Structure learning for model generation

Artem Bochkarev
Advisors: Maxim Fedorov, Vadim Strijov

Skolkovo Institute of Science and Technology
Moscow Institute of Physics and Technology

Introduction

Problem

- Building machine learning models is not automated
- Symbolic regression is a computationally heavy algorithm

Aim and objectives

- Automate building symbolic regression models
- Explore meta learning approach for symbolic regression

Literature

- Genetic programming
  - Kulunchakov, A. S., & Strijov, V. V. (2017). Generation of simple structured information retrieval functions by genetic algorithm without stagnation. *Expert Systems with Applications, 85, 221-230.*
  - Koza, J. R. (1994). Genetic programming as a means for programming computers by natural selection. *Statistics and computing, 4(2), 87-112.*
- Meta learning
  - Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578.*
  - Lemke, C., Budka, M., & Gabrys, B. (2015). Metalearning: a survey of trends and technologies. *Artificial intelligence review, 44(1), 117-130.*
- Tree prediction
  - Alvarez-Melis, D., & Jaakkola, T. S. (2016). Tree-structured decoding with doubly-recurrent neural networks.
  - Jin, W., Barzilay, R., & Jaakkola, T. (2018). Junction Tree Variational Autoencoder for Molecular Graph Generation. *arXiv preprint arXiv:1802.04364.*

Problem statement

Base problem

The regression problem. Let $\mathbf{X}$ be the feature matrix and $\mathbf{y}$ be the target variables.

Base problem satisfies the following conditions.

Base problem conditions

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

Base problem description is a dataset $D = (\mathbf{X}, \mathbf{y})$ which combines features and target variables.

Problem statement

We search the space of symbolic mathematical expressions $\mathfrak{F}$ for models.

Mathematical expression

- Generated by grammar G:

$$g \to B(g, g)|U(g)|S,$$

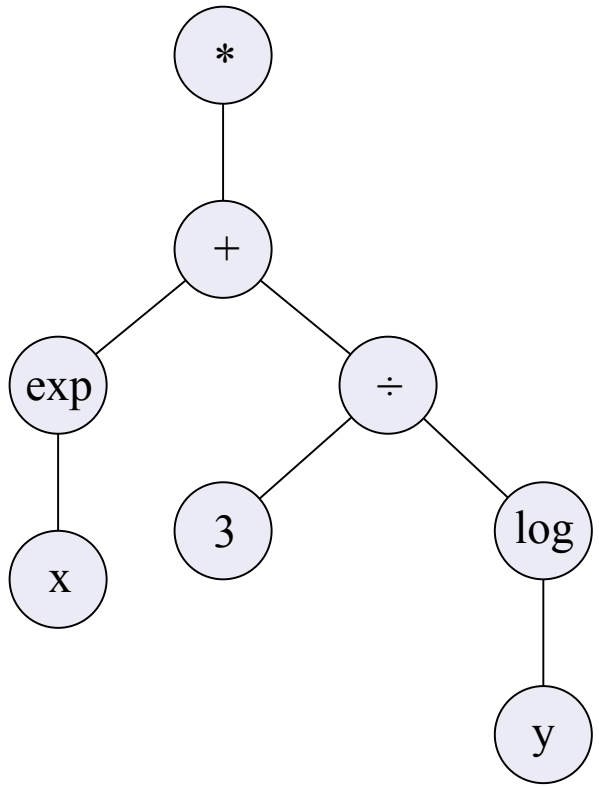
  where B are binary functions $(+, *)$, U are unary functions $(\text{sqrt}, \log, \exp)$ and S are variables.
- $f = g_1 \circ g_2 \circ \cdots \circ g_k$
- Mathematical expression f can be represented with a tree Γ_f

Problem statement

Tree Γ_f corresponding to model f satisfies the following conditions:

Tree Γ_f

1. Symbol $*$ is a root of the tree;
2. Leafs of Γ_f contain variables $x \in S$.
3. Non-leaf vertices v contain corresponding functions g;
4. $\#\text{child}(v) = \text{arity}(g)$;
5. If v_j is a child of v_j, then $\text{dom}(g_j) \supset \text{cod}(g_i)$;
6. Children of g are ordered;



$$f = e^x + \frac{3}{\log(y)}$$

Problem statement

Meta learning dataset

Let the set of pairs $\mathfrak{D} = \{D_i = (\mathbf{X}_i, \mathbf{y}_i), f_i\}_{i=1}^m$ be the data for the meta learning problem. The following conditions are satisfied:

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

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

Meta learning problem

Given the meta learning dataset $\mathfrak{D}$, find the optimal meta model $\mathfrak{g} : D \to f$ which minimizes the error on all base problems:

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

Solution

In order to solve meta learning problem we define the representations for base problem dataset D and model f.

Model representations

1. Symbolic expression of f
2. Tree Γ_f
3. Adjacency matrix $\mathbf{Z}_f$ of tree Γ_f

We select Z_f as a representation of a model.

Base problem representation

The vector of concatenation $\mathbf{d} = [\text{vec}(\mathbf{X}), \mathbf{y}]^T$ is a representation of a dataset.

Solution

Meta model decomposition

With selected representations of base problem and model, the meta model $\mathfrak{g} : D \to f$ is a mapping between the space of vectors $\mathbb{R}^n$ and the space of adjacency matrices of trees $\mathbb{Z}$.
We decompose $\mathfrak{g}$ into two functions:

$$f = \mathfrak{g}(D) = g_{\text{rec}}(g_{\text{clf}}(D)),$$

which are recovery and classification functions.

Solution

Classification

Classification function $g_{\text{clf}} : \mathbb{R}^n \to \mathbb{P}$ is a mapping between datasets and the space of probability matrices.

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

where $\mathbf{P}_f$ is a matrix of probability of edges in a tree Γ_f. g_{clf} is a classification algorithm (logistic regression, random forest).

Matrix recovery

Classification function $g_{\text{rec}} : \mathbb{P} \to \mathbb{Z}$ is a mapping between the space of matrices of edge probabilities to the space of adjacency matrices of trees. We propose two methods for matrix recovery g_{rec}:

- Greedy algorithm
- Dynamic programming

Greedy strategy

Algorithm 1 Greedy algorithm

Input: Matrix of the edge probabilities $\mathbf{P}$
Output: Recovered model f
  Initialize 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
    end if
    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
  end while

Dynamic programming

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

Input: Matrix of the edge probabilities $\mathbf{P}$; current tree f; leaf vertex i of f
Output: $\hat{f}, s(\hat{f})$. $\hat{f}$ is the best continuation of f and has i as its root.

if i is a variable **then**
  return i, 1
end if
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)
end for
$\hat{f} = \arg\max_{f_j} s(\hat{f}_j + (i, j))$ (select optimal continuation for f)
$s(\hat{f}) = \max_{f_j} s(\hat{f}_j + (i, j))$
return $\hat{f}, s(\hat{f})$

Dynamic programming

Choice of scoring function

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

The former score function penalizes deep trees heavily, while the latter allows more complex models.

Dynamic programming

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.

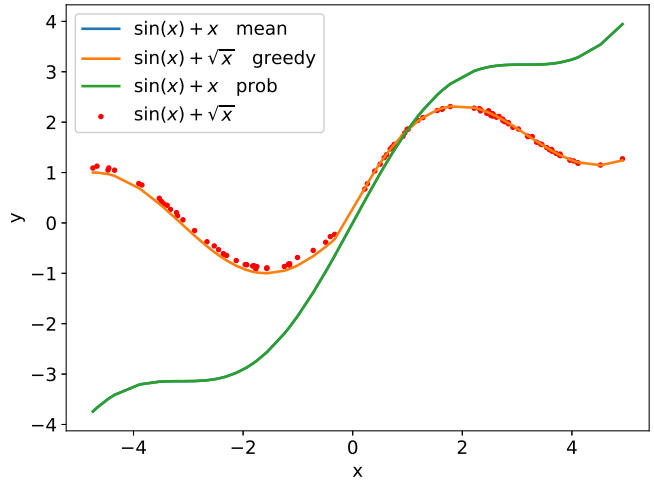
Experiment

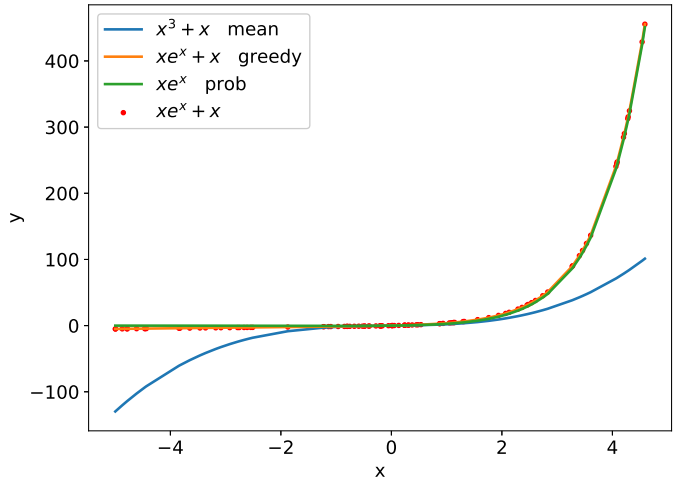
The experiment was conducted on the synthetic data.

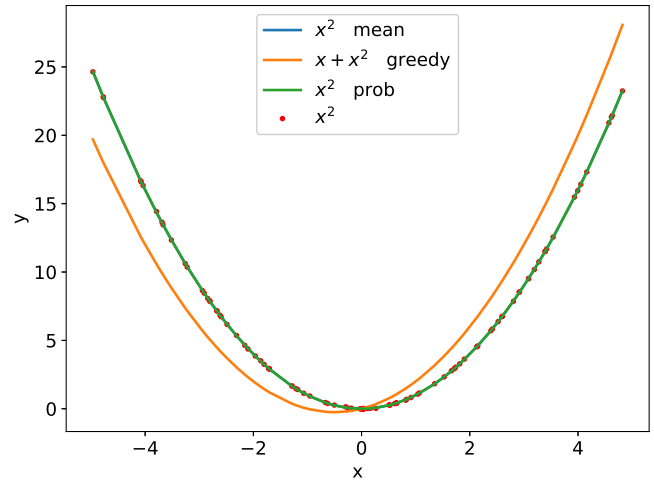
Experiment scheme

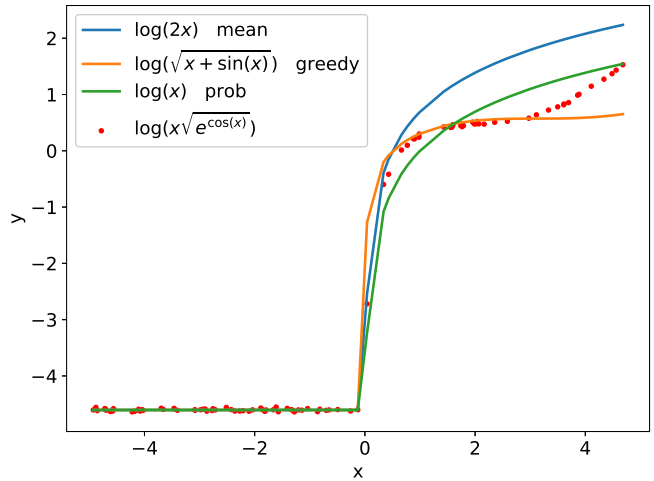
1. Generate ≈ 5000 1-D base problems
   - $\mathbf{X}$ is uniformly distributed on $[-1, 1]$
   - f is a randomly generated non-parametric mathematical expression
   - $\mathbf{y} = f(\mathbf{X}) + \mathcal{N}(0, 0.05)$
2. Split base problems into train and test set
3. Train g_{clf} on a train set
4. Predict matrices of edge probabilities $\mathbf{P}$ for test base problems and recover models from them using g_{rec}

Performance on base problems from test set









Parametric approach

In order to approximate real datasets we introduce parametric approach. Let the best non-parametric model be $f = f_1 \circ \cdots \circ f_n$.

Parametrization

Introduce parameters for each primitive function f_i:

$$f_i(\mathbf{x}, \alpha_{i1}, \alpha_{i0}) = \alpha_{i1} f_i(\mathbf{x}) + \alpha_{i0}.$$

The function f is now parametrized with parameters of its primitives:

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

The resulting function is differentiable, vector of parameters $\boldsymbol{\alpha}$ is found using gradient descent.

Method overview

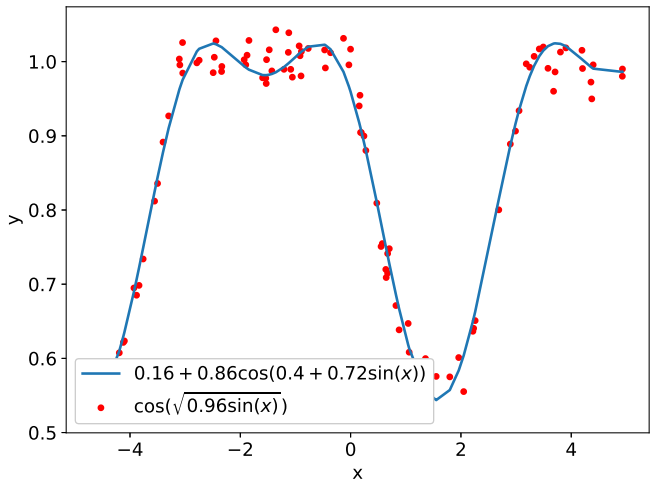
Training phase

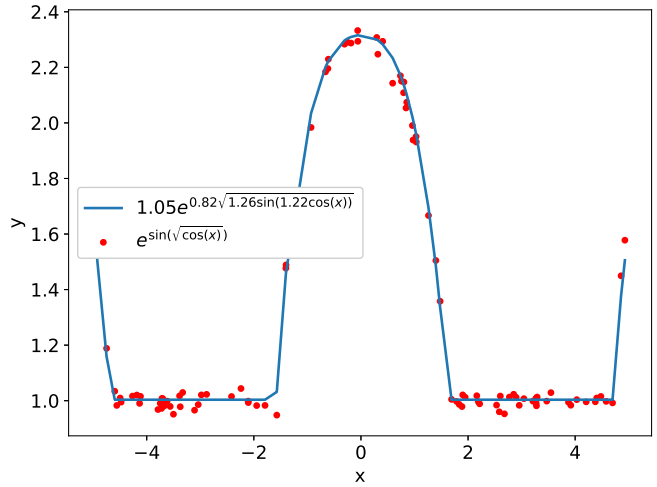
1. Remove constants from models f
2. Train g_{clf} to predict probability matrix $\mathbf{P}$

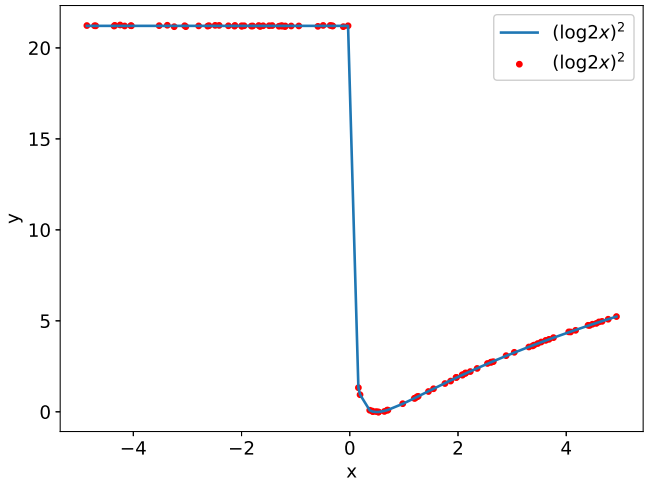
Testing phase

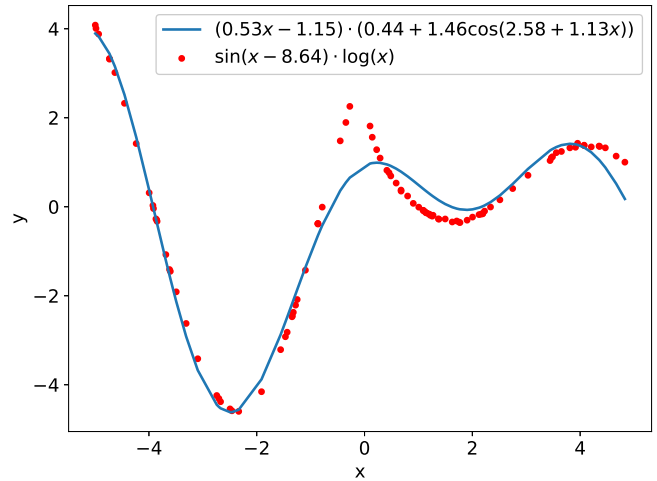
1. Predict matrix $\mathbf{P}$ using g_{clf}
2. Recover model f using g_{rec}
3. Parametrize model f
4. Find optimal parameters α using gradient descent

Performance of parametric approach









Conclusions

- Developed new framework for meta learning
- Conducted parametric and non-parametric experiments
- Method produces parametric symbolic models of good quality