Two-Stage Coefficient Estimation in Symbolic Regression for Scientific Discovery

Masahiro Negishi[*]
OMRON SINIC X
The University of Tokyo
negishi-masahiro1110@g.ecc.u-tokyo.ac.jp

Yoshitomo Matsubara[†]
Spiffy AI
yoshitomo@spiffy.ai

Naoya Chiba
Osaka University
chiba@nchiba.net

Ryo Igarashi
OMRON SINIC X
ryo.igarashi@sinicx.com

Yoshitaka Ushiku
OMRON SINIC X
yoshitaka.ushiku@sinicx.com

Abstract

Symbolic regression (SR) identifies mathematical equations behind data, which plays a crucial role in scientific discovery. Most SR methods involve coefficient estimation, a process of adjusting the coefficient values in the estimated equation to fit the data. However, existing coefficient estimation methods are prone to fail when estimating exponential and non-exponential coefficients simultaneously. To address this challenge, we propose an algorithm that separately estimates exponential and non-exponential coefficients[3]. Our method finds the ground truth coefficient values in a larger number of problems than existing methods, and the performance can be further improved given an order-level initial estimate of the coefficients. We also analyze the time complexity of our algorithm both theoretically and experimentally.

1 Introduction

Symbolic regression (SR) is the task of estimating the underlying mathematical equation behind observed data [9, 13]. Specifically, given n data points $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, the goal is to find a function $f : \mathbb{R}^d \to \mathbb{R}$ such that $y_i = f(\boldsymbol{x}_i)$ for all i. Here, the ground truth f is assumed to be a simple equation composed of predefined mathematical operations, such as $+$(add), $\times$(mul), $\log$, $\sin$, $\exp$, etc. Consequently, the estimated function $\hat{f}$ in SR is generally more interpretable than the complex functions used in typical machine learning, such as neural networks. This high interpretability makes SR particularly useful in various scientific fields, including physics [5, 17, 19], materials science [1, 20, 21], and astrophysics [12], where understanding the laws behind the data is crucial.

To date, numerous SR methods have been proposed, including genetic programming [6–8], deep learning [2, 4, 10, 16], and others [11, 18]. They typically include the following procedure:

1. **Skeleton Estimation:** Estimate a "skeleton" f_{skl} of f. We call the estimated skeleton $\hat{f}_{\text{skl}}$.
2. **Coefficient Estimation:** Optimize coefficients in $\hat{f}_{\text{skl}}$ to better fit $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, resulting in $\hat{f}$.

Here, f_{skl} is a formula derived by replacing the numerical coefficients in f with special variables $\boldsymbol{c}$. We divide $\boldsymbol{c}$ into $\boldsymbol{c}_{\text{exp}}$ and $\boldsymbol{c}_{\text{non}}$, representing exponential and non-exponential coefficients, respectively.

[*]This work was done while the first author was a research intern at OMRON SINIC X Corporation.
[†]This work was done prior to joining Spiffy AI.
[3]https://github.com/omron-sinicx/srsd-coeff-optim

Machine Learning and the Physical Sciences Workshop, NeurIPS 2024.

For example, $f_{\text{skl}}(\boldsymbol{x}; \boldsymbol{c}) = c_1 \times x_0^{c_0} + \log(x_1 + c_2)$, $\boldsymbol{c} = [2, \pi, -3]$, $\boldsymbol{c}_{\text{exp}} = [2]$, and $\boldsymbol{c}_{\text{non}} = [\pi, -3]$ if $f(\boldsymbol{x}) = \pi \times x_0^2 + \log(x_1 - 3)$. In a slight abuse of notation, we will use $\boldsymbol{c}$, $\boldsymbol{c}_{\text{exp}}$, and $\boldsymbol{c}_{\text{non}}$ for both the coefficient variables and their assigned numerical values. The above two steps alleviate the difficulty of SR by decomposing f into its structure and coefficient values, and estimating them individually.

While both steps are essential for SR, previous studies have mainly focused on improving skeleton estimation. This paper demonstrates that existing coefficient estimation methods [3, 14] do not work well when $\hat{f}_{\text{skl}}$ has exponential coefficients, which is common in scientific laws, such as the exponent -2 in $F = \frac{GmM}{r^2}$. We then propose a new optimization method that separately estimates exponential and non-exponential coefficients. We empirically show the superiority of our method with experiments on the Symbolic Regression for Scientific Discovery (SRSD) dataset [15].

2 Existing coefficient estimation methods and their limitations

The coefficient estimation methods employed in existing SR approaches fall into two categories. The first category [11, 18] assumes that $\hat{f}_{\text{skl}}$ has a simple structure like polynomials. Then, the coefficients are estimated by minimizing $\mathcal{L}(\boldsymbol{c}) \coloneqq \frac{1}{n}\sum_{i=1}^{n}(\hat{f}_{\text{skl}}(\boldsymbol{x}_i; \boldsymbol{c}) - y_i)^2$ w.r.t. $\boldsymbol{c}$, often analytically thanks to the simple structure assumption. An explicit limitation of this category is that f cannot be identified if f_{skl} does not have an assumed simple structure, which is often the case for scientific equations.

Thus, gradient-based methods, such as the Levenberg-Marquardt (LM) algorithm [14] and the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [3], are more commonly used [2, 4, 6, 7, 16]. They do not impose a simple structure assumption on $\hat{f}_{\text{skl}}$, and minimize $\mathcal{L}(\boldsymbol{c})$ using the information of $\mathcal{L}'(\boldsymbol{c})$. While these approaches accept any form of $\hat{f}_{\text{skl}}$ as long as it is differentiable, existing SR methods assumed that $\hat{f}_{\text{skl}}$ does not contain exponential coefficients. Many [4, 6, 7, 16] dealt with problems where the ground truth f does not contain exponents, which can prevent them from being used for scientific discovery. Biggio et al. [2] treated f with exponents, but required correct estimation of exponents at the skeleton estimation stage, making the stage more complex and challenging. To make SR more applicable to the identification of scientific laws, we propose an algorithm to optimize both exponential and non-exponential coefficients at the coefficient estimation stage.

3 Proposed method: two-stage optimization

As we will see in Section 4, the LM and BFGS algorithms fail to estimate coefficients mainly because estimating both $\boldsymbol{c}_{\text{exp}}$ and $\boldsymbol{c}_{\text{non}}$ simultaneously is numerically unstable. Therefore, we propose to estimate them separately. Specifically, our proposed method (Algorithm 1) iteratively performs the estimation of $\boldsymbol{c}_{\text{exp}}$ and $\boldsymbol{c}_{\text{non}}$ while always maintaining the top B candidate solutions (beam search). After L_{out} iterations, the best coefficients that minimize $\mathcal{L}(\boldsymbol{c}_{\text{exp}}, \boldsymbol{c}_{\text{non}})$ are chosen from the candidates.

The exponential coefficients $\boldsymbol{c}_{\text{exp}}$ are optimized by a simple brute-force method (Exponential part in Algorithm 1). First, each current candidate value of $\boldsymbol{c}_{\text{non}}$ is paired with all possible patterns of $\boldsymbol{c}_{\text{exp}}$ from $\Sigma^{n_{\text{exp}}}$, where Σ is a predefined set of possible exponents and n_{exp} is a number of exponential coefficients in $\hat{f}_{\text{skl}}$. Of all the pairs generated, the top B pairs that minimize $\mathcal{L}$ the most are selected.

Next, for each $(\boldsymbol{c}_{\text{exp}}^{\text{opt}}, \boldsymbol{c}_{\text{non}})$ of the top B pairs from the previous step, $\boldsymbol{c}_{\text{non}}$ is optimized (Non-exponential part in Algorithm 1). This optimization is also an iteration of two steps. In each iteration, $\boldsymbol{c}_{\text{non}}$ is first optimized with the LM or BFGS algorithm, resulting in $\boldsymbol{c}_{\text{non}}^{\text{opt}}$. Since these algorithms often yield a local optimum solution, the second step tries to jump out of the local optimum by optimizing one dimension of $\boldsymbol{c}_{\text{non}}^{\text{opt}}$. Specifically, $\boldsymbol{c}_{\text{non}}^{\text{opt}}$ is fixed except for one dimension j, and $y_i = \hat{f}_{\text{skl}}(\boldsymbol{x}_i; \boldsymbol{c}_{\text{exp}}^{\text{opt}}, \boldsymbol{c}_{\text{non}}^{\text{opt}}[:j], c, \boldsymbol{c}_{\text{non}}^{\text{opt}}[j:])$ $(1 \leq i \leq n)$ is solved w.r.t. c. Note that this is analytically solvable in most cases. For example, $y_i = c_1 \times x_{i,0}^{c_0} + \log(x_{i,1} + c_2)$ is solvable w.r.t. c_2, i.e., $c_2 = \exp(y_i - c_1 \times x_{i,0}^{c_0}) - x_{i,1}$. Averaging the resulting c over all i gives c^{opt}. Intuitively, the dimension j of $c_{\text{non}}^{\text{opt}}$ is refined using the skeleton information, which is ignored in gradient-based optimization. If the second step does not have an analytical solution w.r.t. the dimension j, the refinement of the dimension is skipped. Finally, we update the value of the dimension where the refined $c_{\text{non}}^{\text{opt}}$ minimizes the loss function the most, and move on to the next iteration.

Algorithm 1 Two-stage optimizer

Hyper Parameter: set of possible exponents Σ, beam size B, number of outer iteration L_{out}, gradient based optimizer O_{grad}, number of inner iteration L_{in}

Input: estimated skeleton $\hat{f}_{\text{skl}}$, data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, and initial values for coefficients $(\boldsymbol{c}_{\text{exp}}^{\text{init}}, \boldsymbol{c}_{\text{non}}^{\text{init}})$

Output: $\hat{f}$

$\mathcal{L}(\boldsymbol{c}_{\text{exp}}, \boldsymbol{c}_{\text{non}}) \leftarrow \frac{1}{n}\sum_{i=1}^n (\hat{f}_{\text{skl}}(\boldsymbol{x}_i; \boldsymbol{c}_{\text{exp}}, \boldsymbol{c}_{\text{non}}) - y_i)^2$ ▷ Loss to minimize.
$S \leftarrow \{(\boldsymbol{c}_{\text{exp}}^{\text{init}}, \boldsymbol{c}_{\text{non}}^{\text{init}})\}$ ▷ Set of current candidates.
$n_{\text{exp}}, n_{\text{non}} \leftarrow |\boldsymbol{c}_{\text{exp}}^{\text{init}}|, |\boldsymbol{c}_{\text{non}}^{\text{init}}|$ ▷ Number of coefficients in $\hat{f}_{\text{skl}}$.
for out-iter $= 1$ to L_{out} **do**
  /* Exponential coefficient estimation starts */
  $S_{\text{non}} \leftarrow \{\boldsymbol{c}_{\text{non}}$ for $(\boldsymbol{c}_{\text{exp}}, \boldsymbol{c}_{\text{non}})$ in $S\}$
  $S_{\text{exp-opt}} \leftarrow \text{argmin-topB}_{(\boldsymbol{c}_{\text{exp}}, \boldsymbol{c}_{\text{non}}) \in \Sigma^{n_{\text{exp}}} \times S_{\text{non}}} \mathcal{L}(\boldsymbol{c}_{\text{exp}}, \boldsymbol{c}_{\text{non}})$ ▷ Set of top-B successful candidates.
  /* Exponential coefficient estimation ends */
  /* Non-exponential coefficient estimation starts */
  $S_{\text{exp-opt-non-opt}} \leftarrow \emptyset$
  for $(\boldsymbol{c}_{\text{exp}}^{\text{opt}}, \boldsymbol{c}_{\text{non}}) \in S_{\text{exp-opt}}$ **do**
    $\boldsymbol{c}_{\text{non}}^{\text{opt}} \leftarrow \boldsymbol{c}_{\text{non}}$
    for in-iter $= 1$ to L_{in} **do**
      $\boldsymbol{c}_{\text{non}}^{\text{opt}} \leftarrow O_{\text{grad}}(\mathcal{L}, \boldsymbol{c}_{\text{non}}^{\text{opt}})$ ▷ Optimize $\boldsymbol{c}_{\text{non}}^{\text{opt}}$ using LM or BFGS algorithm.
      $S_{\text{jump}} \leftarrow \emptyset$
      for $j = 0$ to $n_{\text{non}} - 1$ **do**
        $g_i(c) \leftarrow \hat{f}_{\text{skl}}(\boldsymbol{x}_i; \boldsymbol{c}_{\text{exp}}^{\text{opt}}, \boldsymbol{c}_{\text{non}}^{\text{opt}}[:j], c, \boldsymbol{c}_{\text{non}}^{\text{opt}}[j:]) - y_i \quad (1 \le i \le n)$
        $c^{\text{opt}} \leftarrow \frac{1}{n}\sum_{i=1}^n g_i^{-1}(0)$ ▷ Analytically solvable in most cases.
        S_{jump}.add($[\boldsymbol{c}_{\text{non}}^{\text{opt}}[:j], c^{\text{opt}}, \boldsymbol{c}_{\text{non}}^{\text{opt}}[j:]]$)
      end for
      $\boldsymbol{c}_{\text{non}}^{\text{opt}} \leftarrow \text{argmin}_{\boldsymbol{c}_{\text{jump}} \in S_{\text{jump}}} \mathcal{L}(\boldsymbol{c}_{\text{exp}}^{\text{opt}}, \boldsymbol{c}_{\text{jump}})$ ▷ Change only one dimension of $\boldsymbol{c}_{\text{non}}^{\text{opt}}$.
    end for
    $S_{\text{exp-opt-non-opt}}$.add($(\boldsymbol{c}_{\text{exp}}^{\text{opt}}, \boldsymbol{c}_{\text{non}}^{\text{opt}})$)
  end for
  /* Non-exponential coefficient estimation ends */
  $S \leftarrow S_{\text{exp-opt-non-opt}}$
end for

$\boldsymbol{c}_{\text{exp}}^{\text{best}}, \boldsymbol{c}_{\text{non}}^{\text{best}} \leftarrow \text{argmin}_{(\boldsymbol{c}_{\text{exp}}, \boldsymbol{c}_{\text{non}}) \in S} \mathcal{L}(\boldsymbol{c}_{\text{exp}}, \boldsymbol{c}_{\text{non}})$ ▷ Select the best among the top-B.
$\hat{f} \leftarrow \hat{f}_{\text{skl}}(; \boldsymbol{c}_{\text{exp}}^{\text{best}}, \boldsymbol{c}_{\text{non}}^{\text{best}})$

Before moving on to the experiments, we analyze the time complexity of Algorithm 1. Since $\mathcal{L}(\boldsymbol{c})$ is the mean squared error (MSE) of n samples, its calculation takes $O(n)$ time. If we keep the top B candidates in a priority queue, the exponential part takes $O(\Sigma^{n_{\text{exp}}} \cdot B \cdot (n + \log B))$ time on each outer iteration. For the non-exponential part, suppose each optimization with the LM or BFGS takes T_{grad} time. Then, the time complexity for the non-exponential part is $O(B \cdot L_{\text{in}} \cdot (T_{\text{grad}} + n_{\text{non}} \cdot n))$ at each outer iteration. Therefore, the total time complexity is $O\left(L_{\text{out}} \cdot B \cdot (\Sigma^{n_{\text{exp}}} \cdot (n + \log B) + L_{\text{in}} \cdot (T_{\text{grad}} + n_{\text{non}} \cdot n))\right)$.

4 Experiments and discussions

In this section, we compare our proposed method with existing algorithms, and also see the effect of hyperparameters on time complexity and performance. We selected 112 out of 120 problems from the SRSD dataset[4][5][6] [15] that have at least one numerical coefficient in f, and created f_{skl} by

[4] https://huggingface.co/datasets/yoshitomo-matsubara/srsd-feynman_easy
[5] https://huggingface.co/datasets/yoshitomo-matsubara/srsd-feynman_medium
[6] https://huggingface.co/datasets/yoshitomo-matsubara/srsd-feynman_hard

$$y = \frac{3}{5} \cdot \frac{Q^2}{4\pi\epsilon a} \longrightarrow y = c_2 x_0^{c_0} x_1^{c_1}$$

$$y = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{\theta^2}{2\sigma^2}) \longrightarrow y = c_3 x_0^{c_0} \exp(c_4 x_1^{c_1} x_0^{c_2})$$

Figure 1: Examples of deriving f_{skl} from f in the SRSD dataset [15]. The coefficients and variables in the ground truth f on the left are replaced by $\boldsymbol{c}$ and $\boldsymbol{x}$ respectively.

Table 1: The number of problems ending in global optimum/local optimum/failure and the average execution time of different methods.

Optimization method	Global optimum	Local optimum	Failure	Average time [s]
BFGS	13	56	43	37.97
LM	30	35	47	35.78
Ours with BFGS	60	49	3	38.32
Ours with LM	64	46	2	37.98
Ours with LM (without Jump)	57	53	2	33.62
Ours with LM (estimated init)	84	28	0	36.64

replacing the coefficients in f with variables $\boldsymbol{c}$. See Figure 1 for examples. To focus on the coefficient estimation, we assumed a correct skeleton estimation, i.e. $\hat{f}_{\text{skl}} = f_{\text{skl}}$. Thus, the benchmark task was to estimate f from f_{skl} using $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, where we set $n = 20$. We tested the following five different methods on the task: BFGS, LM, our method (O_{grad}=BFGS), our method (O_{grad}=LM), and our method (O_{grad}=LM) without the jump step in estimating c_{non}. Each dimension of $\boldsymbol{c}^{\text{init}}$ was sampled independently from a uniform distribution on $[0, 1]$. In our proposed methods, we set $\Sigma = [-1, \pm 0.5, \pm 1.5, \pm 2, \pm 3, \pm 4, \pm 5], B = 10, L_{\text{out}} = 2$, and $L_{\text{in}} = 10$. For a fair comparison, we roughly equalized the average execution time by restarting the BFGS and the LM 200 times with different $\boldsymbol{c}^{\text{init}}$. The optimization results were classified into three types: global optimum, local optimum, and failure. We considered $\boldsymbol{c}^{\text{best}}$ to be a global optimum if the relative error between $\boldsymbol{c}^{\text{best}}$ and the true value is below 10^{-4} in all dimensions; otherwise it was a local optimum. For the BFGS and LM algorithms, a failure meant that none of the 200 optimizations finished successfully, while for our methods it meant that none of the inner optimizations by O_{grad} ended successfully.

Table 1 (except for the last row) shows the results for the five different methods. As shown, the BFGS and LM algorithms reach the global optimum in only 13 and 30 out of 112 problems, respectively, indicating that they often fail even when the true skeleton f_{skl} is known. In addition, despite being restarted 200 times, a certain number of problems end in a failure. Although not shown in Table 1, about 95% of the failures are due to either "invalid value encountered in scalar power" or "overflow encountered in scalar power", resulting from the simultaneous optimization of exponential and non-exponential coefficients. On the other hand, our methods (rows 3 and 4) are much more likely to find a global optimum, and the number of failures is much smaller. Comparing rows 4 and 5, it is apparent that the jump step helps to escape from a local optimum. Finally, the LM performs better than the BFGS both on its own and when incorporated into our proposed algorithm.

However, there are still some hard problems where our methods can only find local optimums. To study the characteristics of such problems, we visualize the ratio of global optimum, local optimum, and failure for problems with the same number of exponential and non-exponential coefficients in Figure 2. It shows that our method with the LM mainly improves the possibility of finding a global optimum in problems with few coefficients (lower left region), but it still suffers with many coefficients (upper right region). To improve results for the upper right problems, we assume that we have access to $\boldsymbol{c}^{\text{init}}$ whose order matches that of the ground truth value. In practice, such a good $\boldsymbol{c}^{\text{init}}$ can be obtained, for example, using a transformer model in the skeleton estimation [4]. As shown, access to a good $\boldsymbol{c}^{\text{init}}$ alleviates the difficulty of problems with many coefficients (Figure 2 right) and leads to a higher overall performance (last row in Table 1). In Figure 2, the LM could not find the global optimum on both problems, while our method did on the upper problem. Our method reached the global optimum in the lower problem only when the good initial estimate of $\boldsymbol{c}^{\text{init}}$ was given.

Next, we analyze how the hyperparameters B, L_{out}, and n affect the execution time and the success rate. Here, the success rate is defined as the number of problems ending in the global optimum

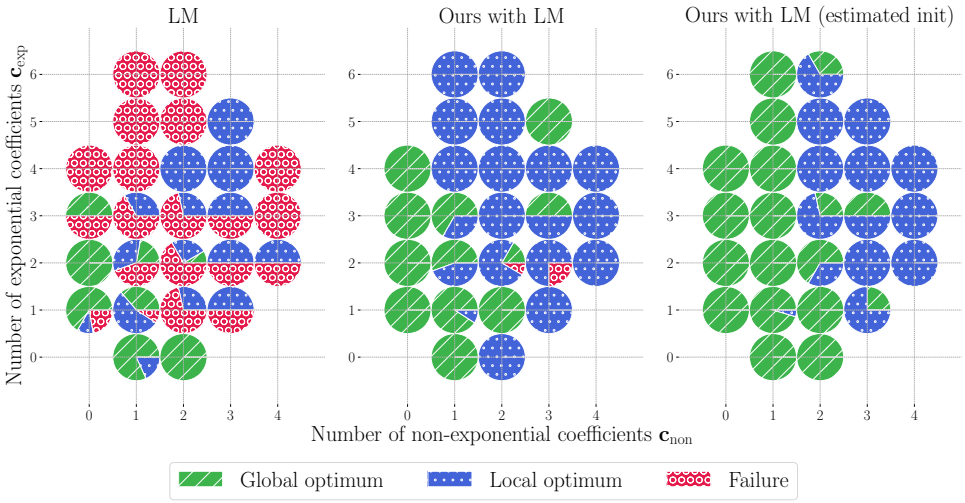

Figure 2: The ratio of the final results, grouped by the number of exponential and non-exponential coefficients. Each circle consists of problems with the same number of coefficients of both types.

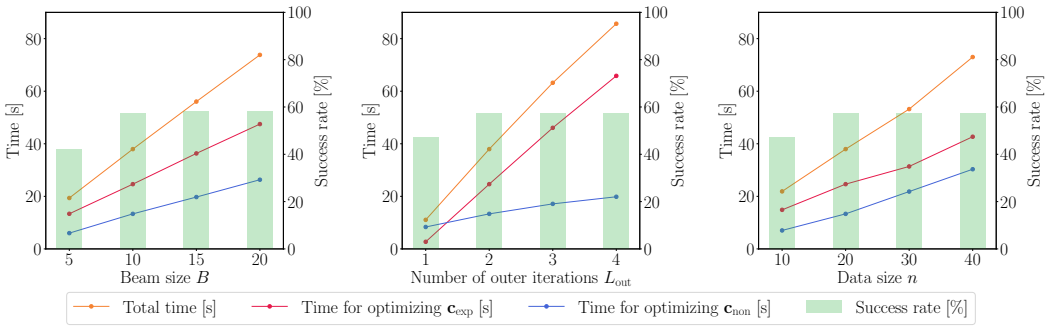

Figure 3: The execution time and success rate under different hyperparameters. The baseline is $(B, L_{\text{out}}, n) = (10, 2, 20)$, and we change one of these parameters in each plot.

over the total number of problems. As a remainder, the theoretical time complexity of Algorithm 1 is $O\left(L_{\text{out}} \cdot B \cdot (\Sigma^{n_{\text{exp}}} \cdot (n + \log B) + L_{\text{in}} \cdot (T_{\text{grad}} + n_{\text{non}} \cdot n))\right)$. Figure 3 shows that the practical exection time is generally linear in B, L_{out}, and n. It is linear in B and n, probably because $n \gg \log B$ and T_{grad} is linear in n in practice. The linearity of T_{grad} comes from the fact that computing $\mathcal{L}'(\boldsymbol{c})$ takes $O(n)$ time. If you take a closer look at the middle plot, you will notice that estimating $\boldsymbol{c}_{\text{exp}}$ takes very little time when $L_{\text{out}} = 1$. This can be explained by the fact that the brute-force estimation is done over $\Sigma^{n_{\text{exp}}}$ candidates on the first iteration, while $\Sigma^{n_{\text{exp}}} \cdot B$ candidates after the second iteration. Overall, the exponential part tends to take longer than the non-exponential part, indicating the importance of future studies on more sophisticated exponential estimation methods. In terms of the success rate, the improvement saturates at $(B, L_{\text{out}}, n) = (10, 2, 20)$ in all plots. Thus, to achieve an even higher success rate, it is necessary to improve the algorithm or the estimation of $\boldsymbol{c}^{\text{init}}$ instead of running the current one for a longer time.

5 Conclusions

We have proposed a coefficient estimation algorithm that optimizes exponential and non-exponential coefficients separately. Our method is much more likely to find the ground truth coefficient values than previous methods, performs even better given an order-level initial estimate of the coefficients, and can be combined with any skeleton estimation method. We have also analyzed how the hyperparameters affect the time complexity and the success rate of the algorithm. In future work, we will replace the brute-force algorithm for estimating exponential coefficients with a more advanced one. We will also combine our method with popular skeleton estimation methods to test performance improvements.

Acknowledgments

This work was supported by JST-Mirai Program Grant Number JPMJMI21G2, JST Moonshot R&D Program Grant Number JPMJMS2236, JSPS KAKENHI Grant Number 21K14130, and JSPS KAKENHI Grant Number 24K23910, Japan. We would also like to thank the reviewers for their insightful comments.

References

[1] Rohit Batra, Le Song, and Rampi Ramprasad. Emerging materials intelligence ecosystems propelled by machine learning. *Nature Reviews Materials*, 6(8):655–678, 2021.

[2] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *International Conference on Machine Learning*, 2021.

[3] R. Fletcher. *Newton-Like Methods*, chapter 3, pp. 44–79. John Wiley & Sons, Ltd, 2000.

[4] Pierre-alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and Francois Charton. End-to-end symbolic regression with transformers. In *Advances in Neural Information Processing Systems*, 2022.

[5] Samuel Kim, Peter Y Lu, Srijon Mukherjee, Michael Gilbert, Li Jing, Vladimir Čeperić, and Marin Soljačić. Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE transactions on neural networks and learning systems*, 32(9): 4166–4177, 2020.

[6] Michael Kommenda, Gabriel Kronberger, Stephan Winkler, Michael Affenzeller, and Stefan Wagner. Effects of constant optimization by nonlinear least squares minimization in symbolic regression. In *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, 2013.

[7] Michael Kommenda, Bogdan Burlacu, Gabriel Kronberger, and Michael Affenzeller. Parameter identification for symbolic regression using nonlinear least squares. *Genetic Programming and Evolvable Machines*, 21(3):471–501, 2020.

[8] John R. Koza and Riccardo Poli. *Genetic Programming*, pp. 127–164. Springer US, 2005.

[9] William La Cava, Bogdan Burlacu, Marco Virgolin, Michael Kommenda, Patryk Orzechowski, Fabrício Olivetti de França, Ying Jin, and Jason H Moore. Contemporary symbolic regression methods and their relative performance. *Advances in neural information processing systems*, 2021(DB1):1, 2021.

[10] Florian Lalande, Yoshitomo Matsubara, Naoya Chiba, Tatsunori Taniai, Ryo Igarashi, and Yoshitaka Ushiku. A transformer model for symbolic regression towards scientific discovery. In *NeurIPS 2023 AI for Science Workshop*, 2023.

[11] Mikel Landajuela, Chak Shing Lee, Jiachen Yang, Ruben Glatt, Claudio P Santiago, Ignacio Aravena, Terrell Mundhenk, Garrett Mulcahy, and Brenden K Petersen. A unified framework for deep symbolic regression. In *Advances in Neural Information Processing Systems*, 2022.

[12] Pablo Lemos, Niall Jeffrey, Miles Cranmer, Shirley Ho, and Peter Battaglia. Rediscovering orbital mechanics with machine learning. *Machine Learning: Science and Technology*, 4(4): 045002, 2023.

[13] Nour Makke and Sanjay Chawla. Interpretable scientific discovery with symbolic regression: a review. *Artificial Intelligence Review*, 57(1):2, 2024.

[14] Donald W Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[15] Yoshitomo Matsubara, Naoya Chiba, Ryo Igarashi, and Yoshitaka Ushiku. Rethinking symbolic regression datasets and benchmarks for scientific discovery. *Journal of Data-centric Machine Learning Research*, 2024.

[16] Brenden K Petersen, Mikel Landajuela Larma, Terrell N. Mundhenk, Claudio Prata Santiago, Soo Kyung Kim, and Joanne Taery Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*, 2021.

[17] Wassim Tenachi, Rodrigo Ibata, and Foivos I Diakogiannis. Deep symbolic regression for physics guided by units constraints: toward the automated discovery of physical laws. *The Astrophysical Journal*, 959(2):99, 2023.

[18] Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

[19] Harsha Vaddireddy, Adil Rasheed, Anne E Staples, and Omer San. Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensor observation data. *Physics of Fluids*, 32(1), 2020.

[20] Yiqun Wang, Nicholas Wagner, and James M Rondinelli. Symbolic regression in materials science. *MRS Communications*, 9(3):793–805, 2019.

[21] Baicheng Weng, Zhilong Song, Rilong Zhu, Qingyu Yan, Qingde Sun, Corey G Grice, Yanfa Yan, and Wan-Jian Yin. Simple descriptor derived from symbolic regression accelerating the discovery of new perovskite catalysts. *Nature communications*, 11(1):3513, 2020.