Skoltech

Skolkovo Institute of Science and Technology

MASTER'S THESIS

Structured learning for model generation

Master's Educational Program: «Data Science»

Student: Artem Bochkarev

Reseach Advisor: Maxim Fedorov, Professor at Skolkovo Institute of Science and Technology

Co-Advisor: Vadim Strijov, Professor at Moscow Institute of Physics and Technology

Стрижов

Moscow, 2018

Skoltech

Skolkovo Institute of Science and Technology

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Структурное обучение для генерации моделей

Магистерская образовательная программа: «Наука о данных»

Студент: Бочкарев Артем Максимович

Научный руководитель: Федоров М.В., Профессор Сколковского института науки и технологий

Со-руководитель: Стрижов В.В., Профессор Московского физико-технического института

Москва, 2018

Structured learning for model generation

Artem Bochkarev

Submitted to the Skolkovo Institute of Science and Technology on 01 June 2018

ABSTRACT:

The study is dedicated to generating models for time series in the form of mathematical expressions. The conventional approach to this problem is to use symbolic regression algorithm. This algorithm has many advantages such as interpretability and high quality of approximation. The downside of this approach is that it is very computationally expensive and there are no theoretical guarantees of convergence. Our research addresses the first of this problems.

We propose a to use a method of meta learning to find new models. This approach uses the knowledge of previously built models for other similar problems. The meta learning dataset then is a set of pairs of problems and corresponding optimal models. Models are represented as trees, therefore the meta learning problem is posed as prediction of tree structure of the model.

We propose algorithm for prediction of tree structure, which consists of two stages. During the first stage, classification function is used to predict the probabilities of the edges in the model tree. The second stage is recovery function which explicitly constructs the tree from the probabilities of its edges. For the second stage we suggested greedy algorithm and two variations of dynamic programming approach. Later, we introduced parameters to the models, which allows them to fit real data. The optimal parameters are found using gradient descent, as the mathematical expressions are differentiable functions.

The computational experiment was conducted on synthetic and real data. The goal of the experiment on the synthetic data was to prove the correctness of the proposed approach and test various choices for classification and recovery functions. We also conducted the experiment on the data from parametric models. The real data were segments of time series from accelerometer measurements and stock prices. The goal was to compare meta learning approach with symbolic regression in terms of speed and quality of built models. The proposed algorithm achieved comparable quality with more than 10x speedup, compared to symbolic regression.

Research Advisor:

Name: Maxim Fedorov

Degree: Doctor of Chemical Science

Title: Professor of Skolkovo Institute of Science and Technology

Co-advisor:

Name: Vadim Strijov

Degree: Doctor of Physical and Mathematical Sciences

Title: Professor of Moscow Institute of Physics and Technology

Contents

1. INTRODUCTION

Genetic programming [1] is a powerful approach for building models. Symbolic regression [2] uses genetic algorithms [3] in order to find the mathematical expression for the optimal approximation model. The resulting model is interpretable and understandable by experts in the application field [4]. The other advantage of genetic programming is that it provides very high quality of approximation. Genetic algorithms often outperform other optimisation methods in complex and multimodal problems [5].

These advantages make genetic programming suitable for many applications. Symbolic regression is used for ranking documents upon user request in information retrieval [6], for classification of time series of physical measurements [7] and for diagnosing pathologies from medical data [8]. It was also shown that genetic programming can improve optimisation of deep neural networks [9]. In [10] authors used symbolic regression models to simplify the structure of neural network. This study shows that one can significantly reduce the number of hidden units in neural network, keeping the quality the same, if the symbolic regression models are used as features for a neural network.

The main drawback of genetic approach is that it is guided random search, therefore in some circumstances it might be very slow due to improper selection of heuristics. The problem is addressed in [11] using acceleration on GPU. The goal of this study is to increase the speed of generating models in the form of mathematical expressions.

The proposed method is based on meta learning approach. The thorough relevant survey of the field is presented in [12, 13]. The aim is to find an optimal model for a new approximation problem (base problem), given the optimal models for previous similar base problems. Each mathematical expression is seen as a binary tree of superpositions of primitive functions. The model in this case is a tree and the problem is to learn to predict a complex structure. The overview of structured learning approaches can be found in [14]. The problem can also be posed as decoding tree structure from some vector representation of base problem.

In [15] authors propose tree autoencoder architecture for generating structures of the molecules. In the paper [16] the model for generating programs from input-output pairs is proposed. The authors of paper [17] presented a neural hybrid tree framework for semantic parsing. The paper [18] introduced doubly-recurrent neural network structure for decoding tree structure, showing results on synthetic data and machine translation problem.

This study is devoted to accelerating of generation symbolic regression models, using structured learning. The problem posed as a meta learning task, in order to use knowledge of previously built models. Symbolic regression models are represented as trees. The main contribution of this research is a proposal of novel method of predicting tree structure of the models. The algorithm is divided in two stages. The first stage is a classifier, which predicts the probabilities of the edges in a model tree. The second stage is a recovery function, which constructs a valid model tree from the probabilities of its edges. Three different variations of recovery function are proposed.

All proposed approaches are extensively tested on synthetic and real data. There are two kinds of synthetic data, generated in this study – parametric and nonparametric models. In computational experiment we compare all variations of proposed method on synthetic data and present the proof of the method's good performance. There are three real datasets used in computational experiments – accelerometer and financial time series. On all three tested datasets proposed method has significantly shorter time of building the model, compared to symbolic regression. At the same time, meta learning approach retains the high quality of output models.

2. PROBLEM STATEMENT

Consider the set of supervised problems with optimal models for them. Each single supervised problem is named “base problem”. The regression problems are considered to be base problems in this study. The model for a base problem is a mathematical expression. The goal of this study is to build a method to find models for base problems automatically.

2.1. Base problem

Denote $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^{n \times l}$ be the feature matrix and $\mathbf{y} = \{y_i\}_{i=1}^n$ be the vector of target variables for the base problem. The feature matrix $\mathbf{X}$ and target vector $\mathbf{y}$ are combined into the dataset $D = (\mathbf{X}, \mathbf{y})$ which is a full description of a base problem. From now on we denote both base problem and its dataset as D.

In this study the following conditions are posed to the base problem:

- $\mathbf{x}_i$ is not random,
- $\{\mathbf{x}_i\}_{i=1}^n$ is an ordered set,
- $\mathbf{y}$ is random,
- $\exists f : y_i = f(\mathbf{x}_i) + \varepsilon_i$,
  - ε_i are independent,
  - ε_i are homoscedastic,
  - $\varepsilon_i \sim \mathcal{N}(0, \sigma)$.

These conditions are satisfied for various kinds of real datasets, for example time series.

2.2. Model for a base problem

The space of mathematical expressions $\mathfrak{F}$ is searched for a model f for base problems. Any mathematical expression is generated by the grammar G of primitive functions:

$$g \to B(g, g)|U(g)|S, \tag{1}$$

where B is a set of binary primitive functions $(+, \times, \dots)$, U is a set of unary primitive functions $(\log(\cdot), \sin(\cdot), \sqrt{\cdot}, \dots)$ and S is a set of variables. Therefore, each mathematical expression f is a superposition of primitive functions from grammar G:

$$f = g_1 \circ g_2 \circ \dots \circ g_k \tag{2}$$

Tree representation. Each mathematical expression f is represented as a binary tree Γ_f, which satisfies the following conditions:

- the root of the tree is a special symbol "*", it has one child vertex,
- leaves of Γ_f contain variables $x \in S$
- each non-leaf vertex v contains primitive function $g \in B \cup U$
- number of children of vertex v equals arity of corresponding primitive function g
- domain of a child vertex v_j contains codomain of a parent vertex v_i: $\text{dom}(v_j) \supseteq \text{cod}(v_i)$
- children vertices are ordered.

The example of a tree for a mathematical expression can be seen on figure 1.

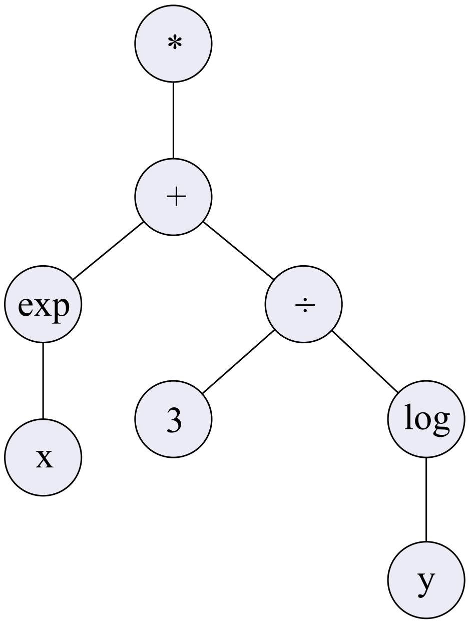


Figure 1: Tree Γ_f for expression $f = e^x + \frac{3}{\log(y)}$

2.3. Meta learning problem

This study explores the approach of meta learning for finding models of base problems. Denote by $\mathfrak{D} = \{D_i = (\mathbf{X}_i, \mathbf{y}_i), f_i\}_{i=1}^m$ the data for the meta learning problem (meta learning dataset).

The following conditions are satisfied for meta learning dataset $\mathfrak{D}$:

- $\mathrm{dom}(\mathbf{x}_i) = \mathrm{dom}(\mathbf{x}_j)\ \forall i, j$ (all $\mathbf{X}$ share the same domain),
- f_i is an optimal model for the base problem D_i in a model space $\mathfrak{F}$:

$$f_i = \arg\min_{f \in \mathfrak{F}} \mathrm{MSE}(\mathbf{y}_i, f_i(\mathbf{X}_i)), \tag{3}$$

where

$$\mathrm{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{l} \sum_{j=1}^{l} (y_j - \hat{y}_j)^2 \tag{4}$$

is a mean squared error of the model on the base problem.

Given the meta learning dataset $\mathfrak{D}$, the goal is to find an optimal meta learning model $\mathfrak{g} : D \to f$ which minimises the error among all base problems:

$$\mathfrak{g} = \arg\min_{\mathfrak{g}} \mathcal{L}(\mathfrak{g}, \mathfrak{D}), \tag{5}$$

$$\mathcal{L}(\mathfrak{g}, \mathfrak{D}) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{MSE}(\mathbf{y}_i, \mathfrak{g}(D_i)(\mathbf{X}_i)). \tag{6}$$

3. META LEARNING APPROACH

In this section we show how to find meta model $\mathfrak{g}$ which generalises well to new base problems. We present framework for generating non-parametric models f and then expand it to allow parameters.

3.1. Choice of representation

The meta learning function is a mapping between base problems D and the space of mathematical expressions $\mathfrak{F}$. In order to define such mapping, define a suitable representation of base problem and its model.

Base problem representation. Denote by $\mathbf{d} = [\text{vec}(\mathbf{X}), \mathbf{y}]^T$ the vector representation of a base problem D. This vector is a concatenation of vectorised feature matrix $\mathbf{X}$ and target vector $\mathbf{y}$

Assumption 1. *All information needed for a generation of a model f for a base problem D is encoded in its representation vector* $\mathbf{d}$.

Model representation. There are three ways to represent a model f:

- the mathematical expression of f,
- the tree Γ_f, corresponding to the model f,
- the adjacency matrix $\mathbf{Z}_f$, corresponding to the tree Γ_f.

The third way allows a vectorised representation of a model f, it is selected for the proposed method of model generation. Therefore, the meta model $\mathfrak{g} : \mathbb{R}^n \to \mathbb{Z}$ is a mapping between vector representations of a model and the space $\mathbb{Z}$ of valid adjacency matrices of mathematical expressions.

3.2. Meta model decomposition

The matrix Z_f, predicted with meta model $\mathfrak{g}$, has to satisfy all conditions on a tree Γ_f, listed in section 2.2. Therefore, the construction of direct mapping $\mathfrak{g}$ is infeasible. The proposed method is to decompose the meta model into two stages:

$$\mathbf{Z}_f = \mathfrak{g}(\mathbf{d}) = g_{\text{rec}}(g_{\text{clf}}(\mathbf{d})), \tag{7}$$

where g_{rec} is a recovery function and g_{clf} is a classification function.

Classification function. $g_{\text{clf}} : \mathbb{R}^n \to \mathbb{P}$ is a mapping between vector representations of a model and the space $\mathbb{P}$ of matrices of edge probabilities. Therefore,

$$g_{\text{clf}}(\mathbf{d}) = \mathbf{P}_f, \tag{8}$$

where $\mathbf{P}_f$ is a matrix of probabilities of edges in the tree Γ_f. g_{clf} is a multi label classification algorithm, which predicts the probability $p_{ij} \in [0, 1]$ that there is an edge between vertices v_i and v_j for any pair of vertices in the tree Γ_f.

Recovery function. $g_{\text{rec}} : \mathbb{P} \to \mathbb{Z}$ is a mapping between the space $\mathbb{P}$ of matrices of edge probabilities and the space $\mathbb{Z}$ of valid matrices for mathematical expressions. g_{rec} is a nonparametric algorithm which selects the edges for Γ_f, based on their probabilities from $\mathbf{P}_f$. The resulting tree satisfies conditions from section 2.2.

In this study we propose two different methods for tree recovery, based on greedy strategy and dynamic programming approaches.

3.3. Greedy algorithm

The first approach to matrix recovery is to use greedy strategy. The algorithm 1 builds the tree step-by-step, adding edges with highest probability, starting from the root. The algorithm stops early if the depth of the tree reached defined limit. Therefore the following corollary is true.

Algorithm 1: Greedy algorithm

```
Data: Matrix of the edge probabilities $\mathbf{P}$
Result: Recovered model $f$
Initialise set of open vertices $S = \{*\}$;
while $S \neq \emptyset$ and maximum complexity is not reached do
    Extract vertex $i$ from $S$;
    if $i$ is a variable then
        continue;
    Select vertex $j = \arg\max_j \mathbf{P}_{ij}$ (the vertex with the highest edge probability);
    Grow tree $f$ with edge $(i, j)$;
    Add $j$ to the set of open vertices $S$;
```

Corollary 1. *The greedy algorithm of matrix recovery has $O(1)$ complexity.*

The corollary 1 also implies that the greedy algorithm is the fastest way to recover tree from matrix of edge probabilities.

3.4. Dynamic programming

The second approach to matrix recovery is to use dynamic programming approach. In this case on each step the of the algorithm 2 the problem of tree recovery is divided into smaller problem, which are combined to maximise some score $s(f)$.

There are two possible variants for score function $s(f)$:

- $s(f) = \prod\limits_{e \in f} P_e$, i.e. the product of all edges probabilities (tree likelihood);
- $s(f) = \frac{1}{n} \sum\limits_{e \in f} P_e$, i.e. score is the average probability of the edges in the tree.

Intuitively, the former score function penalises deep trees heavily, while the latter allows more complex models.

It is straightforward to show that algorithm 2 uses dynamic programming approach.

Bellman's principle of optimality. *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.*

Corollary. *Algorithm 2 satisfies Bellman's principle of optimality.*

Proof. Consider arbitrary step of the algorithm. The initial state is a given tree f' and initial decisions are vertex choices that lead to the construction of such tree.

Then the algorithm finds the best subtree given the initial state, satisfying principle of optimality. ■

Algorithm 2: Recursive procedure $r(\mathbf{P}, f, i)$ for dynamic programming

```
Data: Matrix of the edge probabilities $\mathbf{P}$; current tree $f$; leaf vertex $i$ of $f$
Result: $\hat{f}, s(\hat{f})$. $\hat{f}$ is the best continuation of $f$ and has $i$ as its root.
if $i$ is a variable then
    return $i$, 1
for each unused vertex and variable $j$ do
    $f_j = f + (i, j)$  (grow tree $f$ with the edge $(i, j)$);
    $\hat{f}_j, s(\hat{f}_j) = r(P, f_j, j)$  (find optimal continuation for $f_j$);
$\hat{f} = \arg\max_{f_j} s(\hat{f}_j + (i, j))$  (select optimal continuation for $f$);
$s(\hat{f}) = \max_{f_j} s(\hat{f}_j + (i, j))$;
return $\hat{f}, s(\hat{f})$
```

3.5. Parametrisation

In previous sections the method for building mathematical expressions was introduced. Next step is to expand it to parametric case. It allows the method to work on real data and provides better approximation quality.

Suppose the nonparametric mathematical expression f is an output of some recovery algorithm from previous sections. From the section 2.1,

$$f = g_1 \circ g_2 \circ \dots \circ g_k, \tag{9}$$

where g_i is a nonparametric primitive function. To parametrise model f, let us parametrise each primitive function g_i:

$$g_i(\mathbf{x}, \alpha_{i1}, \alpha_{i0}) = \alpha_{i1} g_i(\mathbf{x}) + \alpha_{i0} \tag{10}$$

The parameters of model f are the parameters of its primitive functions:

$$f(\mathbf{x}) \to f(\mathbf{x}, \boldsymbol{\alpha}) \tag{11}$$

The resulting function is differentiable and the optimal parameters are found using gradient descent.

The proposed method is described in algorithm 3 and algorithm 4.

```
Algorithm 3: Training procedure
  Data: Meta learning dataset $\mathfrak{D} = \{D_i = (\mathbf{X}_i, \mathbf{y}_i), f_i\}_{i=1}^m$
  Result: Optimal meta model $\mathfrak{g}$
  for each base problem $D_i$ do
      remove parameters (constants) from model $f_i$;
      represent model $f_i$ with adjacency matrix $\mathbf{Z}_{f_i}$ of its corresponding tree $\Gamma_{f_i}$;
      represent base problem $D_i$ with a vector $\mathbf{d}_i = [\text{vec}(\mathbf{X}_i), \mathbf{y}_i]^T$
  train multi label classifier $g_{\text{clf}}$ on the set of pairs $\{(\mathbf{d}_i, \mathbf{Z}_{f_i})\}$;
```

```
Algorithm 4: Inference procedure
  Data: Base problem $D = (\mathbf{X}, \mathbf{y})$
  Result: Optimal model $f$
  represent base problem $D$ with a vector $\mathbf{d} = [\text{vec}(\mathbf{X}), \mathbf{y}]^T$;
  predict probability matrix $\mathbf{P}_f : \mathbf{P}_f = g_{\text{clf}}(\mathbf{d})$;
  recover adjacency matrix $\mathbf{Z}_f : \mathbf{Z}_f = g_{\text{rec}}(\mathbf{P}_f)$;
  parametrise the model $f \to f(\boldsymbol{\alpha})$;
  find optimal $\boldsymbol{\alpha}$ using gradient descent;
```

4. COMPUTATIONAL EXPERIMENT

The proposed method was tested on generated and real data. The goal of the experiment is to compare variations of the algorithm on synthetic data, and then compare its performance with symbolic regression on the real data. All the experiments were conducted on 1-D time series data.

4.1. Synthetic data

The goal of this experiment is to prove that meta learning method works and test different variations of it. For all the experiments in this study we chose the following properties of the models:

- the depth of the tree doesn't exceed 10,
- Binary operators: $+, \times$,
- Unary operators: $\sin, \cos, \exp, \log, \frac{1}{x}, \sqrt{x}, x^2$.

The full scheme of generating synthetic data can be found in the algorithm 5. The difference between parametric and nonparametric setup is the presence of the parameters in the tree on the step 3.

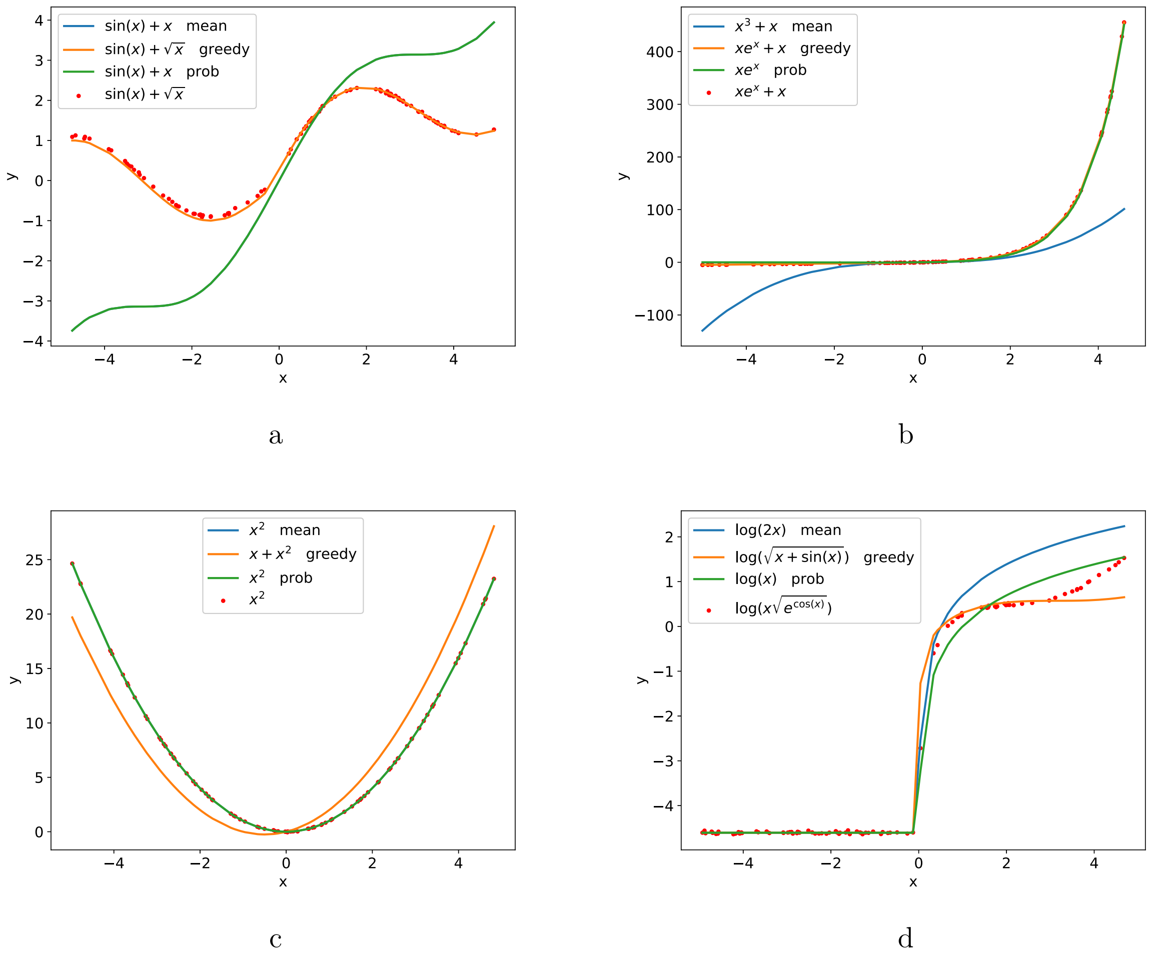
Algorithm 5: Generate data for synthetic experiment

```
   Result: Synthetic dataset $\mathfrak{D}$
1  for $i = 1, \dots, m$ do
2      sample $n$ points $\mathbf{x} = \{x_i\}_{k=1}^{n}$ uniformly from $[-5, 5]$;
3      create random tree $\Gamma_{f_i}$ (start building tree from root "*", sampling operators or
         variables with equal probabilities);
4      generate target variable $\mathbf{y}$ and add gaussian noise: $y_k = f_i(x_k) + \mathcal{N}(0, 0.05)$;
5      base problem $D_i = (\mathbf{x}, \mathbf{y})$;
6      add problem-model pair $(D_i, f_i)$ to the meta learning dataset $\mathfrak{D}$;
7  split dataset $\mathfrak{D}$ into train $\mathfrak{D}_{\text{train}}$ and test $\mathfrak{D}_{\text{test}}$;
```

4.1.1. Nonparametric approach

In this experiment the size of the dataset $\mathfrak{D}$ is 5000, $|\mathfrak{D}_{\text{train}}| = 4500$, $|\mathfrak{D}_{\text{test}}| = 500$. Greedy algorithm and two variations of dynamic programming were tested as recovery functions. Random forest, neural network with 2 hidden layers and logistic regression were variations of classification functions.

The examples of generated models for the test set are shown on the figure 2. The red dots are generated data and coloured lines are models, obtained with various recovery functions. Random forest was used as a classification algorithm for generation of these examples.



a

b

c

d

Figure 2: Performance of nonparametric approach on the test set.

These examples shows that proposed method not only approximates data well, but also recovers the structure of the model correctly in most cases. This observation proves, that in parametric case we will achieve high quality not because parameters tuning, but because of correct structure prediction. Full results are shown in the table 1. The numbers in the table is an error of meta model on the test set, as defined in (6). Random forest is the classification method of highest quality in our experiments. As expected the performance of greedy algorithm as a recovery function is worse than dynamic programming.

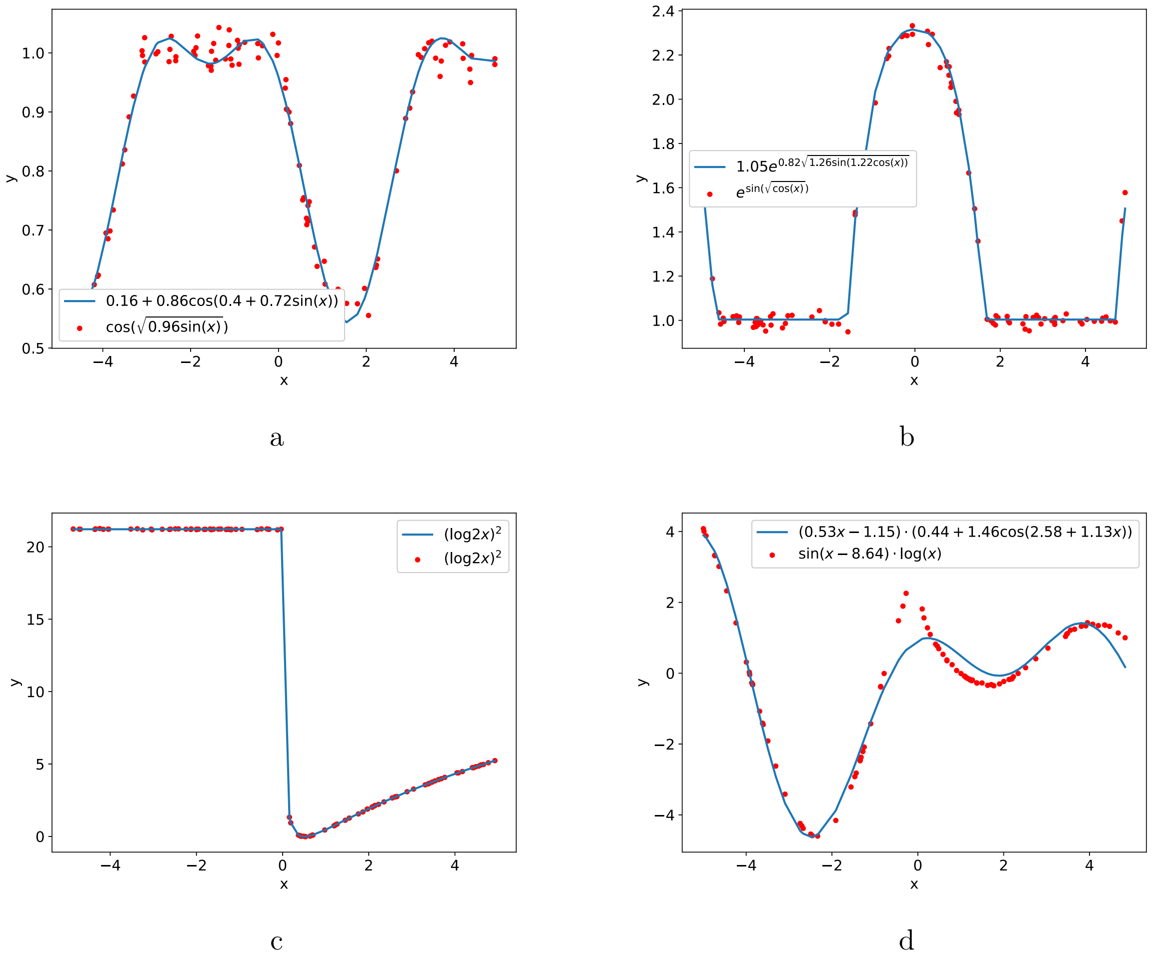
Table 1: Results in a nonparametric case.

	Random Forest	*Neural network*	*Logistic regression*
Greedy algorithm	5.45	5.81	6.3
DP (tree likelihood)	5.41	5.65	5.97
DP (mean probability)	**5.32**	5.72	6.12

4.1.2. Parametric approach

The setup of the experiment in parametric settings is the same, as in nonparametric case.

Our conclusions from nonparametric case stay true in parametric case as well. Random forest is the best classifier, and greedy algorithm has larger error than dynamic programming on the test set. The examples of models for the test set, obtained from random forest and tree likelihood recovery function are shown in the figure 3. Red dots are ground truth data and blue line is a predicted model. The results of the experiment are shown in the table 2.



a

b

c

d

Figure 3: Performance of parametric approach on the test set.

Table 2: Results in a parametric case.

	Random Forest	*Neural network*	*Logistic regression*
Greedy algorithm	7.02	7.13	7.35
DP (tree likelihood)	**6.88**	6.93	7.01
DP (mean probability)	6.92	6.94	6.99

These examples show that parametric method correctly recovers underlying model from the data. Even if it fails to predict the structure exactly, introduction of parameters allows model to have low error. In some cases the structure and the parameters are recovered exactly.

4.2. Real data

The goal of the experiment on the real data is to show that the proposed method is faster than symbolic regression, without significant drop in quality of the predicted models. The real experiment is conducted on three time series datasets.

To compare the performance of meta model and symbolic regression, we need to fit the latter on each base problem. The algorithm 6 shows the procedure for extracting meta learning dataset from real time series data.

Algorithm 6: Generate data for real experiment

```
  Result: Real dataset $\mathfrak{D}$
1 for $i = 1, \dots, m$ do
2     sample a segment of $n$ points from time series: $\mathbf{x} = \{x_i\}_{k=1}^n, \mathbf{y} = \{y_i\}_{k=1}^n$;
3     find the optimal approximating model $f_i$ for the segment using symbolic
        regression;
4     base problem $D_i = (\mathbf{x}, \mathbf{y})$;
5     add problem-model pair $(D_i, f_i)$ to the meta learning dataset $\mathfrak{D}$;
6 split dataset $\mathfrak{D}$ into train $\mathfrak{D}_{\text{train}}$ and test $\mathfrak{D}_{\text{test}}$;
```

In this experiment we compare the quality of models obtained from meta model and from symbolic regression. We also compare the time of finding optimal model on the step 3 and of inference procedure of meta model (algorithm 4). The greedy algorithm is selected as a recovery function for the fastest inference of the proposed method. The random forest with 500 estimators is selected as a classification function in all of the following experiments.

Accelerometer time series. The first real data of our experiment are accelerometer time series [19]. Time series are measurements of acceleration from wrist-worn accelerometer. The sampling rate is 32Hz. There are 14 different activities present in the data (walking, going upstairs, eating etc.). Time series were collected from activity of 16 people. Each class for each person was divided into segments of length $n = 100$. Total number of sampled segments $m = 3000$, 2500 of them are train problems and 500 are test. The example of time series segment for walking and corresponding symbolic regression model are shown on figure 4.

Mean error and mean inference speed of symbolic regression and proposed approach on test data is shown in table 3. The proposed approach significantly lowers inference type without damaging quality of the model. This is desirable pattern and proves the correctness of our method.

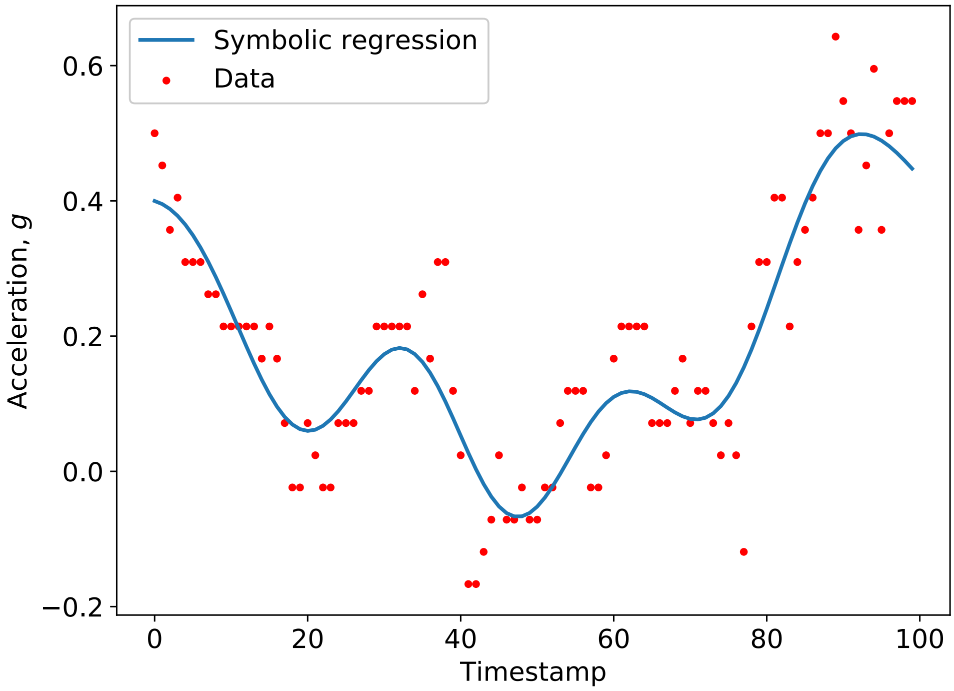


Figure 4: Example of time series segment

Table 3: Results comparison on accelerometer dataset

	MSE	Inference speed, sec.
Symbolic regression	0.052	5.12
Meta model	0.054	0.23

Daily foreign exchange rates. The dataset [20] contains time series of exchange rate between USD and foreign currency for the period from 31 December 1979 to 31 December 1998. The time series contain 4770 data points. Using algorithm 6 we generate $m = 1000$ time series segments of length $n = 100$. The size of the training set and test set is 900 and 100 accordingly. The example of the time series segment and corresponding symbolic regression model are shown on figure 5.

Mean squared error and mean inference speed of two approaches are shown in the table 4. The proposed meta learning method provides 20x speedup of inference type in comparison with symbolic regression approach. Moreover, the models from meta model are marginally worse than those, obtained with symbolic regression.

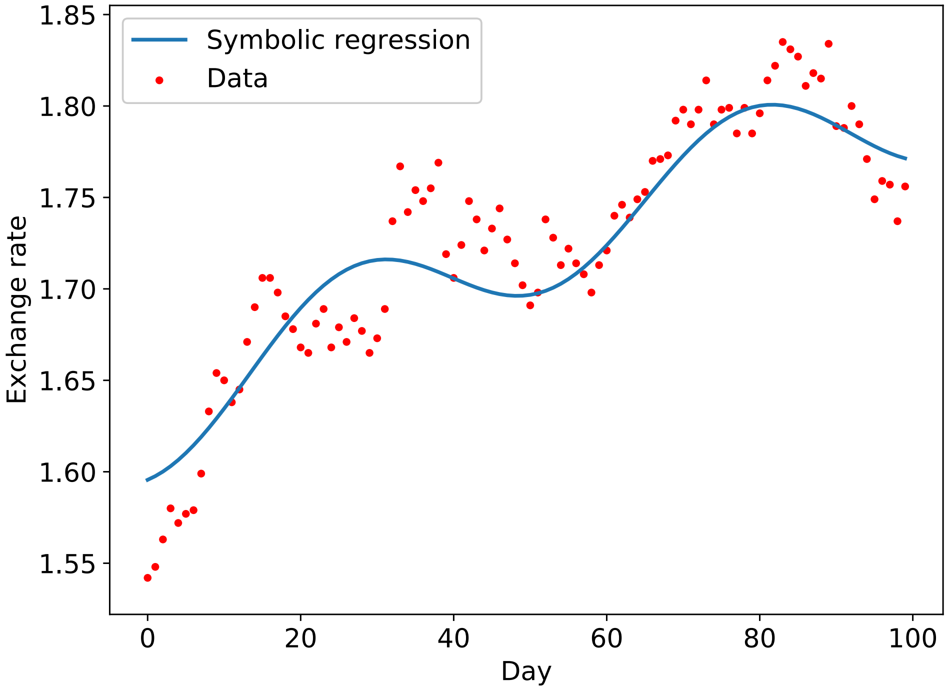


Figure 5: Example of time series segment

Table 4: Results comparison on exchange dataset

	MSE	Inference speed, sec.
Symbolic regression	0.012	6.02
Meta model	0.014	0.28

Stock prices. The dataset [21] contains time series of IBM common stock closing prices for the period from 2 Jan 1962 to 31 Dec 1965. The time series contain 1008 data points. Using algorithm 6 we generate $m = 500$ time series segments of length $n = 100$. The size of the training set and test set is 400 and 100 accordingly. The example of the time series segment and corresponding symbolic regression model are shown on figure 6.

Comparison of the two approaches on the stock data is shown in 5. The speedup and the performance of the proposed method is very similar to the result on the exchange dataset. Meta learning approach gives great speedup without big sacrifice in a quality of the models.

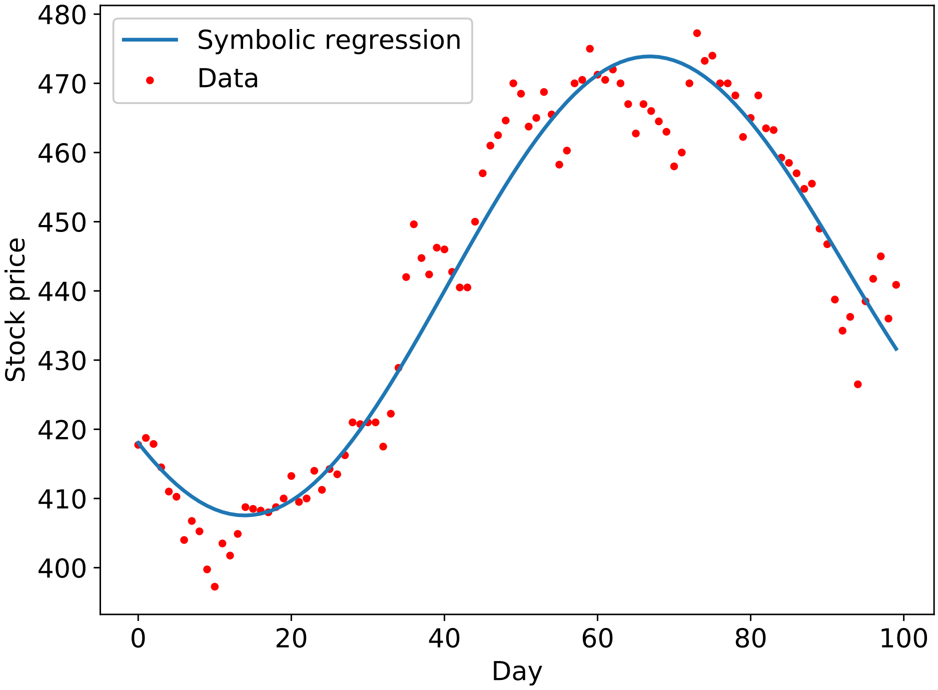


Figure 6: Example of time series segment

Table 5: Results comparison on stock price dataset

	MSE	Inference speed, sec.
Symbolic regression	3.13	6.34
Meta model	3.22	0.31

4.3. Discussion

Our experiments show that meta learning approach for generating symbolic regression models indeed works. In nonparametric case its predictions are mostly accurate. Dynamic programming paradigm to tree recovery outperforms greedy strategy in terms of quality of models.

Nonetheless greedy strategy can be more useful in applications, because the speed of model generation is much greater. Experiments on parametric data show that parameters allow proposed method to recover good model, even if the structure was wrongly predicted. This implies that introducing parameters improves method's robustness and allow it to approximate real data. In the experiment on the real datasets, the speed of building models using proposed method is significantly greater that that of symbolic regression. Meta learning approach is thus suitable for applications where the speed of building interpretable models is of high importance.

Future work. Proposed method has several shortcomings which will be addressed in future work. First, it is important to expand the method to multi dimensional case. Second, we hope to make the approach work on data without order. Both these points require different approach to encoding of the base problem, which is efficient and is not dependant on the order of objects in the data. The other goal which we have for future work is to make the method work end-to-end, i.e. build differentiable mapping from the space of base problems to the space of valid model trees.

5. CONCLUSION

This study is dedicated to generating models in a form of mathematical expressions. We propose meta learning approach to automate this process and avoid costly symbolic regression. The meta learning problem is posed as prediction of the tree structure of the model. The meta learning dataset are pairs of base problems and corresponding models. The two-staged algorithm for predicting tree structure was proposed. We proposed several variations of tree recovery. Parametric and nonparametric cases were described.

The computational experiment was conducted on synthetic and real data. The experiments on synthetic data proved the ability of the meta learning approach to find good models for unseen data. Experiment on real time series of accelerometer and financial data showed effectiveness of the proposed approach in comparison with symbolic regression.

REFERENCES

[1] John R Koza. *Genetic Programming II, Automatic Discovery of Reusable Subprograms.* MIT Press, Cambridge, MA, 1992.

[2] Ivan Zelinka, Zuzana Oplatkova, and Lars Nolle. Analytic programming–symbolic regression by means of arbitrary evolutionary algorithms. *Int. J. of Simulation, Systems, Science and Technology*, 6(9):44–56, 2005.

[3] Lawrence Davis. Handbook of genetic algorithms. 1991.

[4] Helen E Johnson, Richard J Gilbert, Michael K Winson, Royston Goodacre, Aileen R Smith, Jem J Rowland, Michael A Hall, and Douglas B Kell. Explanatory analysis of the metabolome using genetic programming of simple, interpretable rules. *Genetic Programming and Evolvable Machines*, 1(3):243–258, 2000.

[5] Carmen G Moles, Pedro Mendes, and Julio R Banga. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome research*, 13(11):2467–2474, 2003.

[6] AS Kulunchakov and VV Strijov. Generation of simple structured information retrieval functions by genetic algorithm without stagnation. *Expert Systems with Applications*, 85:221–230, 2017.

[7] Damian R Eads, Daniel Hill, Sean Davis, Simon J Perkins, Junshui Ma, Reid B Porter, and James P Theiler. Genetic algorithms and support vector machines for time series classification. In *Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation V*, volume 4787, pages 74–86. International Society for Optics and Photonics, 2002.

[8] Athanasios Tsakonas, Georgios Dounias, Jan Jantzen, Hubertus Axer, Beth Bjerregaard, and Diedrich Graf von Keyserlingk. Evolving rule-based systems in two medical domains using genetic programming. *Artificial Intelligence in Medicine*, 32(3):195–216, 2004.

[9] Omid E David and Iddo Greental. Genetic algorithms for evolving deep neural networks. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 1451–1452. ACM, 2014.

[10] Артем Максимович Бочкарев, Иван Львович Софронов, and Вадим Викторович Стрижов. Порождение экспертно-интерпретируемых моделей для прогноза проницаемости горной породы. *Системы и средства информатики*, 27(3):74–87, 2017.

[11] Cheng-Chieh Li, Jung-Chun Liu, Chu-Hsing Lin, and Winston Lo. On the accelerated convergence of genetic algorithm using gpu parallel operations. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing 2015*, pages 1–16. Springer, 2016.

[12] Christiane Lemke, Marcin Budka, and Bogdan Gabrys. Metalearning: a survey of trends and technologies. *Artificial intelligence review*, 44(1):117–130, 2015.

[13] Pavel Brazdil and Christophe Giraud-Carrier. Metalearning and algorithm selection: progress, state of the art and introduction to the 2018 special issue, 2018.

[14] Sebastian Nowozin, Christoph H Lampert, et al. Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4):185–365, 2011.

[15] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364*, 2018.

[16] Emilio Parisotto, Abdel-rahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou, and Pushmeet Kohli. Neuro-symbolic program synthesis. *arXiv preprint arXiv:1611.01855*, 2016.

[17] Raymond Hendy Susanto and Wei Lu. Semantic parsing with neural hybrid trees. In *AAAI*, pages 3309–3315, 2017.

[18] David Alvarez-Melis and Tommi S Jaakkola. Tree-structured decoding with doubly-recurrent neural networks. 2016.

[19] Dataset for ADL Recognition with Wrist-worn Accelerometer Data Set. `https://archive.ics.uci.edu/ml/datasets/Dataset+for+ADL+Recognition+with+Wrist-worn+Accelerometer`. [Online; accessed 29-May-2018].

[20] Daily foreign exchange rates, 31 December 1979 – 31 December 1998. `http://bit.ly/1XonNrs`. [Online; accessed 29-May-2018].

[21] IBM common stock closing prices. http://bit.ly/2H0Nf1w. [Online; accessed 29-May-2018].