ISSN 1407-7493

COMPUTER SCIENCE
DATORZINĀTNE

2008-34

APPLIED COMPUTER SYSTEMS
LIETIŠĶĀS DATORSISTĒMAS

AN APPROACH FOR POLYNOMIAL REGRESSION MODELLING USING CONSTRUCTION OF BASIS FUNCTIONS

POLINOMU REGRESIJAS MODELĒŠANAS PIEEJA PIELIETOJOT BĀZES FUNKCIJU KONSTRUĒŠANU

Gints Jēkabsons, *Riga Technical University,*
Institute of Applied Computer Systems,
Meza 1/3, LV-1048, Riga, Latvia, M.sc.ing., gintsj@cs.rtu.lv

Jurijs Lavendels, *Riga Technical University,*
Institute of Applied Computer Systems,
Meza 1/3, LV-1048, Riga, Latvia, as.prof. Dr.sc.ing., jurisl@cs.rtu.lv

Polynomial regression, subset selection, basis function construction, heuristic search, state space

1. Introduction

In regression modelling to describe the relation between variables commonly a polynomial regression model is used. Polynomials are very flexible and often used when there is no theoretical model available.

To obtain a polynomial regression model, which describes the relations in data sufficiently well and does not overfit, typically the subset selection approach [1] is used where the goal is to find the best subset of basis functions which gives the best predictive performance of the regression model. Before the subset selection step, in order to enrich the candidate model space, a finite set of *predefined* basis functions is created and, after that, the subset selection is performed with the basis functions. The basis functions typically are defined as products of the original variables each raised to some order (a positive integer).

Thus the goal is to find a subset that maximises the predictive performance of the resulting regression model. In order to find the subset some kind of search must be performed. The simplest search strategy is the exhaustive search. Although exhaustive search guarantees to find the best subset, it needs exponential runtime and thus is impractical in most cases.

Another class, called heuristic search methods, efficiently traverse the space of subsets, by adding and deleting the basis functions, and use an evaluation function that directs the search into areas of increased performance. The typical examples of heuristic search methods are the Forward Selection (also known as Sequential Forward Selection, SFS) and the Backward Elimination (also known as Sequential Backward Selection, SBS) [2]. SFS starts with an empty set of selected basis functions and iteratively adds the function leading to the highest performance increase to the set of selected functions, until the performance cannot be enhanced any further by adding a single function. SBS starts with the complete function set and iteratively removes the function whose removal yields the maximal performance increase.

The approach of subset selection assumes that the chosen *fixed* full set of *predefined* basis functions contains a subset which is sufficient to describe the target relation sufficiently well. However we argue that in most cases the necessary set of basis functions is not known and needs to be guessed or chosen by experience (e.g. by specifying the maximal order of the resulting polynomial). In many cases that means a non-trivial (and long) trial and error process that may generate sets of functions, working with which, in some problems of moderate dimensionality, may become computationally too demanding even for the heuristic search methods (as it will be demonstrated in the empirical experiments of this paper). A more convenient and efficient way would be to let the modelling method itself construct the basis functions necessary for creating the regression model with adequate predictive performance.

In this paper we consider a polynomial regression modelling approach with automatic construction of basis functions using heuristic search in the resulting infinite candidate model space. The approach does not require the user to predefine the set of basis functions for model creation. We also list five of the possible refinement operators, which allow the search to find better models as well as to do it more efficiently, and introduce an instance of the approach – a new regression modelling method called Sequential Floating Forward Polynomial Construction (SFFPC), which is named similarly to the subset selection method Sequential Floating Forward Selection (SFFS) [3] on which the search strategy of SFFPC is based.

The rather recently proposed method Constrained Induction of Polynomial Equations for Regression (CIPER) [4] also may be viewed as an instance of the approach. However it has some drawbacks regarding the set of the refinement operators used, which we tried to eliminate in our proposed polynomial regression modelling method.

To evaluate the considered approach in form of our proposed polynomial regression modelling method, SFFPC, we empirically compare it to two well known subset selection methods SFS and SFFS, as well as to CIPER both on artificial and real world data.

In the following section we shortly describe the subset selection approach and take a look at the characteristics of a search problem. Next we describe the approach with basis function construction, describe its relation with the subset selection approach, and discuss the differences and similarities of the search problem characteristics. After that we list five possible operators for refinement of polynomial regression models and shortly consider the applicability of existing search strategies. Then, we shortly describe an existing search method, CIPER, which may be viewed as an instance of the approach, and indicate its drawbacks. Finally we propose a new polynomial regression modelling method, SFFPC, which uses a specific set of model refinement operators to perform the search. An empirical comparison of the proposed method with the other three existing methods, SFS, SFFS, and CIPER, is presented in the last part of this paper.

2. Subset selection in polynomial regression modelling

A polynomial regression model may be defined by a linear summation of basis functions:

$$\hat{y} = \sum_{i=1}^{k} a_i f_i(x)$$

where a_i are model's parameters; k is the number of the *used* basis functions (equal to the number of model's parameters); and $f(x)$ are the basis functions which generally may be defined as a product of original input variables each raised to some order:

$$f_i(x) = \prod_{j=1}^{d} x_j^{r_{ij}} \quad (1)$$

where d is the number of the original variables; r_{ij} is the order of the j-th variable in the i-th basis function (a non-negative integer);. Note that when all r_j's of a basis function are equal to 0, we have the intercept term. Since polynomial regression models are linear in the parameters, the usual linear model tools may be applied – the parameters a_i of the regression models are estimated using the ordinary least-squares method, OLS.

Usually in the subset selection approach the basis functions are chosen such that the order of each possible polynomial model does not exceed a previously chosen highest allowed order p, i.e. each $r_{ij} \in \{0,1,...,p\}$ and $\sum_{j=1}^{d} r_{ij} \le p$ for all i. Then the number of all defined basis functions is

$$m = \prod_{i=1}^{p} (1 + d/i) \quad (2)$$

and the number of all possible subsets, from which we want to find the best, is equal to 2^m.

Summarizing [5,6,7], in order to characterize a heuristic search problem one must define the following: 1) initial state of the search; 2) available state-transition operators; 3) search strategy; 4) termination condition; 5) evaluation measure. Note that in the rest of this paper instead of the term "state-transition operator" we will use the term "(model) refinement operator" (as in [4]), which is somewhat more convenient in the context of regression modelling.

In context of polynomial regression subset selection typically the *initial states* are empty, full or randomly chosen subsets of basis functions; the available *refinement operators* are addition and deletion of any one basis function; the *search strategies* are the successive addition of basis function (as in SFS) or successive removal of them (as in SBS); the *termination* corresponds to finding of state in which none of the refinement operators can lead to a better state.

Concerning the *evaluation measure*, the evaluation of models corresponding to alternative subsets of basis functions, also known as model selection problem [8,9,10,11], is most commonly done in two ways: using complexity penalization criteria or resampling techniques. The former in contrast to the latter usually does not require high computational resources and allow one to use all the available data for training. In general one can define such criteria as a sum of deviance of the model and the complexity penalty: CR = *deviance* + *penalty*. In the least-squares regression problem the deviance term is equal to $n\log(SSE/N)$, where SSE is Sum of Squared Error in training data. Some of the most widely known and used complexity penalization criteria are Akaike's Information Criterion (AIC) [8] (with its small sample corrected version (AICC) [11]) and the two-stage code Minimum Description Length criterion (MDL) [10]. The *penalty* term for AIC is $2k$, for AICC is $2k + (2k(k+1))/(n-k-1)$, and for MDL is $k\log(n)$, where n is the number of data cases in the training data. Note that the best fitting model is that whose criterion value is the lowest. In this paper we used AICC and a slightly modified MDL (see Section 3.).

The typical state space of subset selection is such that each state represents a subset. For m basis functions there are m bits in each state, each bit indicates whether a function is present ("1") or absent ("0"), and all the states are ordered in such a way that in each successive layer of state space the states have one included basis function more than in previous layer (see Figure 1). A search algorithm implements a chosen search strategy and traverses the space trying to find the state in which the according basis function subset forms the best regression model. The goal is to find at least a suboptimal model, with as less state transitions as possible, as each transition requires evaluation of a number of models, what includes OLS parameter estimation for each model.

The downside of the subset selection approach is that it assumes that the chosen *fixed* full set of *predefined* basis functions contains a subset which is sufficient to describe the target relation sufficiently well. However we argue that in most cases the necessary set of basis functions (or previously chosen highest allowed order, p) is not known and needs to be guessed or maybe chosen by experience as each regression problem may need a different value. In many cases that means a non-trivial (and long) trial and error process that, as it will be demonstrated in Section 4, in many problems of moderate dimensionality may even become computationally too demanding.

A more convenient way would be to let the modelling method itself construct the basis functions necessary for creating the regression model with adequate predictive performance. In the next section we will consider exactly such an approach.

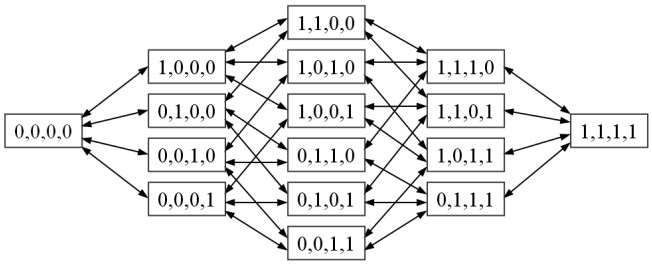

Figure 1. A small example of a typical state space in subset selection

3. The approach with basis function construction

To obtain a regression modelling method, which can by itself construct the necessary basis functions, we replace the standard refinement operators of subset selection with other operators which not only allow adding or deleting the basis functions but also allow changing the basis functions themselves. We call this a basis function construction approach.

In Figure 2 there is shown relation between subset selection and function construction. Subset selection operates with a string of bits where each bit indicates whether a predefined function is present ("1") or absent ("0"). Basis function construction approach on the other hand operates directly with the orders of each variable in each function as well as creates new functions as necessary. Thus function construction does not operate with the one-dimensional string of bits. Instead it operates with a matrix of non-negative integers where a cell in i-th row and j-th column contains a value of rij in equation (2) which is the order of the j-th variable in the i-th basis function. As it may be noticed, the space of candidate regression models is now infinite, and we can generate polynomials of arbitrary complexity.

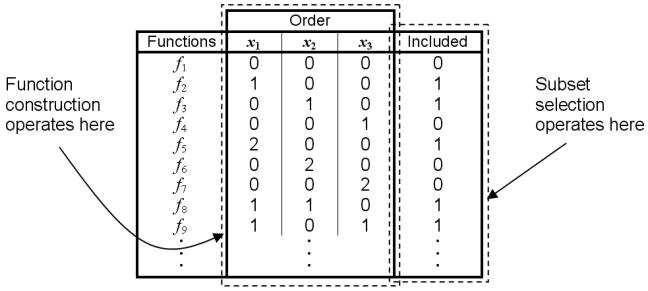

Figure 2. Relation between subset selection and function construction

In the previous section we listed five basic issues on which one must pay attention when characterizing a heuristic search problem. Now let us look at the issues in the context of the basis function construction approach.

As the state space has become infinite a natural *initial state* of the search is now the state where the subset of basis functions is empty. Or one may also choose a state which has some very few simple functions (e.g. just one function which corresponds to the intercept term or all the functions which correspond to the first order terms). A small set of simple functions might be also generated randomly.

Some of the possible *refinement operators* and applicable *search strategies* are discussed in Section 3.1. and 3.2. respectively.

The typical *termination condition*, which is met when the search locates a state in which none of the refinement operators can lead to a better state, is of course a natural choice also in the function construction approach.

Concerning the *evaluation measure*, because of the fact that during the search the orders of the variables in the basis functions may be changed without changing the actual number of basis functions, many complexity penalization criteria, such as AIC, AICC, or two-stage version of MDL may tolerate unnecessary increase of orders in basis functions, as they cannot detect a change in model's structure if the number of parameters stayed the same. To deal with this problem in [4] the authors of CIPER introduced a simple modification of the two-stage MDL which, instead of penalizing the model by the number of its parameters, penalizes it by its sum of all orders in all basis functions:

$$\mathrm{MDL} = n \log(SSE/n) + l \log(n)$$

where $l = \sum_{i=1}^{k} \sum_{j=1}^{d} r_{ij}$ is the "length" of polynomial.

However, we used AICC criterion and, to preserve its original form, we did not modify it. Instead we used another approach – additional penalization using Akaike's weights [8,9] only in the very moment of comparison. When using a refinement operator which raises the order of a variable without changing the number of model's basis functions, we ask the Akaike's weight w_{new} of the newly constructed model to be at least 10% higher than that of the "old" model. In other words, the weight (which may also be directly interpreted as the conditional probability of the new model being better than the competitor) of the new model should be >60% instead of >50%:

$$w_{\mathrm{new}} = \frac{\exp(-0.5\Delta\mathrm{AICC})}{1+\exp(-0.5\Delta\mathrm{AICC})} > 0.6$$

where $\Delta\,\mathrm{AICC} = \mathrm{AICC}_{\mathrm{new}} - \mathrm{AICC}_{\mathrm{old}}$.

3.1. Refinement operators

Using efficient refinement operators is vital for the search process for the best regression model to be successful. In this section we discuss some of the possible refinement operators.

Generally there are two different basic ways to refine an existing model: adding/deleting the basis functions and operating with the orders of variables in an existing basis function (e.g. increasing or decreasing them).

Here are the five considered refinement operators:

- *Operator1*: Increasing of one of the orders in one of the existing basis functions by 1.
- *Operator2*: Addition of a new basis function with one of the orders set to 1.
- *Operator3*: Addition of an exact copy of already existing basis function with one of the orders increased by 1.
- *Operator4*: Decreasing of one of the orders in one of the existing basis functions by 1.
- *Operator5*: Deletion of one of the existing basis functions.

We categorize the listed refinement operators as complication operators (the first three) and *simplification* operators (the last two). If the search is started from an empty or some small set of functions, the complication operators do the main job – they "grow" the regression model. The simplification operators on the other hand work as purifiers – they decrease the unnecessarily high orders and delete the unnecessary basis functions.

The first two complication operators were already introduced in [4] where they were used in CIPER. However using only these two operators three issues can arise, all of which can lead to getting stuck in local minima too early, which in turn can result in poor predictive performance of the resulting regression model: 1) In the first iterations the branching factor of the state space may be too low – the search algorithm may have too few choices to be able to continue the search. 2) Operator1 increases an order in an existing function but does not take into consideration the possibility that both versions of the basis function may be needed. The only way to reconstruct the lost function is to start from scratch – by using the Operator2. 3) Without the use of simplification operators a regression model may contain unnecessarily high orders and include unnecessary basis functions which may prevent truly necessary modifications (the so-called nesting effect [3]).

The first two issues are addressed by the Operator3. The last one issue is addressed by the two simplification operators.

In Figure 3 there is shown a small example of a state space if we use all except the third refinement operator. Each state represents a set of basis functions which are included in the regression model – a matrix which corresponds to the columns named "Order" in Figure 2. Connections created by the Operator3 are not shown as they would be cross-layer connections between the states.

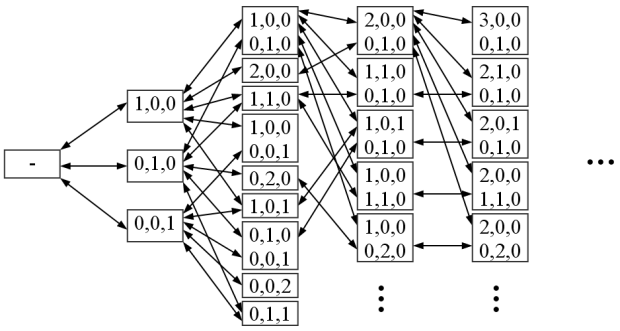

Figure 3. A small example of a state space in function construction.
For simplicity the function with all 0's and connections of Operator3 are omitted

3.2. Search strategies

Most search strategies that are applicable to subset selection also can be used in function construction. One exception is the strategies that start their search from the full subset (e.g. SBS). As in function construction approach there exists no full subset, the only way to make these strategies work would be to define the full subset to be some sufficiently large set of functions, definition of which mostly would have no special reason, and moreover, it generally would bring us back to the approach of subset selection. Another exception is the strategies that require the state-representing data structures to be of constant length and are not generally biased towards simpler models (e.g. the strings of bits in most Genetic Algorithms). However with appropriate modifications they might become applicable.

In this paper we will consider only the directly applicable search strategies the simplest of which is the SFS. However SFS moves only forward, in direction of more complex models, so it would use only the complication operators and, because of the resulting low branching factor and the nesting effect, frequently would stuck in local minima too early. Two of the possible remedies for this problem are Beam Search strategy [6] and the SFFS strategy [3]. The former is used in CIPER (see next section for a brief summary on this method) while the latter is used in our introduced method which we discuss in the Section 3.4.

3.3. Constrained Induction of Polynomial Equations for Regression

CIPER [4] was developed in the context of inductive databases and constraint-based data mining. As already said, CIPER uses only the first two complication operators and, as well as SFS, only searches forward, however, as a compensation for the low branching factor, Beam Search strategy is used.

Here is an overview of CIPER's settings: *Initial state:* the state with one function that corresponds to the intercept term (this function stays in the model at all times and is not allowed to be modified or deleted). *Refinement operators*: the first two complication operators listed in Section 3.1. *Search strategy*: Beam Search (with default beam width equal to 16). *Termination condition*: when no further improvements are possible. *Evaluation measure*: the modified two-stage MDL (see Section 3).

We already discussed CIPER's drawbacks related to the used refinement operators in Section 3.1. In the next section we will introduce a new basis function construction method which tries to avoid these drawbacks.

3.4. Sequential Floating Forward Polynomial Construction

In this section we introduce a new method for polynomial regression modelling which is another instance of the function construction approach – SFFPC. It uses all five refinement operators listed in Section 3.1. However, because of the resulting higher branching factor, in problems of moderate and high dimensionality using of the Beam Search strategy would have very high relative time-complexity. Moreover, because of the larger forward steps of the Operator3 and backward steps of both simplification operators, the Beam Search strategy would have to either periodically re-evaluate already evaluated states or maintain a sufficiently large *tabu* list of the evaluated states to avoid the re-evaluation. Therefore, to maintain simplicity and efficiency, as the search strategy we use the floating search method SFFS – hence the similar name of the SFFPC.

Here is an overview of SFFPC's settings: *Initial state*: the state with one function that corresponds to the intercept term (this function stays in the model at all times and is not allowed to be modified or deleted). *Refinement operators*: all five listed in Section 3.1. *Search strategy*: SFFS. *Termination condition*: when no further improvements are possible. *Evaluation measure*: all refinement operators are used with AICC except the first where Akaike's weights with $w_{new}>0.6$ is used (see Section 3).

In Figure 4 there is shown the pseudocode of SFFPC's search procedure. Before the search starts, the *BestModel* is initialized with the simplest model – the model with one function which corresponds to the intercept term. The model is then evaluated (parameter estimation using OLS and AICC value calculation). Each iteration consists of two phases – forward and backward. In the first phase the complication operators are used. All possible models which result from the usage of the three operators on *BestModel* are generated and evaluated. Next, the one best new model, which has the most improved performance over the *BestModel*, is found (with additional comparison of Akaike's weights, if the model was generated using the Operator1). If a better model was found, it becomes the new *BestModel*, otherwise search procedure ends (*BestModel* now holds the best found model). In the second phase the simplification operators are used. The second phase works basically the same as the first except that it ends only when it is impossible to generate a model which is better than the *BestModel*. At the end of the second phase the search always proceeds to the next iteration.

```
BestModel := {the one function which correspond to the intercept term (with r_1j := 0, j = 1..d)}
BestModel.AICC := Evaluate(BestModel)
repeat
    //forward phase
    NEWMODELS := {all models which result from the complication operators on BestModel }
    CurrBestModel := BestModel
    foreach TestModel ∈ NEWMODELS do
        TestModel.AICC := Evaluate(TestModel)
        if TestModel.AICC < CurrBestModel.AICC and
          ( TestModel was not generated using the Operator1 or
            CalculateAkaike'sWeight(TestModel.AICC, BestModel.AICC) > 0.6 ) then
                CurrBestModel := TestModel
    endfor
    BestModel := CurrBestModel
    if BestModel did not change then
        exit
    //backward phase
    repeat
        NEWMODELS := {all models which result from the simplification operators on BestModel }
        foreach TestModel ∈ NEWMODELS do
                TestModel.AICC := Evaluate(TestModel)
                if TestModel.AICC < BestModel.AICC then
                        BestModel := TestModel
        endfor
    until BestModel did not change
until forever
```

Figure 4. Pseudocode of SFFPC's search procedure

4. Empirical experiments

The main goal of the performed experiments is to compare the two instances of the basis function construction approach, SFFPC and CIPER, with the two popular instances of subset selection approach, SFS and SFFS, in terms of both, predictive performance of the induced regression models as well as necessary computational resources. We also compared SFFPC and CIPER and ascertain the advantages and disadvantages of SFFPC's ability to use the additional refinement operators versus CIPER's beam search strategy. The performance of the methods is evaluated on three artificial problem data sets and three data sets from the WEKA project website (http://www.cs.waikato.ac.nz/ml/weka/).

All the experiments were performed on Pentium 4 2.4GHz computer with Hyper Threading turned on. Note that the time consumption presented in the tables is only a rough measurement as the methods are implemented in different software and with different levels of optimization of calculations. In the experiments we used our in-house software with implementations of SFS, SFFS, SFFPC, as well as a version of CIPER where instead of the MDL we used AICC criterion (exactly like the SFFPC uses it). This allowed us to compare the use of refinement operators and search strategies without any hindrance because of the different criteria. As an implementation of the original CIPER we used the original software which is publicly available at http://ai.ijs.si/pljubic/ciper/ciper.html, kindly provided by the authors of the method. In both versions of CIPER we used the default beam width, 16.

In experiments with the artificial problem data sets we randomly generated training data set with 150 cases with all the input variables uniformly distributed over the interval [0,1], and tested the induced regression models on unseen test data set of 10000 randomly generated cases. We repeated the process for 10 different training data sets and averaged the results. In all the other experiments we estimated predictive error of the induced models on unseen data

samples using 10-fold Cross Validation (CV) and averaged the results. The predictive performance of a model is measured in terms of relative root mean squared error, RRMSE, defined as model's root mean squared error divided by standard deviation of the dependent variable y, both calculated using the unseen examples of the test set.

The three artificial problem data sets were generated with the following functions:

$$F_1 = \prod_{i=1}^{7} x_i + \prod_{i=4}^{10} x_i + \varepsilon$$

$$F_2 = \prod_{i=1}^{7} x_i + \prod_{i=4}^{10} x_i + x_{10} \prod_{i=4}^{10} x_i + \varepsilon$$

$$F_3 = \sin(\pi x_1 / 2)\sin(x_2)\cos(x_3)\cos(x_4) + \sin(\pi x_5 / 2)\sin(x_6)\cos(x_7)\cos(x_8) + \varepsilon$$

where ε is a normal noise with mean equal to zero and standard deviation $\sigma = 0.001$ for the first two functions and $\sigma = 0.01$ for F_3. If the function which generated the data is unknown, regression modelling with subset selection approach with the F_1 and F_2 data sets is impractical. With F_1 high predictive performance can be reached only by defining the maximal order of polynomials to be at least $p = 7$ (this of course would need to be guessed or obtained by experimenting) creating $m = 43758$ basis functions (according to equation (2)) which leads us to a state space with number of states equal to $2^{43758} \approx 3 \cdot 10^{13172}$. With F_2 it would be $2^{109395} \approx 1.5 \cdot 10^{32931}$ as the maximal order must be at least $p = 8$.

Table 1 presents the results of experiments with the four regression modelling methods used on the artificial problem data sets. Here the methods of basis function construction are clearly superior – they were able to relatively quickly find models with high predictive performance. With F2 both versions of CIPER did not perform so well – by using only the first two refinement operators they could not construct the third basis function which is an exact copy of the second basis function having the order of x10 increased by 1. As the third basis function would need to be constructed from scratch, the problem is that, even if the method would not stuck in local minima, at some iteration there may exist two exact copies of one basis function, which is of course not allowed. The SFFPC on the other hand manages very well – after creation of the second basis function it just uses the third refinement operator to create a copy of it with the order of the 10th variable increased by 1.

It should be noted that with the first two artificial problems the models found by SFFPC contain slightly more basis functions than necessary. This is explainable by the fact that the AICC criterion does not assume the true model to be among the candidates and may give preference to more complex models [9,11].

The three data sets from the WEKA project website are the following: "autoMpg" (392 data cases, 7 input variables, "bodyfat" (252 data cases, 14 input variables), and "housing" (506 data cases, 13 input variables). Before dividing the data sets into CV folds, the order of the cases was randomized.

The results, presented in the Table 2, confirm that the function construction methods can have the same or better predictive performance as the subset selection methods, without the necessity to choose the maximal order. They also show that, in common with the subset selection methods, when the goal quantity of the basis functions in the model is relatively low (as with "autoMpg" and "bodyfat" data sets), the needed computational resources are much lower.

Table 1. Results for the artificial problems. Average RRMSE error (%), AICC, elapsed time (s), and *k*

Method	F_1 RRMSE	F_1 AICC	F_1 Time / *k*	F_2 RRMSE	F_2 AICC	F_2 Time / *k*	F_3 RRMSE	F_3 AICC	F_3 Time / *k*
SFS, $p = 2$	71.82	-1238.7	1.8 / 28	74.27	-1100.8	0.8 / 18	14.84	-1037.4	0.8 / 29
SFS, $p = 3$	64.60	-1410.8	39 / 47	65.66	-1302.6	39 / 48	8.62	-1263.2	13 / 42
SFS, $p = 4$	54.55	-1647.3	320 / 62	58.61	-1560.2	419 / 69	8.31	-1337.2	70 / 47
SFS, $p = 5$	40.42	-1946.3	2251 / 78	43.93	-1811.6	2074 / 77	**7.07**	-1382.2	212 / 47
SFFS, $p = 2$	73.05	-1229.5	1.4 / 19	74.50	-1101.6	1.1 / 15	14.90	-1039.3	1.5 / 29
SFFS, $p = 3$	66.03	-1431.6	35 / 36	68.42	-1302.7	29 / 34	8.47	-1289.9	19 / 39
SFFS, $p = 4$	57.64	-1698.6	416 / 58	58.56	-1577.8	391 / 55	**6.93**	-1351.2	51 / 35
SFFS, $p = 5$	41.41	-2041.7	2842 / 75	42.43	-1877.4	1831 / 64	8.05	-1407.9	245 / 42
CIPER+MDL	**0.75**	-	1.5 / 4	11.63	-	2.9 / 5	11.44	-	12 /.15
CIPER+AICC	3.94	-2146.5	80 / 17	9.69	-2106.1	150 / 20	11.57	-1327.7	186 / 27
SFFPC	2.33	-2095.4	1.3 / 9	**1.95**	-2086.7	1.4 / 9	**6.59**	-1418.8	34 / 32

Table 2. Results for the WEKA data sets. Average RRMSE error (%), AICC, elapsed time (s), and *k*

Method	autoMpg RRMSE	autoMpg AICC	autoMpg Time / *k*	bodyfat RRMSE	bodyfat AICC	bodyfat Time / *k*	housing RRMSE	housing AICC	housing Time / *k*
SFS, $p = 2$	36.77	719.1	0.2 / 11	15.30	-44.7	1.1 / 13	41.72	1018.1	29 / 48
SFS, $p = 3$	37.28	691.1	3.1 / 18	**14.14**	-130.5	11 / 13	**38.52**	809.6	997 / 83
SFS, $p = 4$	37.82	664.7	28 / 26	14.87	-109.1	157 / 16	53.63	711.3	7600 / 97
SFS, $p = 5$	**36.21**	674.7	59 / 20	37.72	-147.4	1902 / 17	-	-	-
SFFS, $p = 2$	**36.56**	711.2	0.3 / 9	16.10	-33.4	1.4 / 11	42.16	1011.7	31 / 32
SFFS, $p = 3$	36.84	687.2	2.7 / 14	**14.14**	-91.7	11 / 11	41.88	795.8	615 / 54
SFFS, $p = 4$	37.93	658.4	17 / 16	15.85	-90.4	147 / 12	**39.02**	727.4	3636 / 57
SFFS, $p = 5$	36.57	670.9	41 / 16	14.81	-143.9	1727 / 12	-	-	-
CIPER+MDL	38.31	-	1.2 / 7	14.71	-	1.2 / 6	45.21	-	23 / 14
CIPER+AICC	38.03	656.3	37 / 14	14.46	-171.7	58 / 12	46.34	783.0	2850 / 33
SFFPC	**37.06**	678.0	2.9 / 13	**12.81**	-186.7	2.4 / 11	**36.91**	683.6	1059 / 53

CIPER, in contrast to SFFPC, can not delete basis functions which have become obsolete, which seems to be the main reason for SFFPC superiority over CIPER+AICC. Comparing the CIPER+AICC and CIPER+MDL, it can be observed that MDL forces the CIPER to construct much simpler models, on the one hand making the method much faster and more robust to overfitting, but on the other hand making the method to underfit.

An additional observation is that, using the AICC, SFFS in comparison with SFS much often overfits the data, however we think that in function construction using the SFFS search strategy is preferable to SFS as the overall branching factor is much lower than that in subset selection.

5. Conclusion

In this paper we considered a basis function construction approach in polynomial regression modelling. The approach is different from the standard subset selection approach which assumes that the predefined fixed full set of basis functions contains a subset which is sufficient to describe the target relation sufficiently well. The function construction approach on the other hand offers an automatic construction of basis functions using heuristic search. Consequently the user does not need to predefine the set of basis functions for model creation. We also introduced an instance of the approach – a new regression modelling method. The performed empirical experiments showed that the method has a potential to efficiently construct regression models of relatively high predictive performance.

Directions of future research include considering also other types of refinement operators or search strategies which may lower the probability of getting stuck in local minima, as well

as considering other model evaluation methods which are more robust to overfitting than the used criteria.

Acknowledgement

This work has been partly supported by the European Social Fund within the National Program "Support for the carrying out doctoral study program's and post-doctoral researchers" project "Support for the development of doctoral studies at Riga Technical University".

References

1. Rawlings J.O. "*Applied Regression Analysis*" // Wadsworth & Brooks/Cole, Pacific Grove, CA, 1998
2. Aha D.W., Bankert R.L. "*A comparative evaluation of sequential feature selection algorithms*" // In: Learning from Data, Fisher D., Lenz H.J. (eds.), Springer, New York, USA, 1996, p. 199-206
3. Pudil P., Novovicova J., Kittler J. "*Floating search methods in feature selection*" // In: Pattern Recognition Letters, 15, 1994, p. 1119-1125
4. Todorovski L., Ljubic P., Dzeroski S. "*Inducing polynomial equations for regression*" // In: Lecture notes in computer science, Lecture notes in artificial intelligence, 3201, Springer, Berlin, 2004, p. 441-452
5. Ginsberg M.L. "*Essentials of Artificial Intelligence*" // Morgan Kaufmann, 1993
6. Russell S.J., Norvig P. "*Artificial Intelligence: A Modern Approach*" // Prentice Hall, Englewood Cliffs, New Jersey 07632, 1995
7. Blum A.L., Langley P. "*Selection of relevant features and examples in machine learning*" // In: Artificial Intelligence, 97, 1997, p. 245-271
8. Akaike H. "*A new look at the statistical model identification*" // In: IEEE Transactions on Automatic Control, 19, 1974, p. 716-723
9. Burnham K.P., Anderson D.R. "*Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*" // Springer, 2002
10. Rissanen J. "*Modeling By Shortest Data Description*" // In: Automatica, 14, 1978, p. 465-471
11. Hurvich C.M., Tsai C-L. "*Regression and time series model selection in small samples*" // In: Biometrika 76, 1989, p. 297-307

Jēkabsons G., Lavendels J. Polinomu regresijas modelēšanas pieeja pielietojot bāzes funkciju konstruēšanu

Polinomu regresijas modelēšanā parasti tiek izmantota apakškopas izvēles pieeja. Šajā pieejā tiek pieņemts, ka izvēlētā fiksētā iepriekš definēto bāzes funkciju kopa satur apakškopu, ar kuru pietiek, lai pietiekoši labi aprakstītu mērķa sakarību. Taču mēs uzskatam, ka nepieciešamo bāzes funkciju kopa visbiežāk nav zināma un to ir jācenšas uzminēt vai izvēlēties, izmantojot pieredzi. Daudzos gadījumos tas nozīmē sarežģītu (un ilgu) mēģinājumu un kļūdu procesu, kas var ģenerēt funkciju kopas, ar kurām, pat izmantojot heiristiskas pārmeklēšanas metodes, strādāt ir nepraktiski pat problēmās ar mērenu sarežģītību. Ērtāks veids būtu ļaut modelēšanas metodei pašai konstruēt bāzes funkcijas, kas tai ir nepieciešamas regresijas modeļu izveidei ar pietiekamu paredzēšanas spēju. Rakstā tiek apskatīta polinomu regresijas modelēšanas pieeja ar automātisku bāzes funkciju konstruēšanu, izmantojot heiristisku pārmeklēšanu. Lietotājam modeļu izveidei nav iepriekš jādefinē bāzes funkciju kopa. Pētījumu rezultātā tika izstrādāta apskatītās pieejas instance – jauna regresijas modelēšanas metode, kas spēj ģenerēt jebkuras sarežģītības polinomus. Lai novērtētu piedāvāto metodi, tā tiek salīdzināta ar divām labi zināmām un bieži lietotām apakškopas izvēles metodēm, izmantojot kā mākslīgus tā arī reālās pasaules datus.

Jekabsons G., Lavendels J. An approach for polynomial regression modelling using construction of basis functions

In polynomial regression modelling typically the subset selection approach is used. This approach assumes that the chosen fixed full set of predefined basis functions contains a subset which is sufficient to describe the target relation sufficiently well. However we argue that in most cases the necessary set of basis functions is not known and needs to be guessed or chosen by experience. In many cases that means a non-trivial (and long) trial and error process that may generate sets of functions, working with which, in many problems of moderate dimensionality, may become computationally too demanding even for the heuristic search methods. A more convenient way would be to let the modelling method itself construct the basis functions necessary for creating the regression model with adequate predictive performance. In this paper we consider a polynomial regression modelling approach with automatic construction of basis functions using heuristic search. The user does not need to predefine the set of basis functions for model creation. As a result we introduce an instance of the approach – a new regression modelling method that can generate polynomials of arbitrary complexity. To evaluate the proposed method we compare it to two well known and widely used subset selection methods both on artificial and real world data.

Екабсон Г., Лавендел Ю. Подход моделирования полиномиальной регрессии с использованием конструирования базисных функций

В процессе моделирования полиномиальной регрессии обычно используется подход выбора подмножества. При этом подходе принимается, что фиксированное множество базисных функций содержит и подмножество базисных функций, которое с необходимой степенью точности описывает исследуемую взаимосвязь. По нашему убеждению множество базисных функций неизвестно и его приходится выбирать на основе имеющегося опыта или просто угадывать. Во многих случаях это приводит к длительному и сложному процессу опытов и ошибок генерации множеств функций. Работа с такими большими множествами непрактична, даже используя эвристические алгоритмы для проблем средней сложности. Более удобно придать методу моделирования свойства генерации базисных функций, которые необходимы для построения регрессионной модели с необходимой степенью предвидения. В статье рассмотрен подход моделирования полиномиальной регрессии с использованием генерации базисных функций и эвристического перебора. Пользователю, при этом, нет необходимости заранее определить множество базисных функций. В результате исследований предложен метод, который позволяет генерировать полиномы любой сложности в задачах моделирования регрессионных зависимостей. Для оценивания качества предложенного метода, он сравнивается с двумя другими распространенными методами, используя как искусственные данные, так и данные полученные экспериментально.