to the recursive formula  $|G(m, l)| = |G(m - 1, l - 1)| + |G(m, l - m)|$ . The first additive term corresponds to the cases when the  $G'$  groups contain linear terms (there is  $a_i$  with value 1), while the second corresponds to the cases when the  $G'$  groups where all  $a_i > 1$ . In the first case, when removing the linear term, we obtain polynomials with  $m - 1$  terms and length  $l - 1$ . In the second case, we can remove one variable from all the terms, which leads to polynomials with same number of terms ( $m$ ) and length  $m - l$ . Figure 3 depicts he relationship between  $G$  and  $G'$  classes of polynomial structures.

Now, having this partitioning and number of polynomials in each partition, we can decompose the code for each candidate polynomial in four components. First, we have to encode its length  $l$  and for this we need  $\log(l) + 2\log(\log(l))$  bits (the second double logarithm term is necessary, since we do not know the magnitude of  $l$  in advance). Second, we encode the number of terms  $m$ , for which we need  $\log(l)$  bits (remember that  $m \leq l$ ). Third, we can identify a particular  $G'$  class within the class  $G(m, l)$  using  $\log(|G(m, l)|)$  bits. Finally, we identify the specific polynomial structure within  $G'$  using  $\log(|G'(a_1, a_2, \dots, a_m)|)$  bits.



**Fig. 3.** General overview of the partitioning of polynomial structures. The small sets correspond to  $G'$  classes (e.g., the set  $G'(5, 2, 1)$ ). In turn, we group them into a larger classes of structures  $G$  that have same length and size.

Putting these four components together gives us the final formula:  $L(H) = 2\log(l) + 2\log(\log(l)) + \log(|G(n, l)|) + \log(G'(a_1, a_2, \dots, a_n))$ .  
for number of bits necessary to encode the polynomial structure.

### 3.2 Encoding Data

Rissanen provides a formula for calculating the stochastic complexity of a model obtained using linear regression [5]:

$$W = \min_{\gamma} \{ (N - k) \log(\hat{\tau}) + k \log(\hat{R}) + (N - k - 1) \log\left(\frac{1}{N - k}\right) - (k - 1) \log(k) \}$$

where  $\gamma$  index goes through the all possible subsets of variables involved in the linear regression,  $k$  is the number of elements in  $\gamma$ ,  $N$  is the size of the dataset,  $\hat{\tau}$  is the maximum likelihood estimation of the model error, and  $\hat{R} = \frac{1}{n} \hat{c}^T (X^T X)^{-1} \hat{c}$  ( where  $\hat{c} = (X^T X)^{-1} X^T y$  ). The stochastic complexity of the model then is  $2W$ . Intuitively, this corresponds to the length of the code necessary to encode the errors of the linear regression model ( $L(D|H)$ ) together with the constant parameters of the linear model. The later is closely related to the model error and thus can not be encoded separately, that is together with the model structure, which is what is mostly done when using (ad-hoc) MDL principle in machine learning algorithms. For further details, see [5].

## 4 Single-target Polynomial Regression

The task of single-target polynomial regression is to induce a polynomial equation from numeric data that can predict the value of a single numeric variable. Ciper original implementation used an ad-hoc MDL based heuristics function. In this section we will present the ad-hoc heuristics and compare it with the improved MDL heuristics (Section 3) on single target datasets.

### 4.1 Ad-hoc MDL

The ad-hoc MDL heuristic function is given by

$$MDL(P) = \text{len}(P) \cdot \log(m) + m \cdot \log(MSE(P))$$

where  $P$  is the polynomial equation being evaluated,  $len(P)$  is its length,  $MSE(P)$  is its mean squared error, and  $m$  is the number of training examples.

This evaluation function is based on the Akaike and Bayesian information criteria for regression model selection [3]. The second term of the ad-hoc MDL heuristic function measures the degree of fit of a given equation and the first term introduces a penalty for the complexity of the equation. With this penalty the MDL heuristic function introduces preference toward simpler equations.

## 4.2 Empirical evaluation for the heuristic functions

The main goal of the performed experiments is to evaluate the predictive performance of Ciper using the different heuristics described above. We compared them with the standard regression methods, implemented in the data mining suite Weka [9]. The performance of the methods is evaluated on fifteen data sets from the UCI Repository [4] and another publicly available collection of regression data sets [8]. These data sets have been widely used in other comparative studies.

In all the experiments presented here, we estimate the predictive performance on unseen examples using 10-fold cross validation. The predictive performance of a model  $M$  is measured in terms of relative root mean squared error ( $RRMSE$ ). The Ciper algorithm that uses ad-hoc MDL heuristic will be referred to as ad-hoc Ciper, and the Ciper algorithm that uses the improved MDL heuristic will be referred to as MDL Ciper.

The last two columns in Table 2 give the performance comparison between ad-hoc and improved MDL. It is noticeable that MDL Ciper performs better than ad-hoc Ciper. The statistical significance is tested using a paired t-test. If the p-value is smaller than 0.05 then we reject the null hypothesis, and conclude that the difference is statistically significant. The + sign in the table is used when the improvements we introduce perform significantly better and the – sign is used when they perform worse.

We found that ad-hoc Ciper never performs better and MDL Ciper performs better on six datasets. We can conclude that Ciper with MDL heuristics performs better than Ciper using ad-hoc heuristics. Ciper clearly outperforms linear regression, and performs much better than regression trees on more datasets. Also Ciper is comparable to model trees (Ciper was better on three and model trees on three datasets).

## 5 Multitarget Polynomial Regression

The task of multi-target polynomial regression is to induce a polynomial equation from numeric data that can predict the value of several numeric variables.

A multi-target polynomial model can be defined as:

$$P = C_0 + \sum_{i=1}^m C_i \cdot T_i$$

**Table 2.** Predictive performance in terms of relative root mean square error of commonly used regression methods implemented in Weka: linear regression (LR), model trees (MT), and regression trees (RT). Also comparison of ad-hoc Ciper and MDL Ciper.

Data set	LR	RT	MT	ad-hoc	mdl
2dplanes	0.5427 +	0.2272	0.2270	0.2270	0.2270
autoprice	0.4715 +	0.5426 +	0.3659	0.4128	0.3815
bank32nh	0.6858	0.7512 +	0.6739	0.8119 +	0.6751
basketball	0.7737 +	0.8850 +	0.7737	0.7784	0.7738
bodyfat	0.1643	0.3293 +	0.1557 -	0.2834	0.1663
cal-housing	0.6037	0.5177 -	0.4778 -	0.5903 +	0.5767
cpu-small	0.5371 +	0.2247 -	0.1738 +	0.4161 +	0.1628
elusage	0.4781 +	0.6560 +	0.4372	0.4009	0.4009
fried-delve	0.5265 +	0.3550 +	0.2780 +	0.2903 +	0.1996
house-8l	0.7869 +	0.6250	0.5929	0.6097	0.6123
housing	0.5281 +	0.5095 +	0.4286	0.4264	0.4184
kin-8nm	0.7661 +	0.6837 +	0.6093 +	0.8463 +	0.5570
mv	0.4309 +	0.0475 +	0.0131 -	0.0440 +	0.0214
pw-linear	0.4954 +	0.5640 +	0.3258	0.3301	0.3310
vineyard	0.7133	0.8617	0.7458	0.5254	0.6749

where  $T_i = \prod_{j=1}^n x_j^{a_{i,j}}$  are the terms. Here  $C_i$ ,  $i = 1..m$  and  $C_0$  are constant vectors (not constants) and  $C_i \neq \mathbf{0}$ . The number of coordinates of these vectors is the number of targets we want to predict.

An example of a multi-target polynomial equation is  $P = (1, 2, 5) \cdot x^2y + (3, 5, 7) \cdot xy^3 + (2, 3, 10)$ . It is equivalent to three single target polynomial equations  $P_1 = 1 \cdot x^2y + 3 \cdot xy^3 + 2$ ;  $P_2 = 2 \cdot x^2y + 5 \cdot xy^3 + 3$ ;  $P_3 = 5 \cdot x^2y + 7 \cdot xy^3 + 10$ , i.e  $P = (P_1, P_2, P_3)$  This equation model predicts three targets.

In this way a single equation can be used for predicting more targets. The idea is that the complexity of this equation will be smaller then the complexity of a set of equations (one equation per each target). If the single target equations depend on the same term then we will need less to encode the multi-target model then the single target models. Also, in this case we may obtain a model that have better predictive capabilities because the risk of overfitting will be smaller.

The Ciper algorithm for multi-target polynomial regression goes just the same as in the single target case (Table 1 ). The data can be represented as a matrix  $M$ , where the number of rows is the number of instances, and the number of columns is the number of terms plus one (the first column is filled with ones). We calculate the coefficients  $C_i$  of the equation as

$$C = (M^T \cdot M)^{-1} \cdot (M^T \cdot Y)$$

where  $Y$  is the matrix of values we are trying to predict. We have introduced some optimizations for obtaining the coefficients. In this equation the multiplication is computationally expensive because of the large number of rows. If we have

terms  $T_1$ ,  $T_2$ ,  $T_3$  and  $T_4$ , such that  $T_1 \cdot T_2 = T_3 \cdot T_4$ , then the appropriate elements in matrices  $M_{T_1, T_2}^T \cdot M_{T_1, T_2}$  and  $M_{T_3, T_4}^T \cdot M_{T_3, T_4}$  are equal. We store all generated elements from the matrices  $M^T \cdot M$ . We use it later to calculate the matrices of the subsequently generated polynomials. Even more this matrices are the same for every target. This optimization considerably lowers the amount of calculations.

Notice that the language and the complexity constraints mentioned in Section 2 can equally be used in the multi-target case.

Because the structure of the polynomial hasn't changed, the complexity is the same for the multi-target case like in the single target case. Summing the stochastic complexities of the linear model for each target with the complexity of the structure we have the total complexity of the multi-target model.

## 6 Empirical Evaluation

### 6.1 Real datasets

The main goal of the performed experiments is to evaluate the predictive performance of Ciper on multi-targets datasets. The performance of the methods is evaluated on five data sets EDM, SIGMEA-REAL, and SIGMEA-SIM.

EDM dataset describes 154 actions taken by a human operator controlling two variables (target variables). It contains eight numeric attributes from which two are the target attributes. SIGMEA-REAL dataset collects 817 measurements of the rate of herbicide resistance of two lines of plants (target variables). It has eight attributes. SIGMEA-SIM dataset describes the effects of the individual field characteristics and cropping systems on the rate of pollen and the seed dispersal rate, (target variables). Dataset includes 10368 cases and it has 13 attributes.

A multi-target model is build for each dataset and a single target model for each of the targets. The predictive performance of a single target model is measured in terms of relative root mean squared error (*RRMSE*). The predictive performance of a multi-target model is presented as an array of the predictive performance of the appropriate single target models.

In all the experiments presented here, we estimate the predictive performance on unseen examples using 10-fold cross validation. The statistical significance is tested using a paired t-test. If the p-value is smaller than 0.05 then we reject the null hypothesis, and conclude that the difference is statistically significant. The + sign in the table is used when the improvements we introduce perform significantly better and the - sign is used when they perform worse.

The results suggest that there is no significant difference in the predictive capabilities whether we make single target models or multi target models.

We counted the number of equations generated with Ciper. It seems that the number of equations generated for multi-target Ciper is usually smaller than the total number of equations needed for building the single target models. The number of equations in the table 3 is averaged from the number of generated equations from the 10 folds.

**Table 3.** Comparison of predictive performance of multi-target and single-target Ciper in terms of relative root mean square error, and number of tested equations.

dataset	target	multi-target	single-target	num. eqs.
EDM	1	0.90703	0.86435	31152
	2	0.79134	0.75032	415
num. tested eqs.		458.0		3156.7
SIGMEA-SIM	1	0.07441	0.07509	139045.1
	2	0.06141	0.06158	157931.2
num. tested eqs.		202345.9		296976.3
SIGMEA-REAL	1	0.71846	0.71190	1497.2
	2	0.64948	0.63467	1812.6
num. tested eqs.		2265.6		3309.8

## 6.2 Using constraints in modeling chemical reactions

To illustrate the use of constraints in discovering dynamics, we address the task of reconstructing a partially specified network of chemical reactions. The part of the network given in bold is assumed to be unknown, except for the fact that  $x_6$  and  $x_7$  are involved in the network. This is a task of revising an equation-based model. A network of chemical reactions can be modeled with a set of polynomial differential equations. The reaction rate of a reaction is proportional to the concentrations of inputs involved (product, e.g.  $x_5 \cdot x_7$ ). It influences the rate of change of all inputs (negatively) and all outputs (positively). The equation structures (left) / full equations (right), corresponding to the partial/full network, are given below.

Partial structure/Full equations	$\{x_5, \mathbf{x}_7\} \rightarrow \{x_1\}; \{x_1\} \rightarrow \{x_2, x_3\}$ $\{x_1, x_2, \mathbf{x}_7\} \rightarrow \{x_3\}; \{x_3\} \rightarrow \{x_4\}$ $\{x_4\} \rightarrow \{x_2, \mathbf{x}_6\}; \{\mathbf{x}_4, \mathbf{x}_6\} \rightarrow \{\mathbf{x}_2\}$
$\dot{x}_1 = -c \cdot x_1 + c \cdot x_5 - c \cdot x_1 \cdot x_2$	$\dot{x}_1 = 0.8 \cdot x_5 \cdot x_7 - 0.5 \cdot x_1 - 0.7 \cdot x_1 \cdot x_2 \cdot x_7$
$\dot{x}_2 = c \cdot x_1 + c \cdot x_4 - c \cdot x_1 \cdot x_2$	$\dot{x}_2 = 0.7 \cdot x_1 + 0.2 \cdot x_4 + 0.1 \cdot x_4 \cdot x_6 - 0.3 \cdot x_1 \cdot x_2 \cdot x_7$
$\dot{x}_3 = c \cdot x_1 + c \cdot x_1 \cdot x_2 - c \cdot x_3$	$\dot{x}_3 = 0.4 \cdot x_1 + 0.3 \cdot x_1 \cdot x_2 \cdot x_7 - 0.2 \cdot x_3$
$\dot{x}_4 = c \cdot x_3 - c \cdot x_4$	$\dot{x}_4 = 0.5 \cdot x_3 - 0.7 \cdot x_4 \cdot x_6$
$\dot{x}_5 = -c \cdot x_5$	$\dot{x}_5 = -0.6 \cdot x_5 \cdot x_7$
	$\dot{x}_6 = 0.2 \cdot x_4 - 0.8 \cdot x_4 \cdot x_6$
	$\dot{x}_7 = -0.1 \cdot x_1 \cdot x_2 \cdot x_7 - 0.1 \cdot x_5 \cdot x_7$

The full equations were simulated for 1000 time steps of 0.01 from a randomly generated initial state (each variable randomly initialized in the interval (0,1)), thus providing a trace of the behavior of the 7 system variables over time.

Subsumption constraints can be used in a natural way. A partially specified reaction network gives rise to equations that involve subpolynomials of the polynomials modeling the entire network.

The knowledge of the partial network can be used to constrain the search through the space of possible equations. The polynomial structures in the equations for  $x_1 \dots x_5$  in the partial network should be subpolynomials of the corresponding equations in the complete network. These subpolynomial constraints

were given to Ciper together with the behavior trace for all 7 variables. The subsumption constraint used in multi-target Ciper is  $x_1 + x_3 + x_1 \cdot x_2 + x_4 + x_5$ .

The generated multi-target model is

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \end{pmatrix} = \begin{pmatrix} -0.50 & 0.00 & 0.00 & 0.00 & -0.70 & 0.81 \\ 0.69 & -0.01 & 0.20 & 0.10 & -0.29 & 0.08 \\ 0.40 & -0.20 & 0.00 & 0.00 & 0.30 & 0.00 \\ -0.01 & 0.50 & 0.00 & -0.70 & 0.00 & 0.03 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.61 \\ 0.00 & 0.00 & 0.20 & -0.80 & 0.00 & 0.00 \\ 0.01 & 0.00 & 0.00 & 0.00 & -0.10 & -0.12 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_3 \\ x_4 \\ x_4 \cdot x_6 \\ x_1 \cdot x_2 \cdot x_7 \\ x_5 \cdot x_7 \end{pmatrix}$$

Ciper successfully reconstructs the equations for the entire network, i.e., for each of the 7 system variables. Discovery without constraints, however, fails for 3 equations. However the predictive error of the generated models without constraints is less than 1 percent as shown in table 4.

**Table 4.** Comparison of predictive performance of multi-target and single-target Ciper in terms of relative root mean square error, and number of tested equations.

dataset	target	multi-target	single-target	num. eqs.
DYN1	1	+ 0.00019	0.00340	71.0
	2	0.00539	0.00546	957.0
	3	0.00038	+ 0.00035	942.5
	4	+ 0.00202	0.02143	71.0
	5	0.00067	0.00067	71.0
	6	0.00121	0.00119	71.0
	7	+ 0.00219	0.01706	71.0
num. tested eqs.		2087.4		2254.5
DYN2	1	0.00020	0.00019	508.2
	2	0.00270	0.00269	514.1
	3	0.00046	0.00046	485.9
	4	0.00065	+ 0.00053	465.0
	5	0.00021	0.00021	36.0
num. tested eqs.		1177.6		2009.2

## 7 Discussion

We have proposed an approach for multi-target polynomial regression, i.e., the task of inducing polynomials that can simultaneously predict the values of several numeric target variables. To this end, we extend the Ciper system for polynomial regression that has been recently modified to employ a principled MDL heuristic in it search for polynomial equations: The latter is empirically shown to perform better on a number of datasets. We adapt this MDL heuristic to the multi-target case as well. The multi-target approach can also take into account language constraints, i.e., constraints on terms to be included in the induced polynomials.

We have empirically compared the multi-target approach to the application of several single-target models. The results of the experiments show that there is no loss in predictive performance when using multi-target models as compared to using multiple single-target models. In addition, the appearance of common terms in the equations for the different targets in the multi-target model makes these models more stable. Because the same equation structure must be good for all targets, and hence the risk of over-fitting is reduced.

Fewer equation structures are considered by multi-target Ciper. We have also included optimizations that use the fact that there are common calculations for each target. This makes the multi-target approach faster than the single-target approach where we build a model separately for each target.

In addition to applying the multi-target approach to several real-world problems, we also apply it to the task of system identification, i.e., discovering dynamics. Here the task is to induce a system of simultaneous differential equations that describe the behaviour of a system whose state changes over time: The state of the system typically consists of a vector of system variables. The time derivatives of the system variables are typically interdependent and expressed as functions (polynomials) of the system variables.

We have used this approach to discover dynamics of chemical reaction networks. In this domain, subsumption constraints have a natural interpretation. They can be used to specify, e.g., a partially known network as prior knowledge of chemical reactions. Constraints proved crucial for the successful reconstruction of an example network.

## References

1. Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth International, Belmont, Ca.
2. Grünwald, P., Myung, I., & Pitt, M. (Eds.). (2005). *Advances in minimum description length: Theory and applications*. Cambridge, Massachusetts: MIT Press.
3. Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. New York: Springer.
4. Newman, D., S. Hettich, C. B., & Merz, C. (1998). UCI repository of machine learning databases.
5. Rissanen, J. (1999). Mdl denoising. *IEEE Transactions on Information Theory*, 46, 2537–2543.
6. Robnik, M. (1998). Pruning regression trees with mdl. *Proceedings of the European Conference on Artificial Intelligence* (pp. 455–459). Brighton, UK: John Wiley and Sons.
7. Todorovski, L., Ljubič, P., & Džeroski, S. (2004). Inducing polynomial equations for regression. *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 441–452).
8. Torgo, L. (1998). Regression datasets.
9. Witten, I. H., & Frank, E. (Eds.). (2005). *Data mining: Practical machine learning tools and techniques*. San Francisco: Morgan Kaufmann.
10. Grünwald, P. D., Myung I. J., Pitt M. A. (Eds.) (2004) *Advances in Minimum Description Length: Theory and Applications*. MIT Press