Symbolic Regression for Probability Density Functions: Bridging Non-parametric and Parametric Density Estimation

by

Angelo Arvind Rajendram

A research paper
presented to the University of Waterloo
in fulfillment of the
research requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Waterloo, Ontario, Canada, 2025

Author's Declaration

I hereby declare that I am the sole author of this research paper. This is a true copy of the research paper, including any required final revisions, as accepted by my examiners.

I understand that my research paper may be made electronically available to the public.

Abstract

Data-driven modeling approaches, such as deep learning methods, excel at capturing complex relationships but remain opaque, limiting their use in domains that require interpretability. This work introduces PDF-SR, a novel framework for symbolic regression of probability density functions, which recovers closed-form expressions for multivariate probability density functions directly from a dataset of samples drawn from an unknown underlying distribution. PDF-SR proceeds in three stages: (i) techniques to decompose the problem into simpler subproblems, such as clustering and structure learning with probabilistic graphical models; (ii) support estimation and nonparametric density estimation to produce a smooth, bounded black-box model of the distribution; (iii) decomposition into marginal and conditional distributions to bias the search for expressions describing the joint distribution; and (iv) symbolic regression, warm started using results from the lower-dimensional marginal ad conditional expression search to recover parsimonious expressions that describe the joint distribution.

We demonstrate PDF-SR on both synthetic and real world examples, including mixtures of Gaussian distributions, multivariate normal distributions, a muon-decay process from particle physics, and a variant of the Rastrigin function as a toy distribution. In each case, PDF-SR is applied to recover the underlying distributions (or close approximations) with low residual error, while producing a Pareto front of expressions that trade off complexity and fidelity.

PDF-SR is an attempt to bridge non-parametric and parametric density estimation, offering interpretable analytic models of probability densities that could accelerate scientific discovery and offer insight into otherwise inscrutable processes.

Acknowledgements

The pursuit of mathematics is, at its heart, a human endeavor. My work is only possible due to my faith in the Lord Jesus and the support and guidance of those around me.

I would like to first thank my supervisor Professor Vijay Ganesh, who has been a source of illuminating insight into the nature of the computational and mathematical sciences. His guidance has helped me move between the two modes of exploiting what is known and exploring what is unknown to produce fruitful work.

I would like to sincerely thank and acknowledge Xieting Chu, whose skills and insight far belie his years and have been invaluable in advancing this research. Also, sincere thanks to Aishik Ghosh and Max Fieg who inspired the original research direction and offered guidance and mentorship during moments when I was caught in the weeds.

I am thankful to my mentors and managers, namely Andre Bigras, Roger Korus, and Ed Harb, and colleagues at Telesat who have both encouraged and enabled me to pursue my interests alongside working full-time.

I am grateful to my friends Aaron Pereira, Shermon Rodrigues, Rowayne Murzello, Bentic Sebastian, Yamer Yusuf, Anand Brunst, and Mylene Roux who have patiently entertained my interests over extended periods and reminded me to nurture aspects of my life beyond work and academic research. I am also grateful to Sacha Guneratne for the conversations at a park in Ottawa, over the summer of 2021, that led me to pursue this degree.

Finally, I am deeply and eternally grateful to my parents, Donna and Kenneth Rajendram, whose steadfast faith and support, both from near and far, have been the solid foundation that has allowed me to succeed in this pursuit.

Dedication

To my parents, whose faith, love, and wisdom are with me always.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

Dedication v

List of Figures viii

List of Tables xi

1 Introduction 1

2 Background 3

2.1 Preliminaries 3

2.1.1 Symbolic Regression 3

2.1.2 Probability Distributions 4

2.1.3 Density Estimation 6

2.2 Related Work 7

2.2.1 PySR 8

2.2.2 Symbolic Regression using Control Variables 9

2.2.3 AI Feynman 9
2.2.4 Deep Learning-Based SR 10
2.2.5 LaSR 10
2.2.6 MESSY Estimation 11

3 Methods **13**
3.1 PDF-SR Workflow 13
3.1.1 Support Estimation 14
3.1.2 Clustering 15
3.1.3 Structure Learning 16
3.1.4 Nonparametric Density Estimation 17
3.1.5 Symbolic Regression 20

4 Results **22**
4.1 Additive and Multiplicative Decomposition 22
4.2 Scaling Dimensionality 29
4.3 Muon Decay 32
4.4 Rastrigin Toy PDF 36
4.5 Conclusion 40

References **42**

APPENDICES **48**

A Hyper-parameters for PDF-SR Experiments **49**
A.1 Mixture of two 2D Normal Distributions 50
A.2 Dimensionality Scaling Experiments 51
A.3 Rastrigin Toy PDF 53
A.4 Muon Decay 54

List of Figures

2.1 An example of a probabilistic graphical model for the distribution $p(x_1, ..., x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)[(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$ [6]. 6

3.1 The overall PDF-SR framework. 13

3.2 Convex hull type support estimates for a simulated dataset. (Left) True support. (Right) Convex hull estimate [18]. 15

3.3 Clustering with DBSCAN. Each point marked with a triangle represents a core-point from a mixture of 2D Gaussians. The shaded area its ϵ-neighborhood (ϵ is a hyperparameter, also called the *bandwidth*). A core point is a sample point that has at least minPts (another hyperparameter) sample points in its ϵ-neighbourhood. [25]. 16

3.4 A PGM of the 4 dimensional Gaussian obtained using *pgmpy*. 17

3.5 Kernel Density Estimation in 1D[11] 19

3.6 Kernel Density reflection trick on a uniform distribution. The KDE estimate is reflected across boundaries, summed together, truncated, and finally normalized. 19

4.1 Clustering samples from the mixture of two 2D Gaussians (additive composition). 23

4.2 A PGM of the 4 dimensional Gaussian obtained using *pgmpy* (multiplicative composition). 24

4.3 The KDE density estimate as a heat map (left). The 3D plot of the same KDE density estimate (right). 25

4.4 Pareto frontier associated with cluster 1 (simplified expressions shown). Expression that simplify to constants are not shown. 26

4.5 A heat map of the residual error between the estimated symbolic density (predicted by the expression at complexity 46) and the ground truth density (left). The maximum residual error is 0.06. A 3D plot of the recovered probability density function (right). 28

4.6 The Spearman correlation coefficient for PDF-SR with the restricted set of operators as the number of dimensions increases from 2 to 6 at 20 iterations (left), 200 iterations (center), and 2000 iterations (right). 30

4.7 Box plots of the raw complexity scores of expressions returned by PDF-SR with the restricted set of operators for dimensions 2 (lowest row) to 6 (highest row) after 20 iterations (left column), 200 iterations (center column), and 2000 iterations (right). 30

4.8 The Spearman correlation coefficient for PDF-SR with the extended set of operators as the number of dimensions increases from 2 to 6 at 20 iterations (left), 200 iterations (center), and 2000 iterations (right). 31

4.9 Box plots of the raw complexity scores of expressions returned by PDF-SR with the extended set of operators for dimensions 2 (lowest row) to 6 (highest row) after 20 iterations (left column), 200 iterations (center column), and 2000 iterations (right). 31

4.10 KDE estimate (left) and 3D plot associated with the Muon decay process (probability density function). 33

4.11 Convex hull estimate of the support of the probability density function. . . 33

4.12 (Pareto frontier associated with the Muon decay process (simplified expressions shown). 34

4.13 A heat map of the residual error between the estimated symbolic density (predicted by the expression at complexity 27) and the kernel density estimate. The estimated support is enclosed in the red curve (left).A 3D plot of the recovered probability density function (right). 36

4.14 The KDE density estimate as a heat map (left). The 3D plot of the same KDE density estimate (right). 37

4.15 (Pareto frontier associated with the rastrigin probability density function (simplified expressions shown). 38

4.16 A heat map of the residual error between the estimated symbolic density (predicted by the expression at complexity 26) and the ground truth density (left). The peak residual error is 0.004. A 3D plot of the recovered probability density function (right). 40

List of Tables

4.1 Ground truth expressions for the two component 2D Gaussians 23

4.2 Expressions simplified with *sympy* alongside their raw complexity score from PySR. Expressions that simplify to constants are not shown. 27

4.3 New expressions simplified with *sympy* alongside their raw complexity score from PySR. 35

4.4 Variant of Rastrigin function used as a probability density function 37

4.5 Expressions simplified with *sympy* alongside their raw complexity score from PySR. Expressions that simplify to constants are not shown. 39

A.1 Hyper-parameters for symbolic regression on each cluster 50

A.2 Hyper-parameters for the dimensionality study with restricted operators . 51

A.3 Hyper-parameters for the dimensionality study with extended operators . . 52

A.4 Hyper-parameters for symbolic regression on the Rastrigin PDF 53

A.5 Hyper-parameters for symbolic regression on the Muon Decay PDF 54

Chapter 1

Introduction

Data-driven approaches, particularly neural networks, are widely used in modeling complex relationships between predictor and target variables. Despite their success across various domains, their black-box nature often obscures the underlying patterns in the data, limiting interpretability. This lack of transparency restricts their use in contexts where accountability and interpretability are essential. Symbolic Regression (SR) offers a promising alternative – a family of techniques that search the space of mathematical expressions and parameters to produce interpretable models that reveal structure intrinsic in the data. SR techniques yield parsimonious symbolic expressions that generalize effectively, provide indicators of prediction reliability, and demonstrate strong noise tolerance. These qualities make them especially valuable in scientific fields such as physics, where interpretable expressions can drive theoretical advances and enable the discovery of new phenomena from data.

Symbolic regression is an NP-hard problem [52]. Notable work in SR includes AI Feynman [49], which rediscovered known physics equations from the Feynman Lectures, and Symbolic Metamodels [2], which transform black-box models into interpretable "white-box" representations. Our work focuses on a specialized problem: symbolic regression for probability density functions (PDF-SR). Techniques applied to this problem can leverage the common structure shared by all probability distributions. More specifically, the relationships between joint, marginal, and conditional distributions, alongside normalization and non-negativity constraints required by valid distributions.

The objective of PDF-SR is to determine a closed-form mathematical expression of the probability density function $f_{\mathbf{X}}(\mathbf{x})$, where $\mathbf{x}$ is a vector in $\mathbb{R}^d$. The data consist of n independent and identically distributed observations, $\mathcal{S} = \{\mathbf{X}_i\}_{i=1}^{N}$, with each random vector $\mathbf{X}_i \sim f_{\mathbf{X}}$. This problem appears in domains such as physics, where probability distributions may describe particle behavior and interactions. In quantum mechanics, for example, the squared wavefunction represents the probability of detecting a particle at a specific location and time. Similarly, distributions of particle properties, such as momentum and energy, help model high-energy collisions and interpret experimental data.

PDF-SR faces three key challenges: (i) the curse of dimensionality, since high-dimensional settings require exponentially larger datasets for adequate model fitting; (ii) the difficulty of identifying complex symbolic expressions, though many phenomena of interest are expected to obey parsimonious expressions; and (iii) the requirement that generated expressions satisfy normalization and non-negativity constraints over the distribution's domain. These constraints are typically not enforced by symbolic regression techniques.

Our proposed approach attempts to address the first two challenges, while the third is handled by filtering out non-realizable candidate functions at the end of the SR pipeline. Our contributions are summarized as follows:

- We first propose a framework to discover the Probability Distribution Function using a four-step method: additive and multiplicative decomposition, nonparametric density estimation, symbolic regression to determine the expressions of the marginals and conditionals, and finally warm-starting an expression search with the marginal and conditional expressions to obtain the expression of the joint probability density functions.

- We explore various techniques to decompose the original problem into simpler subproblems and provide an empirical study of how the method scales with the number of covariates.

Chapter 2

Background

2.1 Preliminaries

2.1.1 Symbolic Regression

Symbolic Regression (SR) is a type of regression analysis that aims to discover mathematical expressions that describe relationships between variables in a dataset while balancing accuracy and simplicity. Unlike traditional regression methods, which assume a pre-defined functional form (e.g. linear or polynomial regression), symbolic regression explores a vast space of potential mathematical expressions to find the most suitable representation of the target variable in terms of the predictor variables.

More formally, given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, where $\mathbf{x}_i$ represents a vector of predictor variables, and y_i denotes the corresponding target, the goal of symbolic regression is to identify a function f such that,

$$y_i \approx f(\mathbf{x}_i) \quad \forall i \in \{1, 2, .., N\}$$

Syntactically valid candidate functions $\hat{f}$ are typically constructed as expression trees from binary operators (e.g., addition, multiplication), unary operators (e.g. sine, cosine, logarithm), and predictor variables. These expression trees adhere to a context-free grammar (CFG): a formal system used to define the syntax of languages. A CFG consists of (i) a set of non-terminal symbols - abstract variables that denote sub-expression structures (ii) a set of terminal symbols, the operators or predictor variables, (iii) a set of production rules that state how each non-terminal can be replaced by a string of terminals and non-terminals, and (iv) a start symbol that serves as the root or seed for the construction of the expression tree.

Symbolic regression is classically approached with genetic programming (GP) [30], with PySR [17] among the state-of-the-art tools. Genetic algorithms allow for flexible search space exploration with few prior assumptions. The algorithm starts with a random population of expression trees and employ genetic operations such as crossover (swapping sub-trees between individuals) and mutation (random alterations to parts of an expression tree) to generate new candidate expressions, with occasional optimization of the constant parameters appearing in the expressions. Each expression is evaluated using a fitness function that measures how well it models the target data. Over successive generations, fitter functions are selected and combined, gradually evolving toward models that capture the underlying relationships in the data.

Symbolic regression is NP-hard [52] and the number of possible mathematical expressions grows exponentially with the number of allowed operations and predictor variables.

2.1.2 Probability Distributions

Probability distributions describe how probabilities are assigned to different outcomes in a sample space. In general, they may be either discrete (modeled by a probability mass function (PMF)) or continuous (modeled by a probability density function (PDF)). Datasets of samples from some unknown random generative process do not conform to the standard setting for symbolic regression as the target variable – the probability density or probability mass – is not included explicitly with each sample. Instead, the target variable must be estimated using density estimation techniques. The absence of the explicit value of the target variable makes density estimation especially challenging in high-dimensional settings.

A continuous probability distribution deals with a random variable X that can take an infinite number of values within a range. The univariate probability density function $f_X(x)$, satisfies

$$P(a \leq X \leq B) = \int_a^b f_X(x)dx$$

where $f_X(x) \geq 0$ (non-negativity constraint) and $\int_{-\infty}^{\infty} f_X(x)dx = 1$ (unit volume constraint) which are also required in the multivariate case. In addition to this, the relationships between random variables are described using joint, marginal, and conditional distributions for multivariate probability density functions. Formally, for a continuous random vector $\mathbf{X} = (X_1, X_2, ..., X_n))$ the joint PDF is denoted as $f_{\mathbf{X}}(\mathbf{x})$. The proabability of $\mathbf{X}$ falling with a region $R \in \mathbb{R}^n$ is,

$$P(\mathbf{X} \in R) = \int_R f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}$$

For any subset of the variables $\mathbf{X}_S = (X_{i_1}, X_{i_2}, ..., X_{i_k})$ where $S = \{i_1, i_2, ..., i_k\} \subseteq \{1, 2, ..., n\}$, the marginal PDF is obtained by integrating out the remaining variables,

$$f_{\mathbf{X}_S}(\mathbf{x}_S) = \int_{\mathbb{R}^{n-k}} f_{\mathbf{X}}(\mathbf{x}) d_{\mathbf{x}_{\bar{S}}}$$

where $\mathbf{x}_{\bar{S}}$ are the variables not indexed in the subset S. For disjoint subsets A and B of variables, the conditional PDF of $\mathbf{X}_A$ given $\mathbf{X}_B = \mathbf{x}_B$ is defined as,

$$f_{\mathbf{X}_A|\mathbf{X}_B}(\mathbf{x}_A|\mathbf{x}_B) = \frac{f_{\mathbf{X}_{A \bigcup B}}(\mathbf{x}_A, \mathbf{x}_B)}{f_{\mathbf{X}_B}(\mathbf{x}_B)}$$

provided that $f_{\mathbf{X}_B}(\mathbf{x}_B) > 0$.

Probabilistic Graphical Models (PGM) & Structure Learning

Probability theory relies on two simple equations known as the sum rule and the product rule. Most probabilistic manipulations are the result of repeated application of these equations [6]. Any probability distribution can be represented as a directed acyclic graph (DAG) called a probabilistic graphical model (PGM): each node represents a random variable, and the links represent probabilistic relationships (or lack of independence) between these variables. In typical convention, the sense of the edge is from the conditioning variable to the dependent variable. General joint distributions may be factorized into a product of conditionals and marginals using the product rule. For the joint distribution over n variables, we have,

$$p(x_1, ..., x_n) = p(x_n|x_1, ..., x_{n-1})...p(x_2|x_1)p(x_1)$$

The PGM representing this distribution is said to be fully connected since there is a link between every node (or random variable) in the graph. A more general PGM where certain variables are independent of others is shown in Figure 2.1. Methods that uncover the PGM from a dataset are known as structure learning.

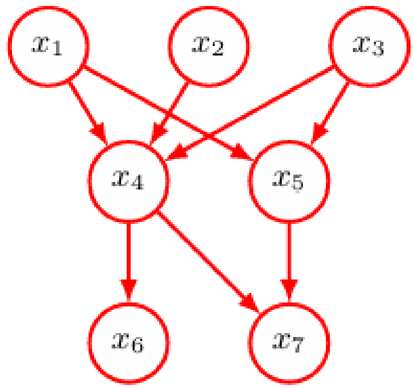

Figure 2.1: An example of a probabilistic graphical model for the distribution $p(x_1, ..., x_7) = p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)[(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$ [6].

A python toolkit for probabilistic graphical models and structure learning is *pgmpy* [4]. For structure learning on continuous data, pygmpy uses the PC algorithm: a constraint-based algorithm that utilizes conditional independence tests to construct the model.

2.1.3 Density Estimation

Density estimation is the process of approximating the probability distribution of a dataset. Given a set of independent and identically distributed (i.i.d.) samples $X_1, X_2, ..., X_n$ drawn from an unknown distribution with probability density function $f(x)$, the goal is to estimate $f(x)$ using the observed data. In general, this is an ill-posed problem, since there exist multiple distributions that fit a given dataset of samples. There are two main types of density estimation methods, parametric density estimation, and non-parametric density estimation.

Parametric Density Estimation

Parametric density estimation assumes that data follow a specific distribution (e.g. Gaussian, Exponential, or Mixture Models) with unknown parameters. The parameters are estimated using methods like Maximum Likelihood Estimation (MLE) or Bayesian Inference. As the functional form is assumed a priori, parametric density estimation is data-efficient, requiring fewer samples for accurate estimation than nonparametric density estimation. However, these approaches may suffer from bias when the underlying distribution is misspecified.

Non-parametric Density Estimation

Non-parametric density estimation makes fewer assumptions about the functional form of the underlying distribution and infers the density directly from the data. Kernel Density Estimation (KDE) is a widely used nonparametric density estimation technique. Other techniques may also be considered based on trade-offs between computational efficiency, smoothness of the density estimate, and robustness to data sparsity (curse of dimensionality).

The computational complexity of naive KDE is $\mathcal{O}(nm)$ where n is the number of samples and m is the number of evaluation points. This complexity may be reduced by KDE variants such as FFTKDE [54] which make use of a binning routine and convolutions of the kernels using the Fast Fourier Transform (FFT), resulting in an overall runtime of $\mathcal{O}(N2^d + n \log n)$, where d is the number of dimensions. Additionally, selecting the appropriate bandwidth hyperparameter can be challenging for high dimensions. Adaptive KDE can use varying bandwidths for each data point improving local feature detection, though this increases variance in low-density regions and the number of hyperparameters.

Alternative techniques for non-parametric density estimation include the less sophisticated Histogram density estimation and Nearest Neighbour (k-NN) density estimation. Tree-based density estimation techniques partition the input space into hyper-rectangles and can be sparse, mitigating issues with the curse of dimensionality [3], though their density estimates may not be as smooth as KDE. Projection pursuit density estimation (PPDE) projects high-dimensional data onto "interesting" low-dimensional subspaces, potentially bypassing the curse of dimensionality [20]. However, finding optimal projections may be computationally expensive. Finally, recent advances in deep learning have seen the development of several novel neural density estimation techniques using auto-regressive models [50], normalizing flows [19], and generative models such as GANs [33].

2.2 Related Work

In this section, we review notable works relevant to symbolic regression and the less studied problem of symbolic expression recovery for probability density functions. Several of the methods discussed here are applicable to the standard symbolic regression problem, with the exception of MESSY density estimation, which is another work that directly addresses the problem of recovering symbolic expressions of probability density functions.

2.2.1 PySR

PySR [17] is an open-source Python library for practical symbolic regression. It is built with a high-performance Julia backend for efficient computation, and its core algorithm uses a multi-population evolutionary algorithm featuring an evolve-simplify-optimize loop designed to effectively search for symbolic expressions.

PySR offers several features tailored for scientific equation discovery such as allowing custom mathematical operators and loss functions for specific domains, handling of noisy data through denoising pre-processing and data weights, providing adaptive parsimony metrics to balance accuracy and complexity, permitting the specification of constraints on the form and size of expressions, as well as providing a scikit-learn-like Python interface. When evaluated on EmpiricalBench [17], PySR has shown strong performance, sometimes outperforming deep learning-based symbolic regression methods. It has been successfully applied in various scientific fields including astrophysics, climate science, materials science, economics, and gravitational wave research. This open source tool helps researchers uncover mathematical laws that govern their data.

PySR represents symbolic expressions as expression trees, and evolves several separate populations through successive generations to explore the search space. Candidate expressions asynchronously migrate between these isolated population enabling them to share information while allowing massive parallelization. Like other evolutionary algorithms, PySR uses mutation and crossover operations. The mutation step may change constants, add nodes to the expression tree, or delete or simplify or intialize subtrees. Crossover operations involve swapping random subtrees between expressions. PySR by default defines the complexity of the expression to be the number of nodes in an expression tree, however, the definition of complexity itself is configurable by the user.

To support the selection of appropriate models, PySR provides a pareto front of the discovered expressions, scoring each in terms of its predictive accuracy against the dataset and the expression complexity, offering researchers a set of models that represent trade-offs between these two criteria.

2.2.2 Symbolic Regression using Control Variables

Symbolic Regression using Control Variables ([27] and [15]) addresses the exponential growth of the search space with the number of variables by leveraging control variables to enhance accuracy and scalability. The multi-variable symbolic regression is decomposed into a set of single-variable SR problems which are subsequently combined.

The general procedure is begins by learning a surrogate model to approximate the observed data. This surrogate model is used to generate data by holding a subset of the variables fixed (controlled variables). Univariate symbolic regression is used to determine the mathematical expression based on the generated data, avoiding expontential blow-up of the search space. Once an expression for the independent variable is obtained, another variable is selected as the independent variable and the procedure is repeated to incorporate the additional variable into the symbolic expression. This is repeated until there are no control variables left.

These methods directly inspire the technique used in this work. Rather than simply decomposing the original function into univariate sub-problems, we decompose the problem of discovering a high-dimensional pdf into its marginals and conditional distribution before attempting to recombine them and discover the full joint probability density function.

2.2.3 AI Feynman

AI Feynman [49] is another symbolic regression algorithm that integrates neural networks with physics-inspired techniques to discover closed-form mathematical expressions from data. AI Feynman systematically decomposes complex equations using symmetry detection, separability analysis, and modular transformations to exploit hidden structure in the data while employing brute-force search and polynomial fitting as base case solvers. The algorithm makes use of feed-forward neural networks to approximate and interpolate the unknown function and subsequently test for symmetry and separability of the independent variables. AI Feynman demonstrated superior performance in recovering physics equations, solving all 100 benchmark problems in the Feynman Symbolic Regression Database, whereas traditional methods like Eureqa solved only 71.

Building on this foundation, AI Feynman 2.0 [48] introduced several key improvements to enhance robustness and extend its capabilities. The updated version incorporates Pareto-optimal symbolic regression, which balances equation accuracy against complexity, discarding overly intricate solutions that do not significantly improve predictive power. AI

Feynman 2.0 also generalizes its modular decomposition approach, moving beyond predefined symmetry types to discover arbitrary modularity in a function's computational graph by analyzing neural network gradients. This enables it to recognize more complex functional dependencies that the original version could not detect.

Additionally, AI Feynman 2.0 integrates normalizing flows, allowing it to perform symbolic regression on probability distributions using only sample data, making it applicable to density estimation problems. To improve robustness against noise and data inconsistencies, the algorithm employs statistical hypothesis testing to refine its brute-force search process, ensuring that only reliable candidate solutions are retained. These advancements make AI Feynman 2.0 significantly more effective at handling high-dimensional, noisy, or structured datasets, extending its utility in physics, machine learning, and scientific discovery.

2.2.4 Deep Learning-Based SR

With advances in deep learning, neural sequence models have been applied to the problem of symbolic regression. Deep Symbolic Regression (DSR) [39] applies reinforcement learning for symbolic regression, while Neural-Guided Genetic Programming (NGGP) [36] is a hybrid approach that combines reinforcement learning with genetic programming. Symbolic Physics Learner (SPL) [46] formulates symbolic regression as Monte Carlo tree search.

Transformer-based approaches have also shown strong performance in SR tasks. Neural Symbolic Regression that Scales (NeSymReS) [5] utilizes transformers to identify expressions from input data. SymbolicGPT [51] employs a T-Net to first embed input data, then uses a GPT model to generate expressions, after which it optimizes the expression constants. End-to-End Symbolic Regression (E2ESR) [28] eliminates the two-stage process by directly predicting complete symbolic expressions, including constants, using a transformer-based architecture. Transformer-Based Planning for Symbolic Regression Planning [45] integrates transformers with Monte Carlo tree search to improve expression discovery.

2.2.5 LaSR

LASR [23] is a novel symbolic regression (SR) method best viewed as an extension to PySR. It enhances the genetic programming component of symbolic regression by integrating a learned concept library guided by a large language model (LLM). LASR uses zero-shot queries to the LLM to discover and evolve abstract concepts occurring in high-performing candidate expressions allowing the method to leverage empirical patterns and intuitions derived from previously observed data. This places the technique closer to the process of human scientific discovery than traditional SR approaches which typically do not assume any prior knowledge. LASR consists of three primary innovations:

- Concept-Guided Hypothesis Evolution: LASR applies standard genetic operations (mutation, crossover, and selection) but interleaves them with LLM-guided transformations, where new expressions are synthesized based on abstractions recorded as textual context in a concept library. This guides the LLM to initialize candidate expressions, recombine them, or mutate them based on context provided via the concept library. A hyperparameter controls how frequently the LLM-based mutation, and crossover operations substitute the standard genetic operations guiding the candidate expressions towards the hypotheses while still permitting local exploration of the search space.

- LLM-Based Concept Abstraction: The concept library is updated during symbolic regression. The highly scoring candidate expressions are contrasted against the poorly scoring ones and provided to the LLM in an attempt to glean abstract concepts in the form of textual context that may inform the search over successive generations of candidate expressions.

- Concept Evolution Mechanism: The learned concepts are themselves subject to concept evolution to include new ideas that logically follow from existing ones. Concepts are also refined to more succinct and general forms.

These three steps form an open-ended alternating optimization loop that combines evolutionary exploration with an LLM's domain-specific knowledge and in-context learning ability. LASR was shown to outperform several competing approaches on the standard Feynman equation task and was used to discover a novel scaling law for LLMs.

2.2.6 MESSY Estimation

Most of the works surveyed here address the standard symbolic regression problem of identifying analytical expressions that describe relationships between predictor variables and target variables. Maximum-Entropy based Stochastic and Symbolic density Estimation (MESSY) [47] is among the few works that directly tackle symbolic regression of probability density functions.

The MESSY approach combines several key ideas. It uses a Gradient-based drift-diffusion process that connects samples to a "guess" symbolic expression for the unknown density function. It leverages the principle of maximum entropy to formulate the guess distribution, motivated by the Maximum Entropy Distribution (MED) being the least biased density estimate given a number of moments of an unknown distribution [29]. It introduces a method to efficiently compute the parameters of the maximum entropy distribution by solving a linear system of equations, avoiding complex optimization. Finally, it employs symbolic regression alongside an acceptance/rejection procedure to find optimal and well-conditioned basis functions for the exponent of the MED. While using the basis functions only in the exponent ensures that the non-negativity and unit area constraints of a valid probability distribution are satisfied, this assumption restricts the family of expressions that can be discovered by this technique.

MESSY supports a multi-level process that improves predictions as the distribution becomes more detailed. By successively finding an MED estimate and removing the samples covered by each estimate, MESSY is able to estimate multi-modal distributions. Additionally, MESSY is able to improve predictions by incorporating known constraints about the support of boundaries of the unknown distribution. This technique can also be used for high-dimensional distributions, however, more empirical evidence is needed to determine the trade-offs between accuracy and cost as the number of dimensions are scaled.

MESSY's computational cost grows linearly with the number of samples for each set of selected basis functions and quadratically with the number of basis functions. This technique outperforms several other existing density estimation techniques for a small to moderate number of samples by providing a low-bias symbolic description of the unknown density.

Chapter 3

Methods

3.1 PDF-SR Workflow

The proposed algorithm is illustrated in Figure 3.1. To discover the underlying probability density function, we first employ non-parameteric density estimation techniques to obtain a black-box model of the PDF, followed by symbolic regression to "whiten" the model and obtain candidate symbolic expressions for the underlying probability density function.

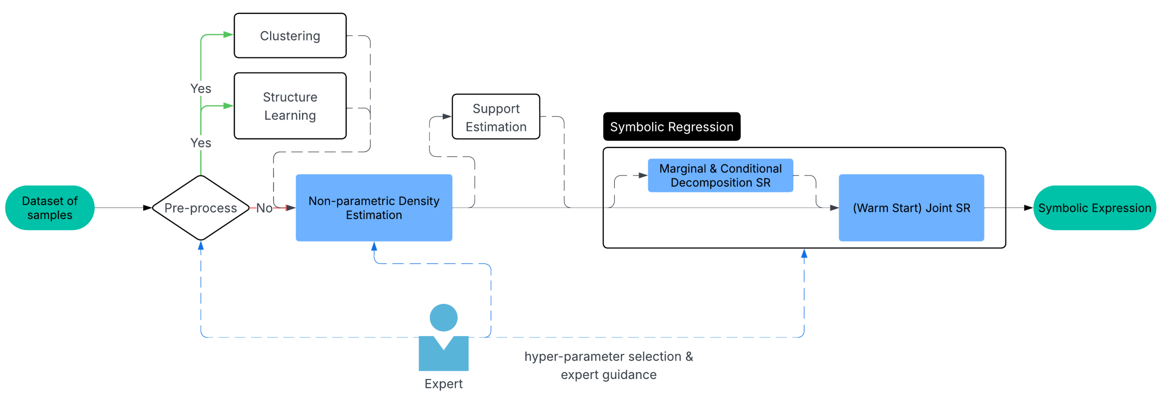

Figure 3.1: The overall PDF-SR framework.

Symbolic regression for probability density functions can be framed as a sequence of sub-problems: (i) Support Estimation, (ii) Clustering, (iii) Structure learning, (iv) Non-parametric density estimation, and (v) Symbolic Regression.

3.1.1 Support Estimation

Support estimation is a critical preliminary step in the PDF-SR pipeline. The support of a probability distribution refers to the subset of the input space $\mathcal{X} \subseteq \mathbb{R}^d$ where the density function $f_{\mathbf{X}}(\mathbf{x})$ is strictly positive.

$$\mathrm{supp}(f_{\mathbf{X}}) = \{\mathbf{x} \in \mathbb{R}^d | f_{\mathbf{X}}(\mathbf{x}) > 0\}$$

Accurately identifying the support of bounded or constrained distributions is crucial to improve the performance and interpretability of the symbolic regression stage of PDF-SR. Relying solely on non-parametric methods for density estimation may overspill the support of the distribution near hard boundaries. With KDE, this occurs as a consequence of kernel smoothing, where the influence of sample points may extend beyond the boundary of the density function, biasing expressions returned by symbolic regression. This may result in expressions that attempt to fit sharp discontinuities or flat tails that do not provide insight into the true mathematical structure of the distribution.

Estimating the support allows for the domain of symbolic regression to be restricted so that it can learn meaningful patterns in the interior of the distribution free from the influence of spurious artifacts at the boundary. For instances where the true support of the distribution is in fact unbounded (e.g. for a normal distribution) treating the problem as having bounded support does not pose an issue since the returned expressions may still extrapolate beyond the estimated support.

A simple technique to estimate the support is level-set thresholding on the KDE density estimate $\hat{f}_n(x)$. The approximate support set is defined as,

$$\widehat{\mathcal{S}}_\tau = \{\mathbf{x} \in \mathbb{R}^d | \hat{f}_n(\mathbf{x}) \geq \tau\}$$

where $\tau > 0$ is a small threshold, chosen heuristically or with user-guidance. This identifies regions where the estimated density is non-negligible and the approximate support can be used to restrict the domain for symbolic regression [9].

If the support is known to be convex, the convex hull, i.e. the smallest convex set containing all samples can be used to estimate the support. The advantage of this technique over level-set thresholding is a better estimation near sharp discontinuities in the density function. A small offset from the convex hull may be used to ensure that sharp drop-offs are excluded from the estimated support. This method suffers when the real distribution is multi-modal or concave since it will include irrelevant empty regions in the domain. A significant limitation for this technique is that it does not scale well to higher dimensions.

More advanced density support estimation techniques include regularized kernel algorithms [42] and support vector machines [40], however, for the examples in this paper we use only level-set thresholding and convex hull support estimation.

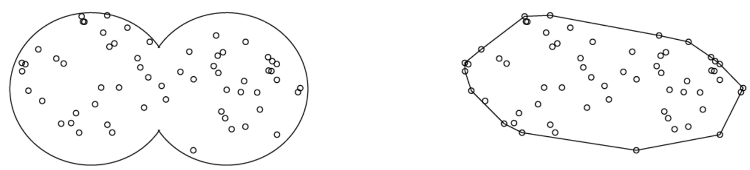

Figure 3.2: Convex hull type support estimates for a simulated dataset. (Left) True support. (Right) Convex hull estimate [18].

3.1.2 Clustering

When dealing with multi-modal data where the modes are well separated in the domain, it can be useful to segment the data into unimodal clusters before symbolic regression. This reduces the expression complexity that would need to be recovered by symbolic regression on each of the clusters as opposed to that which would need to be discovered by a single symbolic regression call on the data from all clusters. Individual component distributions can be combined additively to obtain the symbolic form of the joint distribution. The clustering technique used must (i) automatically infer the number of clusters from the data, (ii) make no parametric assumptions on the cluster shape, and (iii) assign each sample to a cluster (or mark it as noise).

Several variants of model clustering are based on the mean-shift algorithm [21] [12]. Each data point iteratively moves in input space following the gradient of a non-parametric density estimate (such as KDE), converging onto a nearby density mode. A cluster is defined as the basin of attraction of points that land on the same mode. These techniques meet the requirements listed above; however, they are sensitive to the selected KDE bandwidth.

The algorithm used for examples in this work is DBSCAN [55], a density-based clustering method that identifies clusters as contiguous regions of high point density separated by regions of low density. DBSCAN is nonparametric and does not require the number of clusters to be specified in advance. However, care must be taken to ensure that the detected clusters are well separated in the input space, as modes of the distribution may arise from overlapping component distributions and may not be informative as a separate cluster for symbolic regression.

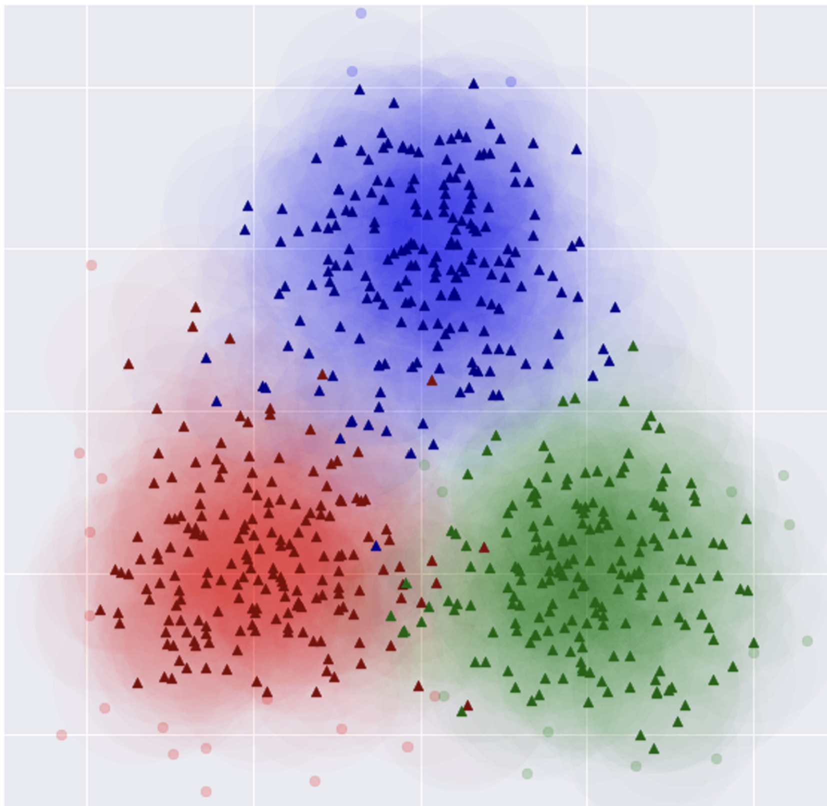

Figure 3.3: Clustering with DBSCAN. Each point marked with a triangle represents a core-point from a mixture of 2D Gaussians. The shaded area its ϵ-neighborhood (ϵ is a hyperparameter, also called the *bandwidth*). A core point is a sample point that has at least minPts (another hyperparameter) sample points in its ϵ-neighbourhood. [25].

3.1.3 Structure Learning

If the structure of the PGM can be learned and we detect independent subsets of the variables, the problem can be decomposed into several smaller subproblems and subsequently combined with the product rule in accordance with the definition of independence.

For example, consider a 4-dimensional joint distribution $p(x_1, x_2, x_3, x_4)$, if two subsets of the variables are independent (e.g. $\{x_1, x_3\}$ and $\{x_2, x_4\}$) are independent, then the joint distribution may be factorized as,

$$p(x_1, x_2, x_3, x_4) = p(x_1, x_4)p(x_3, x_2)$$

We see that the original 4-dimensional density estimation problem has been reduced to the product of the density estimates of two 2-dimensional sub-problems, making computation significantly more efficient. Consequently, identifying connected components in the PGM can significantly reduce the computational requirements for discovering high dimensional probability density functions.

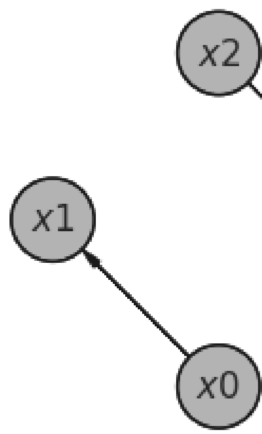

Figure 3.4: A PGM of the 4 dimensional Gaussian obtained using *pgmpy*.

Discovering a PGM is an NP-hard combinatorial problem [13]. Traditional score-based and constraint-based methods for this task leverage approximate search and additional structural assumptions to make the problem tractable in higher-dimensional settings [37]. More recent advances reformulate the task as a continuous optimization problem [59], permitting the use of mature continuous optimization solvers, and also leverage deep learning and generative models to learn the underlying DAG [58].

3.1.4 Nonparametric Density Estimation

Nonparametric density estimation produces a black-box model of the unknown distribution and casts the problem into standard form for symbolic regression, since we have a target variable (the density) for each sample. The accuracy of the estimate is crucial; noisy estimates may bias the resulting symbolic expressions to be overly complex as it attempts to fit spurious artifacts in the black-box model.

With Kernel Density Estimation (KDE), each data point contributes a localized "bump" – a kernel function centered at that point – and the overall estimate is obtained by summing these contributions as illustrated in Figure 3.5. The Kernel Function $K : \mathbb{R}^d \to \mathbb{R}$ quantifies the influence of a data point at a given location. One of the most commonly used kernels is the Gaussian kernel

$$K(x) = \frac{\exp(-\|x\|^2/2)}{\int \exp(-\|x\|^2/2)dx}$$

and with the definition of the Kernel function, the Kernel Density Estimation of a set of independent, identically distributed random samples $\{X_i\}_{i=1}^n$ from an unknown distribution D with density function f, is given by

$$\hat{f}_n(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right)$$

where h is the smoothing bandwidth, a hyperparameter that determines the smoothness of the estimated distribution.

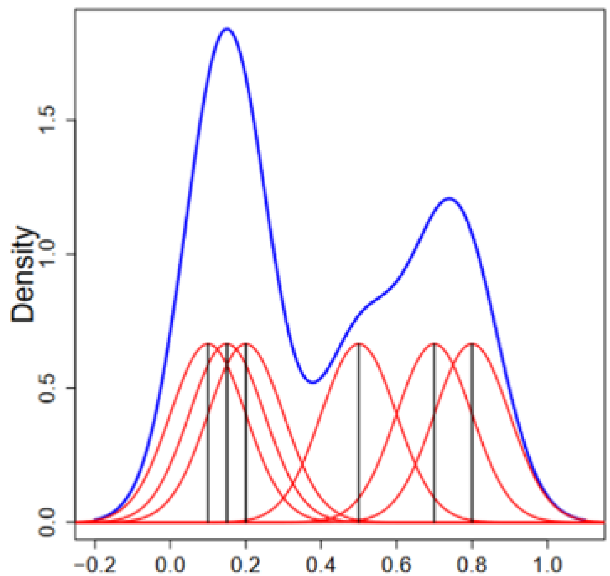

Figure 3.5: Kernel Density Estimation in 1D[11]

The KDE density estimate can be used to inform support estimation via level-set thresholding. Once the density boundary is estimated, the reflection trick augments samples by reflecting them across the boundary to improve the KDE density estimate by reducing bias at the boundary as illustrated in Figure 3.6.

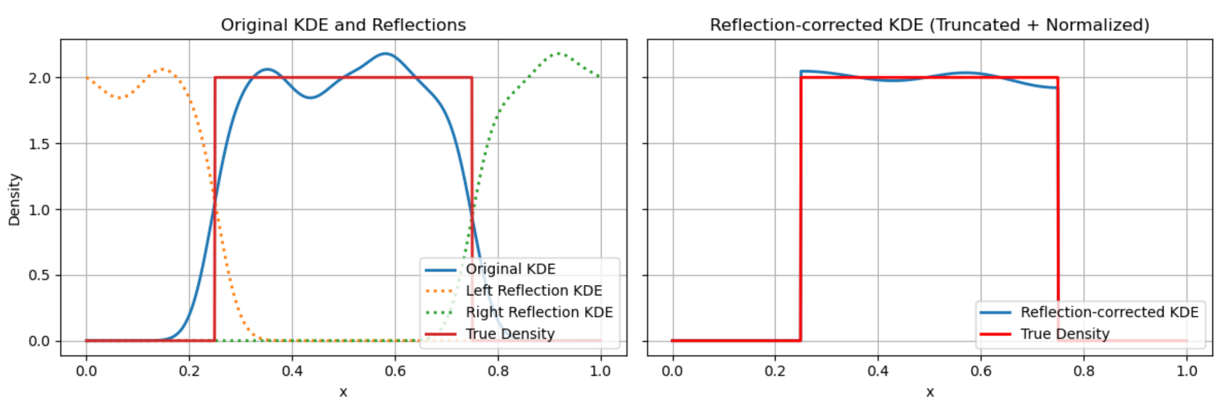

Figure 3.6: Kernel Density reflection trick on a uniform distribution. The KDE estimate is reflected across boundaries, summed together, truncated, and finally normalized.

3.1.5 Symbolic Regression

Since symbolic regression is NP-hard [52], exploiting problem structure can reduce the search space and be crucial to solving the problem. The structure of a joint distribution can be decomposed into a product of marginals and conditionals and may allow high-dimensional density estimation tasks to be decomposed into simpler univariate symbolic regression problems.

Marginals and Conditional SR

Let $f_{X_1,X_2,...,X_d}(x_1, x_2, ..., x_d)$ denote the joint distribution of d-continuous variables. From the product rule in probability, we know this distribution can be factorized as,

$$f_{X_1,X_2,...,X_d}(x_1, x_2, ..., x_d) = f_{X_1}(x_1) \prod_{i=2}^{d} f_{X_i|X_{1:i-1}}(x_i|x_1, ..., x_{i-1})$$

The marginals are univariate functions and the conditionals can be considered univariate if the conditioning variables are fixed. By considering several conditional "slices" in each dimension, we may obtain useful information about the expression describing the conditional distribution across the domain. Each slice informs the overall structure of the target distribution similarly to symbolic regression using control variables [15].

Warm-Start Joint SR

The per-slice conditional distribution symbolic expressions are multiplied with the marginal distributions in the remaining variables to produce a candidate expression to warm-start the joint expression search. This is akin to an independence assumption in the remaining variables,

$$\hat{f}_{X_1,X_2,...,X_d}(x_1, x_2, ..., x_d) = \hat{g}_{X_i|X_S}(x_i|x_S) \prod_{j \in S} \hat{g}_{X_j}(x_j)$$

where $\hat{f}$ is the candidate expression, and $\hat{g}$ are univariate expressions of the marginals and conditionals i indexes the variable over which the conditional expression is defined, and S refers to the set of remaining variables over which the marginal expressions are defined. During the symbolic regression search for the joint distribution, the crossover and mutation operations recombine marginals and various conditional expressions from different slices and across different independent variables in an attempt to reconstruct the joint distribution. This warm-start biases the search towards expressions consistent with the marginals and conditionals obtained in the previous stage.

Chapter 4

Results

We evaluate PDF-SR to demonstrate its merits and weaknesses on various examples. For the examples that follow, PDF-SR fits symbolic expressions to nonnormalized probability density estimates, always scaled to have a peak density value of 10. This permits consistent comparison across examples, regardless of the underlying distribution's scale or complexity. It also facilitates optimization of the symbolic regression objective by avoiding very small or large numerical targets, which can introduce instability in the search and evaluation of symbolic expressions.

Suggested default settings from the *PySR* package were used for the PDF-SR pipeline, and the number of iterations were selected to allow the method a reasonable amount of time to converge on the solution (an 18 hour time budget on 2nd and 3rd generation AMD EPYC CPU cores).

4.1 Additive and Multiplicative Decomposition

Clustering (section 3.1.2) and structure learning section 3.1.3) are two techniques to decompose the problem into simpler sub-problems. Clustering assumes that the problem can be decomposed additively thereby reducing the complexity of the expressions during the expression search, while structure learning assumes that the problem can be decomposed multiplicatively, reducing both the complexity of expressions as well as the dimensionality of the component distributions during symbolic regression.

As a toy problem for clustering with random vectors $\mathbf{X} \in \mathbb{R}^2$ we consider a mixture of two 2D normal distributions,

$$f_{\mathbf{X}}(\mathbf{x}) = \frac{1}{2}\mathcal{N}(\mathbf{x} \mid \mu_{\mathbf{1}}, \mathbf{\Sigma}) + \frac{1}{2}\mathcal{N}(\mathbf{x} \mid \mu_{\mathbf{2}}, \mathbf{\Sigma})$$

$$\mu_{\mathbf{1}} = \begin{pmatrix} -4.0 \\ 4.0 \end{pmatrix}, \quad \mu_{\mathbf{2}} = \begin{pmatrix} 4.0 \\ -4.0 \end{pmatrix}, \quad \mathbf{\Sigma} = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}$$

This joint distribution may be clustered into two component 2D normal distributions as shown in Figure 4.1. The ground truth expressions for each component distribution are listed in Table 4.1.

Table 4.1: Ground truth expressions for the two component 2D Gaussians

Cluster	**Ground Truth Expression**
Cluster 1	$f(x_1, x_2) = \frac{5}{6\pi} \exp\left(-\frac{1}{2}\left(\frac{25}{9}x_1^2 + \frac{25}{9}x_2^2 - \frac{40}{9}x_1x_2 + 40x_1 - 40x_2 + 160\right)\right)$
Cluster 2	$f(x_1, x_2) = \frac{5}{6\pi} \exp\left(-\frac{1}{2}\left(\frac{25}{9}x_1^2 + \frac{25}{9}x_2^2 - \frac{40}{9}x_1x_2 - 40x_1 + 40x_2 + 160\right)\right)$

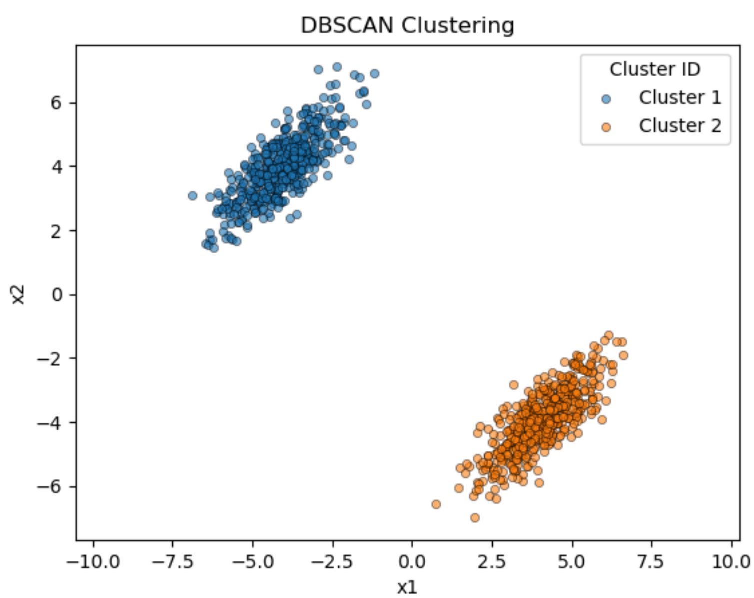

Figure 4.1: Clustering samples from the mixture of two 2D Gaussians (additive composition).

Alternatively, we demonstrate structure learning by considering independent random vectors $\mathbf{X_1}, \mathbf{X_2} \in \mathbb{R}^2$, and their concatenation $\mathbf{X} \in \mathbb{R}^4$. The joint distribution is given by,

$$f_{\mathbf{X}}(\mathbf{x}) = \mathcal{N}(\mathbf{x_1} \mid \mu_\mathbf{1}, \mathbf{\Sigma}) \cdot \mathcal{N}(\mathbf{x_2} \mid \mu_\mathbf{2}, \mathbf{\Sigma})$$

This is a 4D normal distribution with two connected components in its probabilistic graphical model as shown in Figure 4.2. For economy, the parameters of the component distributions in both the mixture model and PGM are the same so that the results can be consolidated for both toy problems.

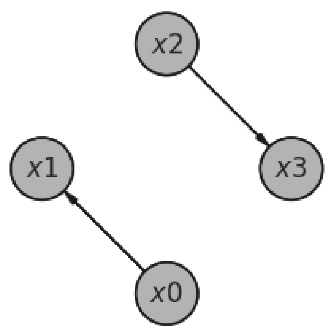

Figure 4.2: A PGM of the 4 dimensional Gaussian obtained using *pgmpy* (multiplicative composition).

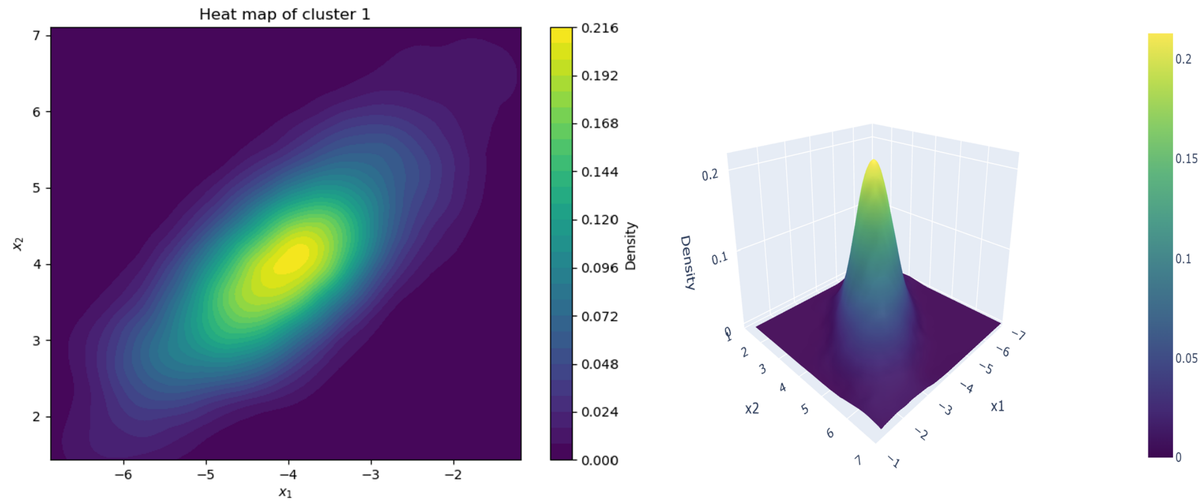

Figure 4.3: The KDE density estimate as a heat map (left). The 3D plot of the same KDE density estimate (right).

The KDE density estimate of cluster 1 is shown in Figure 4.3. The pareto front of expressions for cluster 1 recovered by PDF-SR are shown in Figure 4.4 and Table 4.5. Results for cluster 2 are similar to cluster 1 and are not shown since the functional form of cluster 2 is identical to cluster 1 except for the mean.

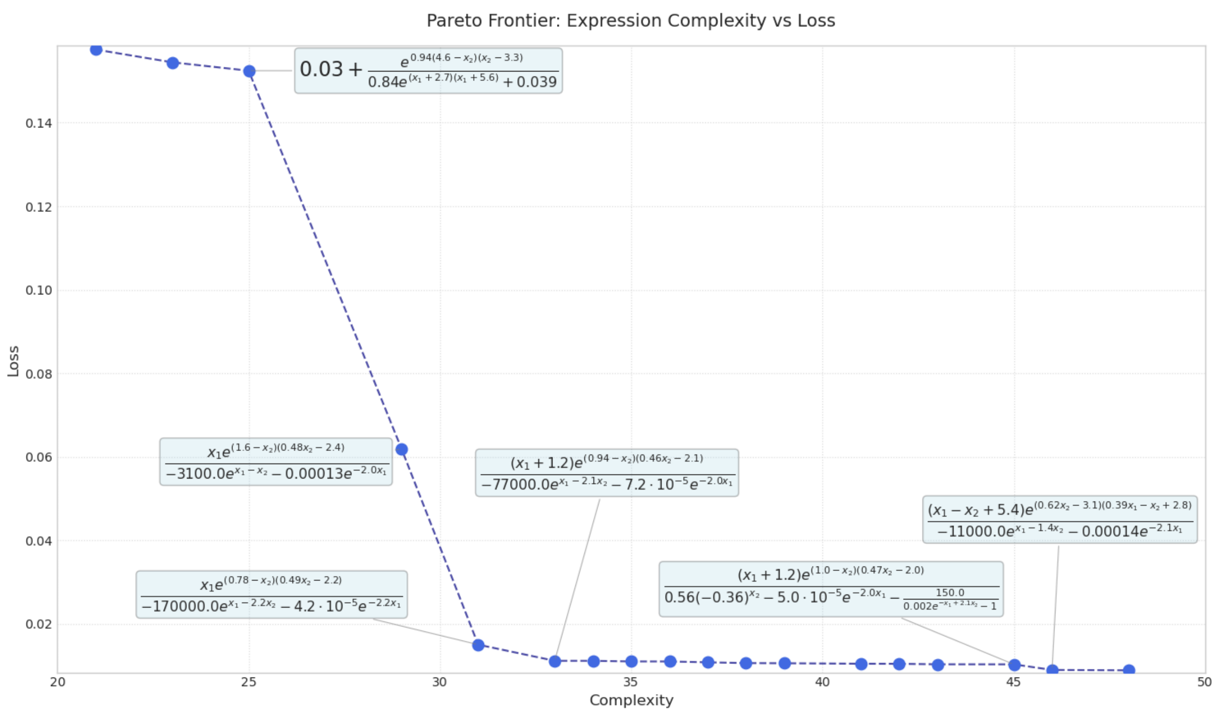

Figure 4.4: Pareto frontier associated with cluster 1 (simplified expressions shown). Expression that simplify to constants are not shown.

Table 4.2: Expressions simplified with *sympy* alongside their raw complexity score from PySR. Expressions that simplify to constants are not shown.

Raw Complexity	Simplified Expression
21	$\frac{e^{-(x_2-4.6)(x_2-3.3)}}{0.84e^{(x_1+2.7)(x_1+5.6)}+0.039}$
23	$\frac{\left(0.03e^{(x_2-4.6)(x_2-3.3)}+1\right)e^{-(x_2-4.6)(x_2-3.3)}}{0.84e^{(x_1+2.7)(x_1+5.6)}+0.039}$
25	$0.03+\frac{e^{-0.94(x_2-4.6)(x_2-3.3)}}{0.84e^{(x_1+2.7)(x_1+5.6)}+0.039}$
29	$-\frac{x_1e^{2.0x_1-(0.48x_2-2.4)(x_2-1.6)}}{3100.0e^{3.0x_1-x_2}+0.00013}$
31	$-\frac{x_1e^{2.2x_1-(0.49x_2-2.2)(x_2-0.78)}}{170000.0e^{3.2x_1-2.2x_2}+4.2\cdot10^{-5}}$
33	$\frac{(-x_1-1.2)e^{2.0x_1-(0.46x_2-2.1)(x_2-0.94)}}{77000.0e^{3.0x_1-2.1x_2}+7.2\cdot10^{-5}}$
34	$\frac{(-x_1-1.2)e^{2.0x_1-(0.46x_2-2.1)(x_2-0.94)}}{77000.0e^{3.0x_1-2.1x_2}+7.2\cdot10^{-5}}$
35	$\frac{(-x_1-1.2)e^{2.0x_1-(0.47x_2-2.0)(x_2-1.1)}}{73000.0e^{3.0x_1-2.1x_2}+5.1\cdot10^{-5}}$
36	$\frac{\left((-x_1-1.2)e^{2.0x_1}-480.0e^{3.0x_1-2.1x_2}-4.5\cdot10^{-7}\right)e^{-(0.46x_2-2.1)(x_2-0.94)}}{77000.0e^{3.0x_1-2.1x_2}+7.2\cdot10^{-5}}$
37	$\frac{(x_1+1.2)\left(1.0e^{x_1-2.1x_2}-0.002\right)e^{2.0x_1-(0.47x_2-2.0)(x_2-1.0)}}{-5.1\cdot10^{-5}e^{x_1-2.1x_2}+150.0e^{3.0x_1-2.1x_2}+1.0\cdot10^{-7}}$
38	$\frac{(-x_1-1.2)e^{2.0x_1-(0.46x_2-2.1)(x_2-0.94)}}{\left(-(-0.31)^{x_2}+77000.0e^{x_1-2.1x_2}\right)e^{2.0x_1}+7.2\cdot10^{-5}}$
39	$-\left(\frac{x_1+1.1}{5.1\cdot10^{-5}e^{-2.0x_1}+\frac{150.0}{0.002e^{-x_1+2.1x_2}-1.1}}+0.0068\right)e^{-(0.47x_2-2.0)(x_2-1.0)}$
41	$-\left(\frac{x_1+1.1}{5.1\cdot10^{-5}e^{-2.0x_1}+\frac{150.0}{0.002e^{-x_1+2.1x_2}-1.0}}+0.014\right)e^{-(0.47x_2-2.0)(x_2-1.0)}$
42	$-\left(\frac{x_1+1.1}{5.1\cdot10^{-5}e^{-2.0x_1}+\frac{150.0}{0.002e^{-x_1+2.1x_2}-1.3}}+0.014\right)e^{-(0.47x_2-2.0)(x_2-1.0)}$
43	$-\frac{(x_1+1.2)e^{-(0.47x_2-2.0)(x_2-1.0)}}{-0.56(-0.36)^{x_2}+5.0\cdot10^{-5}e^{-2.0x_1}+\frac{150.0}{0.002e^{-x_1+2.1x_2}-1.2}}$
45	$\frac{(x_1+1.2)\left(e^{x_1-2.1x_2}-0.002\right)e^{2.0x_1-(0.47x_2-2.0)(x_2-1.0)}}{0.56(-0.36)^{x_2}\left(e^{x_1-2.1x_2}-0.002\right)e^{2.0x_1}-5.0\cdot10^{-5}e^{x_1-2.1x_2}+150.0e^{3.0x_1-2.1x_2}+1.0\cdot10^{-7}}$
46	$\frac{(-x_1+x_2-5.4)e^{2.1x_1+(0.62x_2-3.1)(0.39x_1-x_2+2.8)}}{11000.0e^{3.1x_1-1.4x_2}+0.00014}$
48	$\frac{(-x_1+x_2-5.4)e^{2.1x_1+(0.62x_2-3.1)(0.39x_1-x_2+2.8)}}{\left(11000.0e^{x_1-1.4x_2}+0.0077\right)e^{2.1x_1}+0.00014}$

A plot of the recovered density and the residual error between the densities predicted by the recovered symbolic expression and the ground truth are shown in Figure 4.5. We observe the residuals to be fairly low though still significantly higher than machine precision.

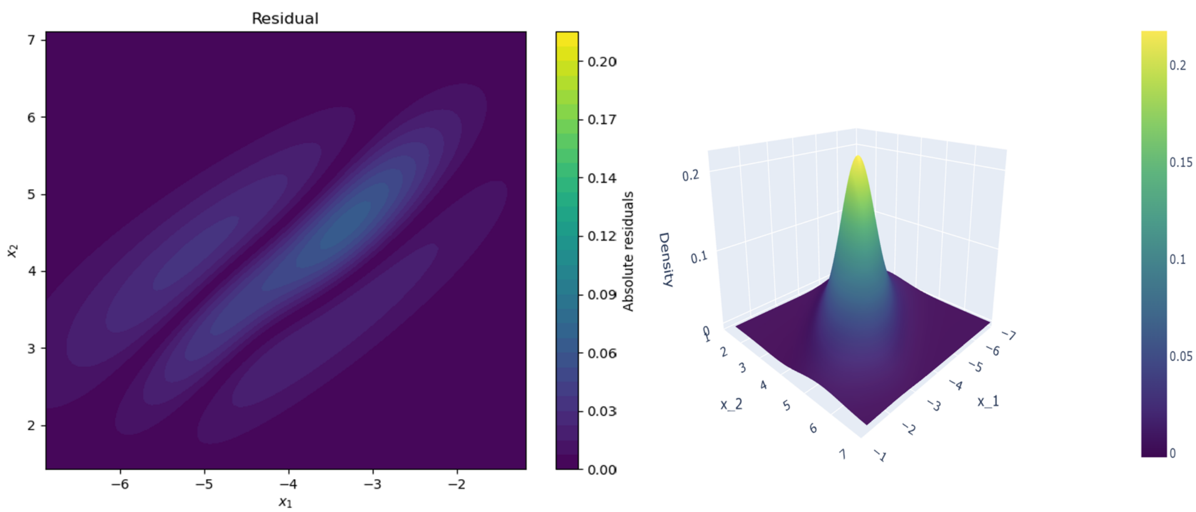

Figure 4.5: A heat map of the residual error between the estimated symbolic density (predicted by the expression at complexity 46) and the ground truth density (left). The maximum residual error is 0.06. A 3D plot of the recovered probability density function (right).

The results appear as a product of exponentials. This is a consequence of warm-starting the search with a product of marginals and conditionals, which are themselves normal distributions. Inspecting the simpler expressions, we observe that the results suggest a gaussian bump with means in the vicinity of the true mean for cluster 1 i.e. $[-4, 4]$. Expressions with the lowest loss (e.g. at complexities 46 and 48) capture interaction terms between x_1 and x_2 suggesting non-zero covariance. Symbolic regression may not recover the exact parameters of the distribution due to noise introduced by the non-parameteric density estimate, yet it nevertheless offers insight into the functional form of the distribution. These insights can subsequently be used to identify an acceptable model class so that data efficient parametric density estimation techniques can be employed to recover the parameters that best fit the data.

With similar results for cluster 2, the expressions for both clusters 1 and 2 can then be combined additively or multiplicatively to recover the original denisty function according to whether clustering or structure learning had been initially used to decompose the problem.

Hyper-parameters used to obtain these results are listed in Table A.1. The hyper-parameters for DBSCAN clustering and *pygmpy* structure learning were chosen based on visual inspection and data exploration.

4.2 Scaling Dimensionality

We attempt to characterize the performance of the technique as the number of covariates increases. We test PDF-SR on multivariate normal distributions with zero mean, unit variance, and covariances of 0.8 in dimensions two to six. The mean square loss (MSE) does not serve as a good metric for evaluating PDF-SR due to the unit volume constraint for valid probability density functions as the number of dimensions increases. The unit volume constraint pulls down the value of the density everywhere due to the exponential increase of the volume of the space. This results in a very low MSE loss in high dimensions, since some small constant ϵ is able to achieve low loss with respect to the density estimated on the support of the distribution.

To avoid this issue, we calculate the Spearman (rank) correlation between the true density and estimated density on a test set of samples not seen during the expression search. The Spearman correlation varies between -1 and 1 (higher is better), measuring the monotonicity of the relationship between the true density and the estimated density, and enables a fair evaluation of PDF-SR as dimensionality increases. Since the expressions returned by PDF-SR may evaluate as negative over a sample point, we take the maximum between the density predicted by the expression or 0 before calculating the score. This is done for each expression returned by PDF-SR, and the maximum Spearman correlation over all the expressions is taken as the final score.

For all experiments, we use the true density values as a target for PDF-SR rather than a KDE estimate. In the first set of experiments we restrict the set of operators to a limited set containing the operators necessary to recover the ground truth expression. We compute the Spearman correlation coefficient for dimensions 2 to 6, and repeat the experiment for PDF-SR run for 20 iterations, 200 iterations, and 2000 iterations. The results are shown in Figure 4.6. Box plots of the raw expression complexities are shown in Figure 4.7. A second set of experiments was repeated where we allow for all standard operators supported by *PySR* in addition to a few custom operator definitions to assess how PDF-SR performs with few prior assumptions. The results are shown in Figure 4.8. Box plots of the raw expression complexities are shown in Figure 4.9.

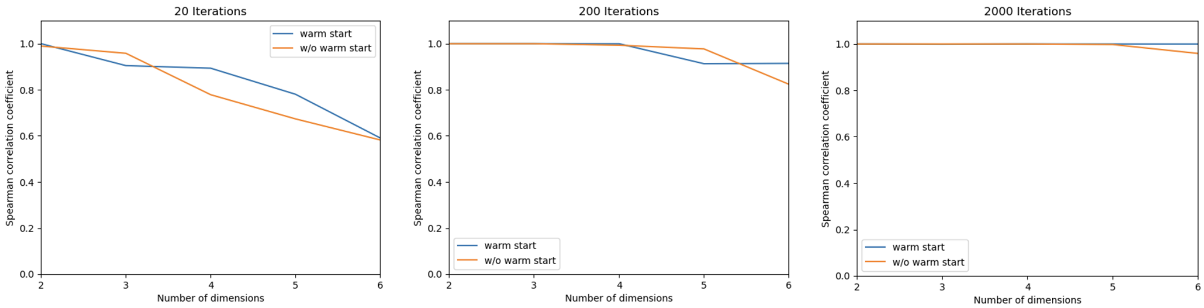

Figure 4.6: The Spearman correlation coefficient for PDF-SR with the restricted set of operators as the number of dimensions increases from 2 to 6 at 20 iterations (left), 200 iterations (center), and 2000 iterations (right).

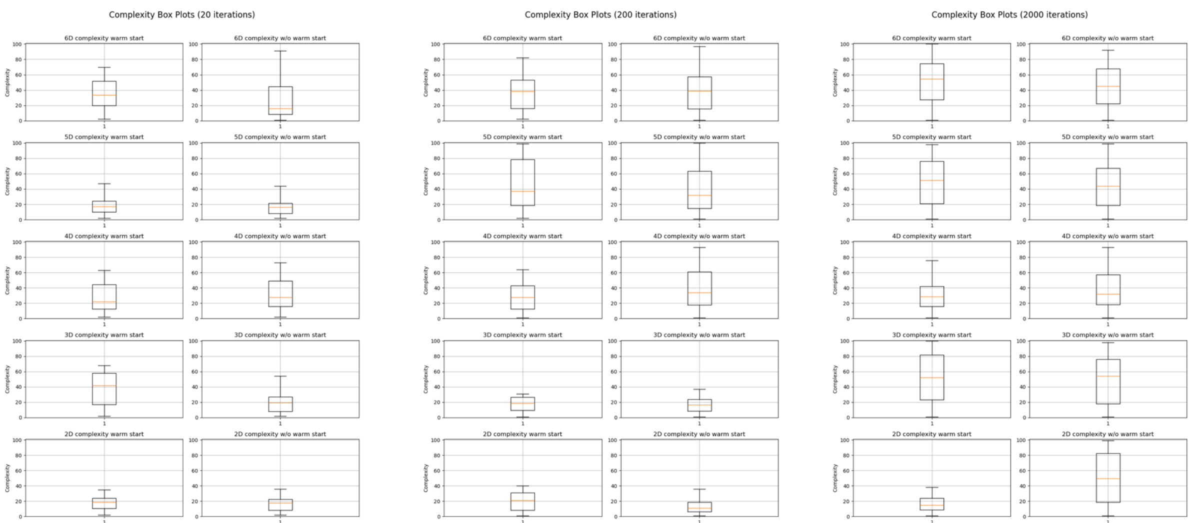

Figure 4.7: Box plots of the raw complexity scores of expressions returned by PDF-SR with the restricted set of operators for dimensions 2 (lowest row) to 6 (highest row) after 20 iterations (left column), 200 iterations (center column), and 2000 iterations (right).

Figure 4.6 does not appear to provide evidence in favor of warm-starting the joint expression search. While the warm started joint expression search consistently won out in 6 dimensions, it is only by a small margin, and sometimes under-performs at lower dimensions.

The box plots in Figure 4.7 seem to show that a larger range of complexity corresponds to worse Spearman correlation coefficients in Figure 4.6. When comparing warm start expression searches to expression search without the warm start, a squashed box plot with lower average complexity often predicts better Spearman correlation. The stretched box plots with larger average complexity scores are indicative of over-fitting and predict poor performance on the test set.

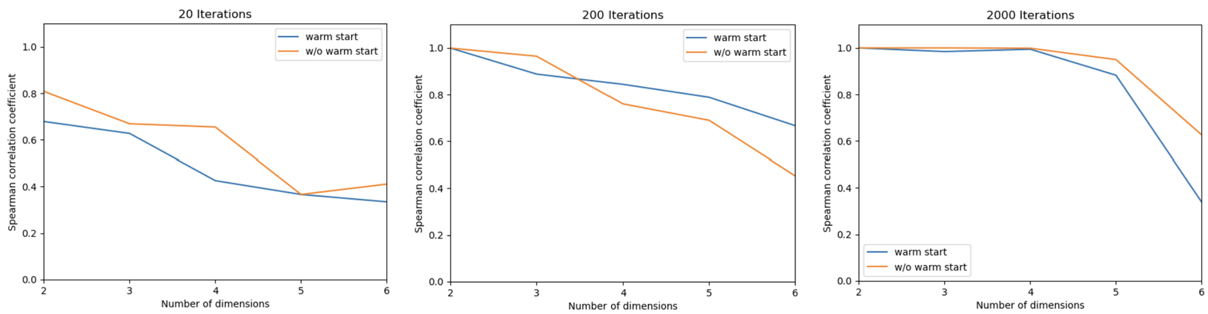

Figure 4.8: The Spearman correlation coefficient for PDF-SR with the extended set of operators as the number of dimensions increases from 2 to 6 at 20 iterations (left), 200 iterations (center), and 2000 iterations (right).

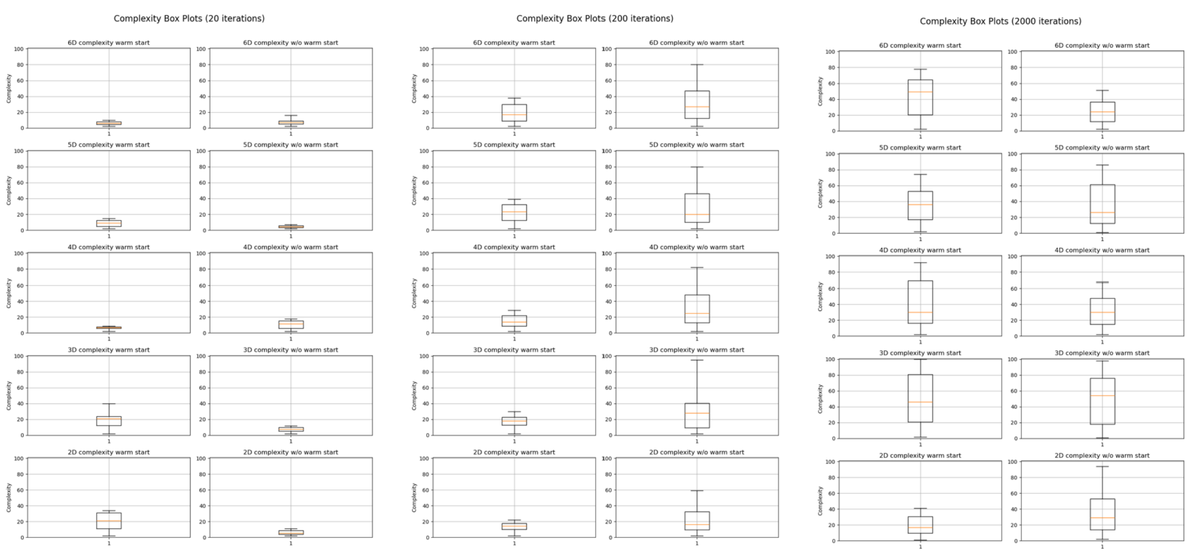

Figure 4.9: Box plots of the raw complexity scores of expressions returned by PDF-SR with the extended set of operators for dimensions 2 (lowest row) to 6 (highest row) after 20 iterations (left column), 200 iterations (center column), and 2000 iterations (right).

Figure 4.8 also does not provide evidence in favor of warm starts when PDF-SR is used with an extended set of operators. The observation made concerning how the box plots in Figure 4.9 relate to the Spearman correlation appear here as well. Although the results do not justify the use of warm starts quantitatively, there may be some qualitative benefit since it biases the search towards certain families of expressions over others. This is particularly useful for probability density functions with a compact support which may not have unique symbolic descriptions.

The results are sensitive to the choice of PDF-SR's hyperparameters and extensive empirical study is necessary to determine the regime where warm starts are useful. For instance, in the case of independence between all covariates, warm starting the joint expression search immediately recovers the correct expression provided that all marginals and conditionals are recovered exactly. The hyperparameters used in these experiments are listed in Tables A.2 and A.3.

4.3 Muon Decay

The data for the following example describe a process in particle physics and is an example of a scenario where tools such as PDF-SR may be useful for scientific advancement. The dataset describes muon decay, where a muon decays into an electron-antineutrino and a muon-neutrino. The dataset features, m_{13}^2 and m_{23}^2, are combinations of the momentum and energy of the outgoing particles. The ground truth expression is of the form,

$$\text{Normalization} \times \frac{(m_{23}^2 - m_\mu^2)(m_{23}^2 - m_e^2)}{(m_{13}^2 + m_{23}^2 - m_e^2 - m_\mu^2 + m_W^2)^2}$$

This expression is a rational function with a quadratic in m_{23}^2 as the numerator and a quadratic in both m_{13}^2 and m_{23}^2 in the denominator. For consistency with other examples, the variables m_{13}^2 and m_{23}^2 will hereafter be referred to as x_1 and x_2. Since the range of values of the raw variables is quite large, we apply min-max scaling to normalize the range of the variables to $[0, 1]$. This has the benefit of preserving the overall functional form of the distribution, albeit with a new parameterization and additional polynomial terms of the same order or lower. Figure 4.10 shows the distribution after min-max scaling is applied.

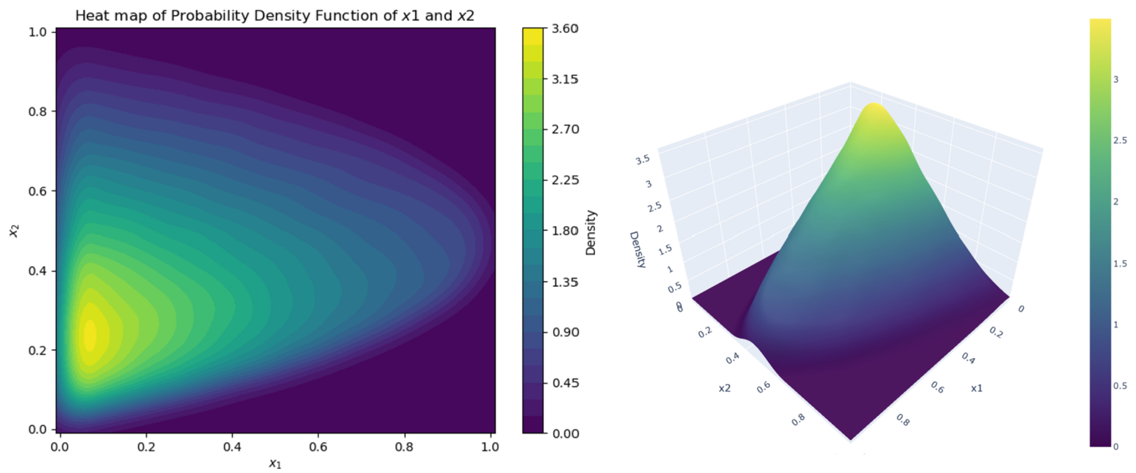

Figure 4.10: KDE estimate (left) and 3D plot associated with the Muon decay process (probability density function).

Since the expression does not describe the distribution over the entire input space, we first estimate the support of the distribution. Figure 4.11, shows the support estimated via the convex hull of the samples, with a small offset applied to trim any boundary artifacts resulting from the KDE estimate.

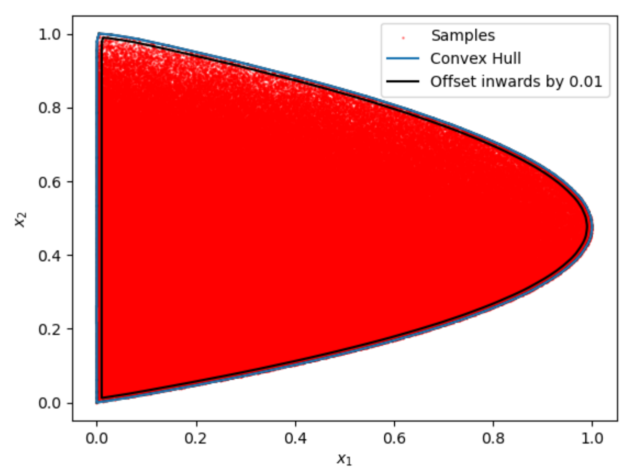

Figure 4.11: Convex hull estimate of the support of the probability density function.

Figure 4.12 and Table 4.3 display the expressions recovered by PDF-SR. We see that the expressions achieve very low loss, though the original rational function is not recovered. This result illustrates the non-uniqueness of symbolic density estimation as several expressive families of expressions exist which are able to approximate functions over a compact support arbitrarily well.

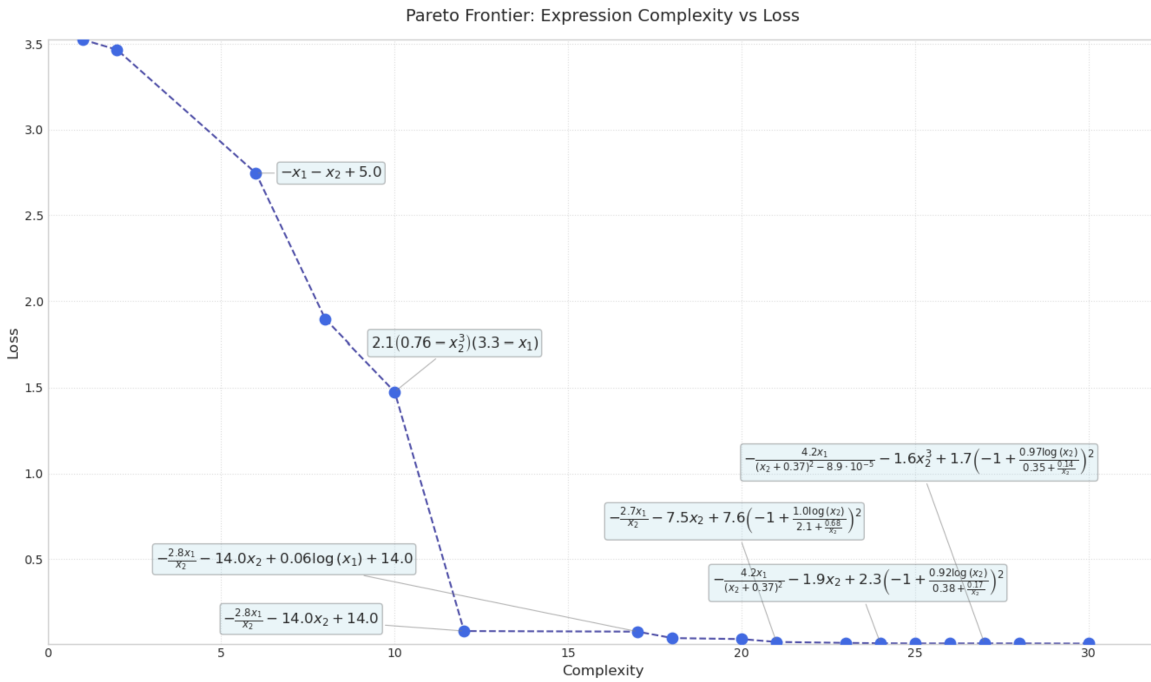

Figure 4.12: (Pareto frontier associated with the Muon decay process (simplified expressions shown).

Table 4.3: New expressions simplified with *sympy* alongside their raw complexity score from PySR.

Raw Complexity	Simplified Expression
1	3.5
2	3.7
6	$-x_1 - x_2 + 5.0$
8	$(x_2 - 3.7)\,(e^{x_2} - 2.8)$
10	$2.1\,(x_1 - 3.3)\,\left(x_2^3 - 0.76\right)$
12	$-\frac{2.8\,x_1}{x_2} - 14.0\,x_2 + 14.0$
17	$-\frac{2.8\,x_1}{x_2} - 14.0\,x_2 + 0.06\,\log(x_1) + 14.0$
18	$\frac{\left(3.7\,x_1 - 15.0\,x_2\right)\left(x_2 + 0.015\,e^{3\,x_1} - 0.95\right)}{x_2}$
20	$\frac{\left(3.5\,x_1 - 16.0\,x_2 + 0.18\right)\left(x_2 + 0.016\,e^{3\,x_1} - 0.95\right)}{x_2}$
21	$-\frac{2.7\,x_1}{x_2} - 7.5\,x_2 + \frac{7.6\left(-0.48\,x_2\,\log(x_2) + x_2 + 0.32\right)^2}{\left(x_2 + 0.32\right)^2}$
23	$-\frac{3.1\,x_1}{x_2 + 0.039} - 4.2\,x_2 + 4.5\left(1 - \frac{0.96\,\log(x_2)}{0.8 + \frac{0.35}{x_2}}\right)^2$
24	$-\frac{4.2\,x_1}{\left(x_2 + 0.37\right)^2} - 1.9\,x_2 + \frac{13.0\left(-x_2\,\log(x_2) + 0.41\,x_2 + 0.18\right)^2}{\left(x_2 + 0.45\right)^2}$
25	$-\frac{4.2\,x_1}{\left(x_2 + 0.37\right)^2} - 1.6\,x_2^3 + \frac{13.0\left(-x_2\,\log(x_2) + 0.36\,x_2 + 0.14\right)^2}{\left(x_2 + 0.4\right)^2}$
26	$-\frac{4.2\,x_1}{\left(x_2 + 0.37\right)^2} - 1.6\,x_2^3 + \frac{13.0\left(-x_2\,\log(x_2) + 0.36\,x_2 + 0.14\right)^2}{\left(x_2 + 0.4\right)^2}$
27	$-\frac{4.2\,x_1}{\left(x_2 + 0.37\right)^2 - 8.9 \times 10^{-5}} - 1.6\,x_2^3 + 1.7\left(1 - \frac{0.97\,\log(x_2)}{0.35 + \frac{0.14}{x_2}}\right)^2$
28	$-\frac{4.2\,x_1}{\left(x_2 + 0.37\right)^2} - 2.4\left(x_2 - 0.13\right)^3 + \frac{14.0\left(-x_2\,\log(x_2) + 0.34\,x_2 + 0.13\right)^2}{\left(x_2 + 0.38\right)^2}$
30	$-\frac{4.2\,x_1}{\left(x_2 + 0.36\right)^2} - 1.8\,x_2^3 + \frac{11.0\left(-x_1 x_2\,\log(x_2) + 0.4\,x_1 x_2 + 0.14\,x_1 + 0.0013\right)^2}{\left(x_1 x_2 + 0.36\,x_1 + 0.0033\right)^2}$

Finally, Figure 4.13 displays the residual error between the recovered density and the kernel density estimate from the samples. We see that recovered expression provides an excellent fit to the data within the estimated support.

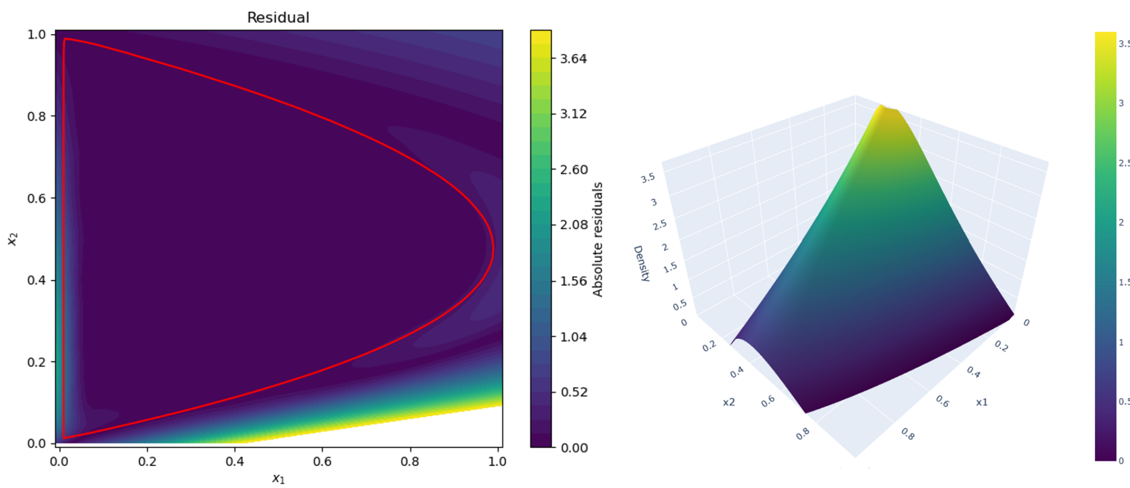

Figure 4.13: A heat map of the residual error between the estimated symbolic density (predicted by the expression at complexity 27) and the kernel density estimate. The estimated support is enclosed in the red curve (left).A 3D plot of the recovered probability density function (right).

4.4 Rastrigin Toy PDF

We use a variant of the Rastrigin function as a toy probability density function. The Rastrigin function is a non-convex, multimodal function commonly used for testing optimization algorithms but not typically studied in the context of probability density functions.

To adapt the Rastrigin function into a pseudo-probability density, we constrain its support, normalize it, and apply a bias to ensure non-negativity over the support to satisfy the requirements of a valid probability distribution. The ground truth expression is given in Table 4.4 and consists of quadratic terms, and cosine terms in both variables.

Table 4.4: Variant of Rastrigin function used as a probability density function

Ground Truth Expression
$f(x_1, x_2) = \frac{1}{586.67}\left(10 + 10x_1^2 + 10x_2^2 - 5\cos(3\pi x_1 - 6.1) - 5\cos(3\pi x_2 - 6.1)\right)$

The KDE density estimate of the Rastrigin variant is shown in 4.14. We observe the characteristic bowl shape due to the quadratic terms and peaks and valleys due to the sinusoidal terms. The pareto front of expressions recovered by PDF-SR are shown in Figure 4.15 and Table 4.16. Note that the recovered expressions describe the nonnormalized density, and consequently the parameters do not match the ground truth exactly although the function form is recovered perfectly.

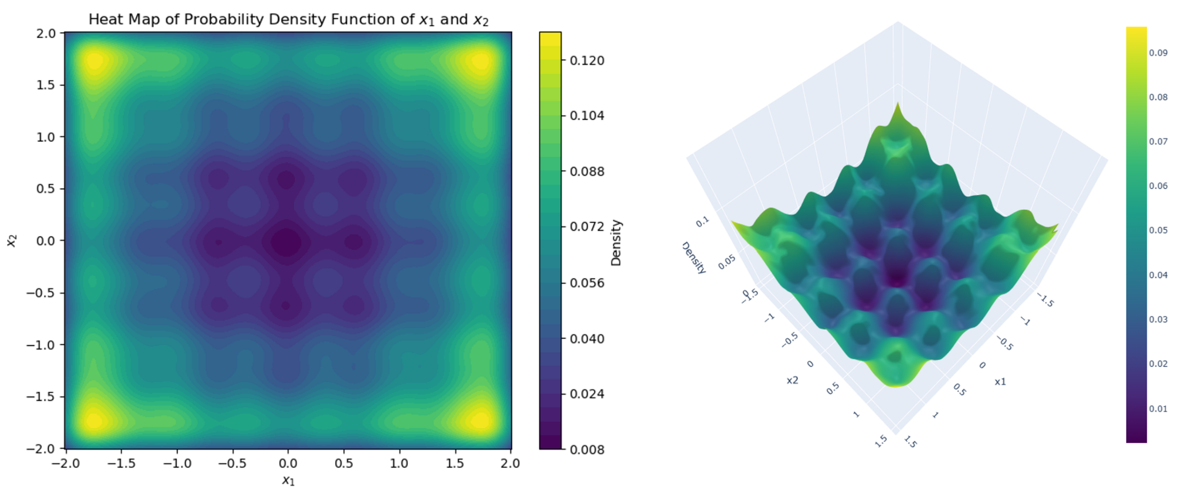

Figure 4.14: The KDE density estimate as a heat map (left). The 3D plot of the same KDE density estimate (right).

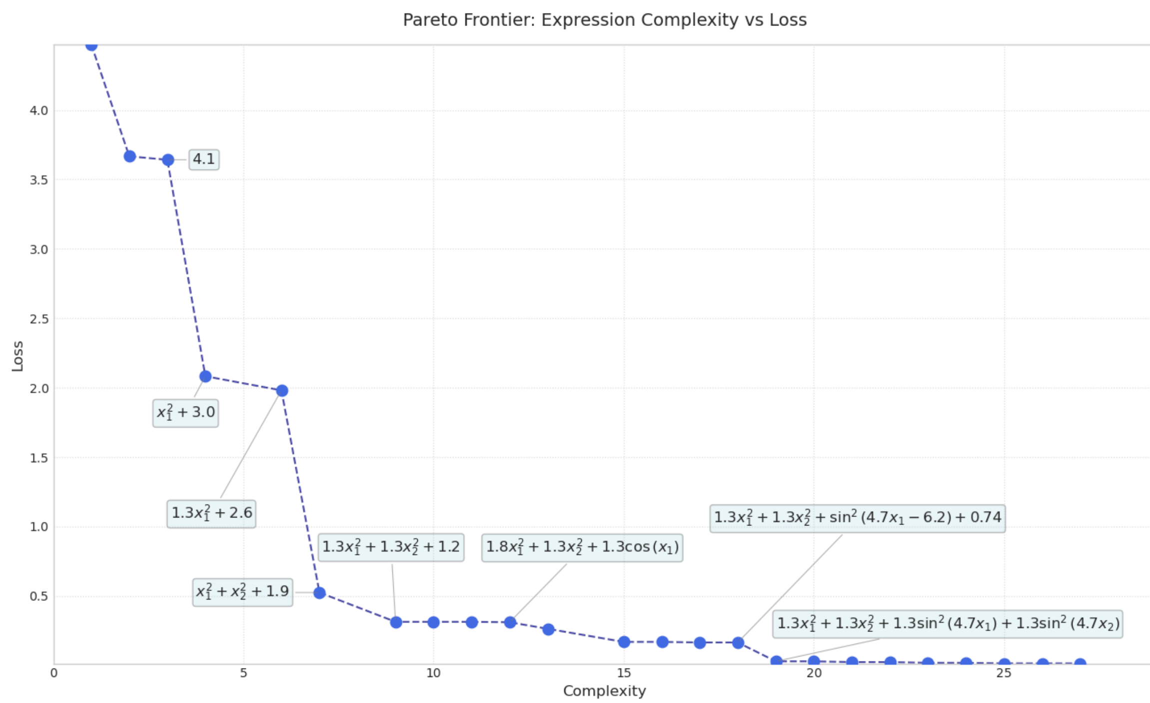

Figure 4.15: (Pareto frontier associated with the rastrigin probability density function (simplified expressions shown).

Table 4.5: Expressions simplified with *sympy* alongside their raw complexity score from PySR. Expressions that simplify to constants are not shown.

Raw Complexity	**Simplified Expression**
4	$x_1^2 + 3.0$
6	$1.3x_1^2 + 2.6$
7	$x_1^2 + x_2^2 + 1.9$
9	$1.3x_1^2 + 1.3x_2^2 + 1.2$
10	$1.3x_1^2 + 1.3x_2^2 + 1.2$
11	$1.3x_2^2 + 1.3\,(x_1 + 0.0042)^2 + 1.2$
12	$1.8x_1^2 + 1.3x_2^2 + 1.3\cos(x_1)$
13	$1.5x_1^2 + 1.5x_2^2 + 1.5\sin^2(4.7x_1)$
15	$1.3x_1^2 + 1.3x_2^2 - 0.5\cos(9.4x_1) + 1.2$
16	$1.3x_1^2 + 1.3x_2^2 - 0.5\cos(9.4x_1) + 1.2$
17	$1.3x_1^2 + 1.3x_2^2 + \sin^2(4.7x_1 - 6.2) + 0.73$
18	$1.3x_1^2 + 1.3x_2^2 + \sin^2(4.7x_1 - 6.2) + 0.74$
19	$1.3x_1^2 + 1.3x_2^2 + 1.3\sin^2(4.7x_1) + 1.3\sin^2(4.7x_2)$
20	$1.3x_1^2 + 1.3x_2^2 + 1.3\sin^2(4.7x_1) + 1.3\sin^2(4.7x_2)$
21	$1.3x_1^2 + 1.3x_2^2 - 0.5\cos(9.4x_1) - 0.5\cos(9.4x_2) + 1.3$
22	$1.3x_1^2 + 1.3x_2^2 - 0.5\cos(9.4x_1) - 0.5\cos(9.4x_2) + 1.3$
23	$1.3x_1^2 + 1.3x_2^2 + \sin^2(4.7x_1) + \sin^2(4.7x_2 - 3.0) + 0.27$
24	$1.3x_1^2 + 1.3x_2^2 + \sin^2(4.7x_1) + \sin^2(4.7x_2 + 3.2) + 0.27$
25	$1.3x_1^2 + 1.3x_2^2 + \sin^2(4.7x_1 + 3.2) + \sin^2(4.7x_2 + 0.1) + 0.27$
26	$1.3x_1^2 + 1.3x_2^2 + \sin^2(4.7x_1 + 0.083) + \sin^2(4.7x_2 + 3.2) + 0.27$
27	$1.3x_1^2 + 1.3\,(x_2 + 0.0019)^2 + \sin^2(4.7x_1 + 3.2) + \sin^2(4.7x_2 + 0.1) + 0.27$

The density predicted by the recovered symbolic expression exhibits a low residual error after rescaling the prediction. Moreover, the correct functional form of the probability density function is clearly identified in expressions of complexity 19 and higher. The hyper-parameters used in PDF-SR for this example are listed in Table A.5.

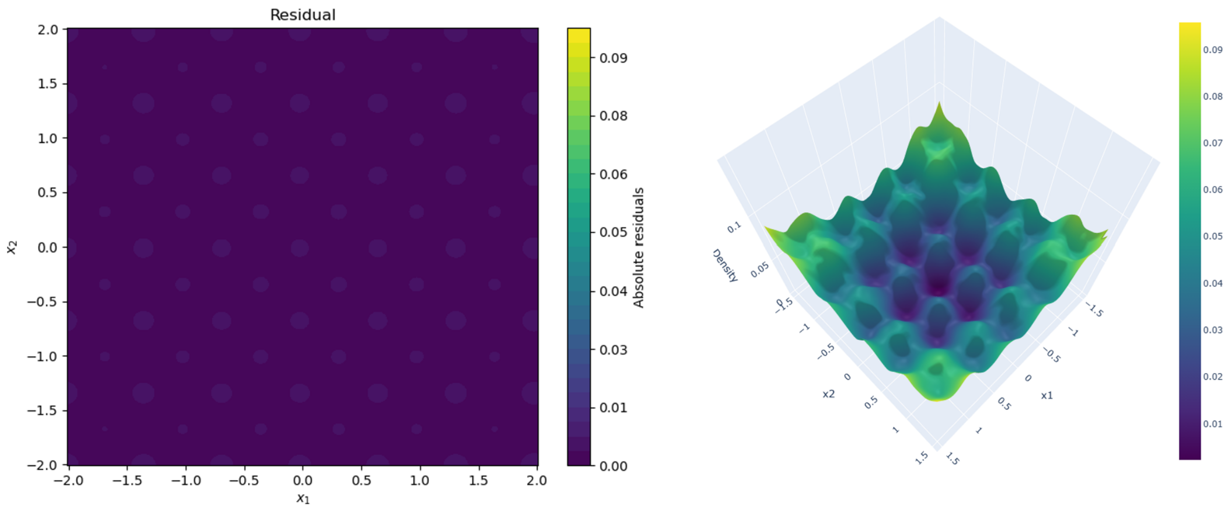

Figure 4.16: A heat map of the residual error between the estimated symbolic density (predicted by the expression at complexity 26) and the ground truth density (left). The peak residual error is 0.004. A 3D plot of the recovered probability density function (right).

4.5 Conclusion

The results presented demonstrate the viability of symbolic regression as a method for recovering interpretable expressions of probability density functions from samples drawn from the distributions. The proposed PDF-SR workflow incorporates clustering, structure learning, non-parametric density estimation, in addition to an attempt to reduce the complexity of the symbolic regression task via warm-starting the expression search with informative marginal and conditional expressions. Empirical evaluations on synthetic and real-world datasets, including the muon decay distribution, indicate that the method is capable of capturing structural features of the true density. It should be noted, however, that certain distributions may not have unique descriptions or may not even have analytic descriptions.

The decomposition of high-dimensional density estimation problems into tractable subproblems may provide qualitative benefits by biasing the search towards certain families of expressions, though preliminary results suggest that this offers little utility in scaling the technique to high dimensions. Additional empirical study is required to evaluate the robustness of the technique in higher-dimensional regimes.

The pipeline also provides qualitative insight into the trade-offs between complexity and fidelity, as shown by the pareto frontiers of symbolic expressions. These insights can be valuable in scientific domains where model interpretability is paramount. Future work may extend this framework by incorporating domain-specific priors more directly into the symbolic search process, refining the use of large language models for expression generation, and improving support estimation in high-dimensional settings. The loss function used for all PDF-SR examples presented in this paper was the mean squared error loss. Further improvement may be gained by using a custom loss function to enforce non-negativity and unit volume constraints.

Finally, hyper-parameter selection for PDF-SR is significant challenge, in particular due to the large number of hyperparameters that govern PDF-SR's behavior. For the various examples here, the hyper-parameters were chosen based on human judgment, data exploration, and recommended defaults. It is possible to incorporate cross-validation and other standard methods for hyperparameter selection, though further empirical study is necessary to determine which hyperparameters are important and which aren't in various problem settings.

An interesting extension to this work would be to consider probability density functions that are defined on manifolds within the input space rather than directly over the input space. Discovering symbolic expressions (if they exist) in such cases would require determining the data manifold and obtaining a symbolic expression for the manifold, for example a sphere in $\mathbb{R}^3$; determining an appropriate coordinate transform, such as Cartesian to spherical for the example of a sphere; and finally using symbolic regression on the transformed data.

Although the symbolic expressions recovered do not always precisely match the ground truth due to limitations introduced by nonparametric estimation and the non-uniqueness of symbolic representations, they nevertheless provide useful approximations that can guide further model selection and parametric fitting. PDF-SR and similar techniques may offer interpretable insights into otherwise inscrutable data-generating processes.

References

[1] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*, 2024.

[2] Ahmed M Alaa and Mihaela van der Schaar. Demystifying black-box models with symbolic metamodels. *Advances in neural information processing systems*, 32, 2019.

[3] Lucio Anderlini. Density estimation trees as fast non-parametric modelling tools. *Journal of Physics: Conference Series*, 762:012042, October 2016.

[4] Ankur Ankan and Johannes Textor. pgmpy: A python toolkit for bayesian networks, 2023.

[5] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *International Conference on Machine Learning*, pages 936–945. Pmlr, 2021.

[6] Christopher Michael Bishop and Hugh Bishop. *Deep Learning - Foundations and Concepts*. 1 edition, 2023.

[7] Clément Bonnet and Matthew V Macfarlane. Searching latent program spaces. *arXiv preprint arXiv:2411.08706*, 2024.

[8] Curtis Bright, Jürgen Gerhard, Ilias Kotsireas, and Vijay Ganesh. Effective problem solving using sat solvers. In *Maple in Mathematics Education and Research: Third Maple Conference, MC 2019, Waterloo, Ontario, Canada, October 15–17, 2019, Proceedings 3*, pages 205–219. Springer, 2020.

[9] Benoit Cadre. Kernel estimation of density level sets, 2005.

[10] Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2013.

[11] Yen-Chi Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*, 1(1):161–187, 2017.

[12] Yen-Chi Chen, Christopher R Genovese, and Larry Wasserman. A comprehensive approach to mode clustering. 2016.

[13] David Maxwell Chickering. Optimal structure identification with greedy search. *J. Mach. Learn. Res.*, 3(null):507–554, March 2003.

[14] Xieting Chu, Hongjue Zhao, Enze Xu, Hairong Qi, Minghan Chen, and Huajie Shao. Neural symbolic regression using control variables. *arXiv preprint arXiv:2306.04718*, 2023.

[15] Xieting Chu, Hongjue Zhao, Enze Xu, Hairong Qi, Minghan Chen, and Huajie Shao. Scalable neural symbolic regression using control variables, 2024.

[16] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[17] Miles Cranmer. Interpretable machine learning for science with pysr and symbolicregression. jl. *arXiv preprint arXiv:2305.01582*, 2023.

[18] Li Cui. *Shape and Geometry in Statistics: Support and Density Estimation.* Ph.d. dissertation, University of California, Davis, Davis, CA, 2021. ProQuest ID: Cui$_u cdavis_0 029 D_2 0897; MerrittID : ark : /13030/m5b34thp$.

[19] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp, 2017.

[20] Jerome H. Friedman, Werner Stuetzle, and Anne Schroeder and. Projection pursuit density estimation. *Journal of the American Statistical Association*, 79(387):599–608, 1984.

[21] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975.

[22] Aishik Ghosh, Xiangyang Ju, Benjamin Nachman, and Andrzej Siodmok. Towards a deep learning model for hadronization. *Physical Review D*, 106(9):096020, 2022.

[23] Arya Grayeli, Atharva Sehgal, Omar Costilla-Reyes, Miles Cranmer, and Swarat Chaudhuri. Symbolic regression with a learned concept library, 2024.

[24] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[25] Jennifer Jang and Heinrich Jiang. Dbscan++: Towards fast and scalable density clustering, 2019.

[26] Piyush Jha, Prithwish Jana, Arnav Arora, and Vijay Ganesh. Rlsf: Reinforcement learning via symbolic feedback. *arXiv preprint arXiv:2405.16661*, 2024.

[27] Nan Jiang and Yexiang Xue. Symbolic regression via control variable genetic programming, 2023.

[28] Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. *Advances in Neural Information Processing Systems*, 35:10269–10281, 2022.

[29] J.N. Kapur. *Maximum-entropy Models in Science and Engineering*. Wiley, 1989.

[30] John R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, USA, 1992.

[31] Tiziano Labruna, Jon Ander Campos, and Gorka Azkune. When to retrieve: Teaching llms to utilize information retrieval effectively. *arXiv preprint arXiv:2404.19705*, 2024.

[32] William B Langdon and Steven Gustafson. Genetic programming and evolvable machines: Five years of reviews. *Genetic Programming and Evolvable Machines*, 6(2):221–228, 2005.

[33] Qiao Liu, Jiaze Xu, Rui Jiang, and Wing Hung Wong. Density estimation using deep generative neural networks. *Proceedings of the National Academy of Sciences*, 118(15), April 2021.

[34] Matteo Merler, Katsiaryna Haitsiukevich, Nicola Dainese, and Pekka Marttinen. In-context symbolic regression: Leveraging large language models for function discovery. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 589–606, 2024.

[35] Sebastian Mežnar, Sašo Džeroski, and Ljupčo Todorovski. Efficient generator of mathematical expressions for symbolic regression. *Machine Learning*, 112(11):4563–4596, 2023.

[36] T Nathan Mundhenk, Mikel Landajuela, Ruben Glatt, Claudio P Santiago, Daniel M Faissol, and Brenden K Petersen. Symbolic regression via neural-guided genetic programming population seeding. *arXiv preprint arXiv:2111.00053*, 2021.

[37] Siqi Nie, Denis Deratani Maua, Cassio Polpo de Campos, and Qiang Ji. Advances in learning bayesian networks of bounded treewidth, 2014.

[38] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[39] Brenden K Petersen, Mikel Landajuela, T Nathan Mundhenk, Claudio P Santiago, Soo K Kim, and Joanne T Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *Proc. of the International Conference on Learning Representations*, 2021.

[40] John Platt, Bernhard Schölkopf, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. Technical Report MSR-TR-99-87, November 1999.

[41] Sergei Popov, Mikhail Lazarev, Vladislav Belavin, Denis Derkach, and Andrey Ustyuzhanin. Symbolic expression generation via variational auto-encoder. *PeerJ Computer Science*, 9:e1241, 2023.

[42] Alessandro Rudi, Ernesto De Vito, Alessandro Verri, and Francesca Odone. Regularized kernel algorithms for support estimation. *Frontiers in Applied Mathematics and Statistics*, Volume 3 - 2017, 2017.

[43] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[44] David W Scott. On optimal and data-based histograms. *Biometrika*, 66(3):605–610, 1979.

[45] Parshin Shojaee, Kazem Meidani, Amir Barati Farimani, and Chandan K. Reddy. Transformer-based planning for symbolic regression. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[46] Fangzheng Sun, Yang Liu, Jian-Xun Wang, and Hao Sun. Symbolic physics learner: Discovering governing equations via monte carlo tree search. *arXiv preprint arXiv:2205.13134*, 2022.

[47] Tony Tohme, Mohsen Sadr, Kamal Youcef-Toumi, and Nicolas G. Hadjiconstantinou. Messy estimation: Maximum-entropy based stochastic and symbolic density estimation, 2024.

[48] Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4860–4871. Curran Associates, Inc., 2020.

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

[50] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation, 2016.

[51] Mojtaba Valipour, Bowen You, Maysum Panju, and Ali Ghodsi. Symbolicgpt: A generative transformer model for symbolic regression. *arXiv preprint arXiv:2106.14131*, 2021.

[52] M. Virgolin and Solon P. Pissis. Symbolic regression is np-hard. *Trans. Mach. Learn. Res.*, 2022, 2022.

[53] Marco Virgolin and Solon P Pissis. Symbolic regression is np-hard. *arXiv preprint arXiv:2207.01018*, 2022.

[54] M. P. Wand and M. C. Jones. *Kernel smoothing / M.P. Wand, M.C. Jones.* Monographs on statistics and applied probability ; 60. Chapman Hall, London ;, 1st ed. edition, 1995.

[55] Daren Wang, Xinyang Lu, and Alessandro Rinaldo. Dbscan: Optimal rates for density based clustering, 2019.

[56] Wikipedia contributors. List of probability distributions – Wikipedia, The Free Encyclopedia, 2024. [Online; accessed 10-September-2024].

[57] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on*

empirical methods in natural language processing: system demonstrations, pages 38–45, 2020.

[58] Yue Yu, Jie Chen, Tian Gao, and Mo Yu. DAG-GNN: DAG structure learning with graph neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7154–7163. PMLR, 09–15 Jun 2019.

[59] Xun Zheng, Bryon Aragam, Pradeep Ravikumar, and Eric P. Xing. Dags with no tears: Continuous optimization for structure learning, 2018.

APPENDICES

Appendix A

Hyper-parameters for PDF-SR Experiments

A.1 Mixture of two 2D Normal Distributions

Table A.1: Hyper-parameters for symbolic regression on each cluster

Hyper-parameter	**Value**	**Notes**
	DBSCAN Clustering	
eps	5	Maximum Neighborhood radius.
min samples	10	Neighborhood minimum size.
	pgmpy Structure Learning	
CI test	Pearson correlation	Conditional independence test.
	Kernel Density Estimation	
bandwidth	0.356	KDE bandwidth selected using scotts factor.
	PySR	
binary operators	[+, -, *, /, pow_int]	pow_int raises a base to an integer exponent.
unary operators	[exp, log]	Allowed unary operators.
maxsize	50	Maximum size of an expression tree.
ncycles per iteration	380	Cycles per iteration of the genetic algorithm.
niterations for marginal sr	4000	Iterations for marginal SR.
niterations for conditional sr	4000	Iterations for conditional SR.
niterations for joint sr	8000	Iterations for joint SR.
num populations for marginal sr	15	Number of populations for marginal SR.
num populations for conditional sr	15	Number of populations for conditional SR.
num populations for joint sr	15	Number of populations for joint SR.
population size for marginal sr	30	Number of expressions per population in marginal SR.
population size for conditional sr	30	Number of expressions per population in conditional SR.
population size for joint sr	30	Number of expressions per population in joint SR
elementwise loss	MSE	Loss function used for SR.

A.2 Dimensionality Scaling Experiments

Table A.2: Hyper-parameters for the dimensionality study with restricted operators

Hyper-parameter	**Value**	**Notes**
	Kernel Density Estimation	
None	—	True density values used
	PySR	
binary operators	[+, -, *, /]	Allowed binary operators.
unary operators	[pow2, pow3, exp, log]	Allowed unary operators.
maxsize	100	Maximum size of an expression tree.
ncycles per iteration	100	Cycles per iteration of the genetic algorithm.
niterations for marginal sr	2000	Iterations for marginal SR.
niterations for conditional sr	2000	Iterations for conditional SR.
niterations for joint sr	[20, 200, 2000]	Iterations for joint SR.
num populations for marginal sr	15	Number of populations for marginal SR.
num populations for conditional sr	15	Number of populations for conditional SR.
num populations for joint sr	15	Number of populations for joint SR.
population size for marginal sr	30	Number of expressions per population in marginal SR.
population size for conditional sr	30	Number of expressions per population in conditional SR.
population size for joint sr	30	Number of expressions per population in joint SR
elementwise loss	MSE	Loss function used for SR.

Table A.3: Hyper-parameters for the dimensionality study with extended operators

Hyper-parameter	**Value**	**Notes**
Kernel Density Estimation		
None	—	True density values used
PySR		
binary operators	[+, -, *, /, ^]	Allowed binary operators.
unary operators	[exp, log, pow2, pow3, pow4, pow5, sin, cos, sqrt, abs, inv, asin, acos, atan, tan, acosh]	Allowed unary operators.
maxsize	100	Maximum size of an expression tree.
ncycles per iteration	100	Cycles per iteration of the genetic algorithm.
niterations for marginal sr	2000	Iterations for marginal SR.
niterations for conditional sr	2000	Iterations for conditional SR.
niterations for joint sr	[20, 200, 2000]	Iterations for joint SR.
num populations for marginal sr	15	Number of populations for marginal SR.
num populations for conditional sr	15	Number of populations for conditional SR.
num populations for joint sr	15	Number of populations for joint SR.
population size for marginal sr	30	Number of expressions per population in marginal SR.
population size for conditional sr	30	Number of expressions per population in conditional SR.
population size for joint sr	30	Number of expressions per population in joint SR
elementwise loss	MSE	Loss function used for SR.

A.3 Rastrigin Toy PDF

Table A.4: Hyper-parameters for symbolic regression on the Rastrigin PDF

Hyper-parameter	**Value**	**Notes**
	Kernel Density Estimation	
bandwidth	0.12	Selected by visual inspection.
	PySR	
binary operators	[+, -, *, /]	Allowed binary operators.
unary operators	[pow2, cos, sin, exp, log]	Allowed unary operators.
maxsize	50	Maximum size of an expression tree.
ncycles per iteration	380	Cycles per iteration of the genetic algorithm.
niterations for marginal sr	4000	Iterations for marginal SR.
niterations for conditional sr	4000	Iterations for conditional SR.
niterations for joint sr	8000	Iterations for joint SR.
num populations for marginal sr	15	Number of populations for marginal SR.
num populations for conditional sr	15	Number of populations for conditional SR.
num populations for joint sr	15	Number of populations for joint SR.
population size for marginal sr	30	Number of expressions per population in marginal SR.
population size for conditional sr	30	Number of expressions per population in conditional SR.
population size for joint sr	30	Number of expressions per population in joint SR
elementwise loss	MSE	Loss function used for SR.

A.4 Muon Decay

Table A.5: Hyper-parameters for symbolic regression on the Muon Decay PDF

Hyper-parameter	**Value**	**Notes**
	Kernel Density Estimation	
bandwidth	0.3	Selected by visual inspection.
	PySR	
binary operators	[+, -, *, /]	Allowed binary operators.
unary operators	[pow2, pow3, exp, log]	Allowed unary operators.
maxsize	30	Maximum size of an expression tree.
ncycles per iteration	380	Cycles per iteration of the genetic algorithm.
niterations for marginal sr	4000	Iterations for marginal SR.
niterations for conditional sr	4000	Iterations for conditional SR.
niterations for joint sr	8000	Iterations for joint SR.
num populations for marginal sr	15	Number of populations for marginal SR.
num populations for conditional sr	15	Number of populations for conditional SR.
num populations for joint sr	15	Number of populations for joint SR.
population size for marginal sr	30	Number of expressions per population in marginal SR.
population size for conditional sr	30	Number of expressions per population in conditional SR.
population size for joint sr	30	Number of expressions per population in joint SR
elementwise loss	MSE	Loss function used for SR.