Recent Advancements in Symbolic Regression at Top ML Conferences (ICML, NeurIPS, ICLR)

Hengzhe Zhang

Victoria University of Wellington

06/12/2024

Table of Contents

Introduction

Machine Learning for Symbolic Regression:

- Symbolic regression has garnered significant attention in recent years within the field of machine learning.
- This work focuses on leveraging neural networks to address symbolic regression problems, a topic widely discussed at top-tier ML conferences.

Categories of Deep Symbolic Regression Methods:

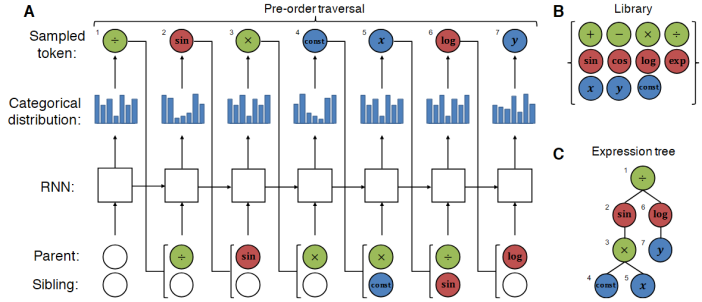
- **Reinforcement Learning-Based Symbolic Regression**
- **Language Model-Based Symbolic Regression**
- **Sparse Learning-Based Symbolic Regression**

Reinforcement Learning-Based Symbolic Regression

Motivation

Proposed Solution:

- Utilize a recurrent neural network (RNN) to generate mathematical expressions.
- Train using a *risk-seeking policy gradient* to optimize best-case performance.



Risk-Seeking Policy Gradient vs. Standard Policy Gradient

Standard Policy Gradient (SPG):

- Objective:

$$J_{\text{std}}(\theta) = \mathbb{E}_{\tau \sim p(\tau|\theta)} [R(\tau)]$$

- Gradient:

$$\nabla_\theta J_{\text{std}}(\theta) = \mathbb{E}_{\tau \sim p(\tau|\theta)} [R(\tau) \nabla_\theta \log p(\tau|\theta)]$$

- Focuses on **average performance**.

Risk-Seeking Policy Gradient (RSPG):

- Objective:

$$J_{\text{risk}}(\theta; \epsilon) = \mathbb{E}_{\tau \sim p(\tau|\theta)} \left[R(\tau) \,\middle|\, R(\tau) \geq R_\epsilon(\theta) \right]$$

- Gradient:

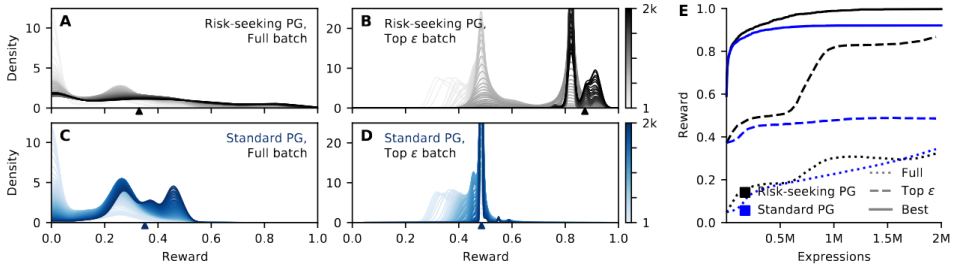
$$\nabla_\theta J_{\text{risk}}(\theta; \epsilon) = \mathbb{E}_{\tau \sim p(\tau|\theta)} \left[(R(\tau) - R_\epsilon(\theta)) \nabla_\theta \log p(\tau|\theta) \,\middle|\, R(\tau) \geq R_\epsilon(\theta) \right]$$

- Focuses on **best-case performance (top-ϵ rewards)**.

Key Differences:

- RSPG modifies the gradient to emphasize high-reward trajectories, ignoring lower rewards.

Risk-Seeking vs. Standard Policy Gradient



- **A and B: Risk-Seeking Policy Gradient (RSPG)**
  - Focuses on top-ϵ rewards, prioritizing best-case outcomes.
  - Shifts the distribution toward higher rewards over time.
- **C and D: Standard Policy Gradient (SPG)**
  - Optimizes for mean performance.
  - Peaks early but lacks focus on top-reward trajectories.
- **E: Reward Comparison**
  - RSPG achieves higher best-case and top-ϵ rewards.
  - SPG demonstrates better average reward (full batch).

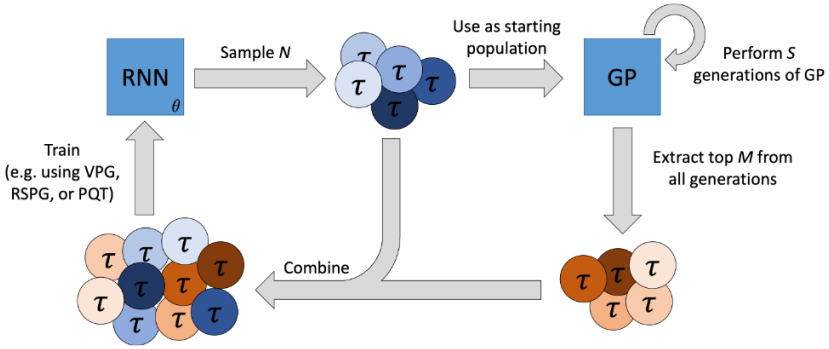
Deep Symbolic Regression

Benchmark	Expression	DSR	PQT	VPG	GP	Eureqa	Wolfram
Nguyen-1	$x^3 + x^2 + x$	100%	100%	96%	100%	100%	100%
Nguyen-2	$x^4 + x^3 + x^2 + x$	100%	99%	47%	97%	100%	100%
Nguyen-3	$x^5 + x^4 + x^3 + x^2 + x$	100%	86%	4%	100%	95%	100%
Nguyen-4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	100%	93%	1%	100%	70%	100%
Nguyen-5	$\sin(x^2)\cos(x) - 1$	72%	73%	5%	45%	73%	2%
Nguyen-6	$\sin(x) + \sin(x + x^2)$	100%	98%	100%	91%	100%	1%
Nguyen-7	$\log(x + 1) + \log(x^2 + 1)$	35%	41%	3%	0%	85%	0%
Nguyen-8	$\sqrt{x}$	96%	21%	5%	5%	0%	71%
Nguyen-9	$\sin(x) + \sin(y^2)$	100%	100%	100%	100%	100%	–
Nguyen-10	$2\sin(x)\cos(y)$	100%	91%	99%	76%	64%	–
Nguyen-11	x^y	100%	100%	100%	7%	100%	–
Nguyen-12	$x^4 - x^3 + \frac{1}{2}y^2 - y$	0%	0%	0%	0%	0%	–
	Average	**83.6%**	75.2%	46.7%	60.1%	73.9%	–

Symbolic Regression via Neural-Guided GP Population Seeding (NIPS 2021)

Methodology:

- The neural network generates the initial population.
- GP evolves the population through mutation (P_m) and crossover (P_c).



Symbolic Regression via Neural-Guided GP Population Seeding (NIPS 2021)

	Recovery rate (%)			
	All	Nguyen	R	Livermore
Ours	**74.92**	**92.33**	**33.33**	**71.09**
GEGL [Ahn et al., 2020]	64.11	86.00	**33.33**	56.36
Random Restart GP (i.e. GP only)	63.57	88.67	2.67	58.18
DSR (i.e. RNN only) [Petersen et al., 2021]	45.19	83.58	0.00	30.41
95% confidence interval	±1.54	±1.76	±2.81	±1.32

Proposed Unified Framework:

- Integrates five key strategies:
  - Recursive problem simplification (AI Feynman).
  - Neural-guided search (Deep Symbolic Regression).
  - Large-scale pre-training (NeSymReS).
  - Evolutionary search (Genetic Programming).
  - Sparse linear models (SINDy).

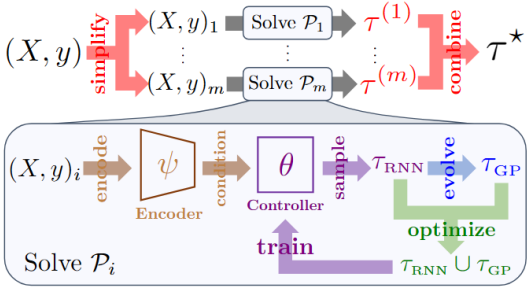
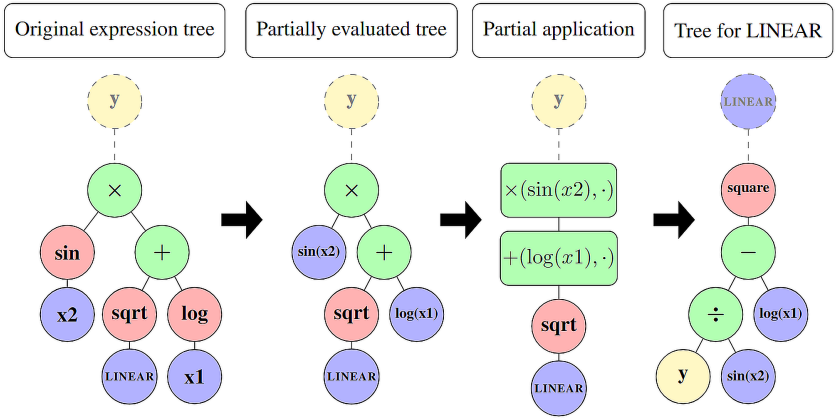
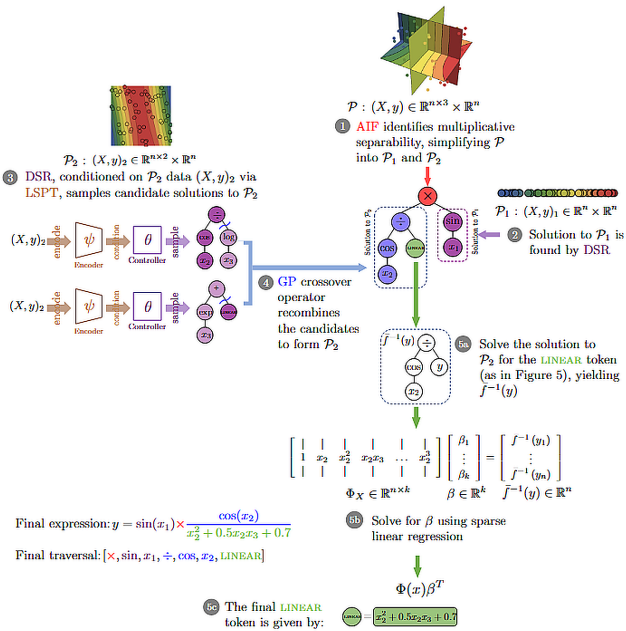


Figure 1: Unified deep symbolic regression. The five integrated solution strategies are color-coded: AIF, DSR, LSPT, GP, LM.

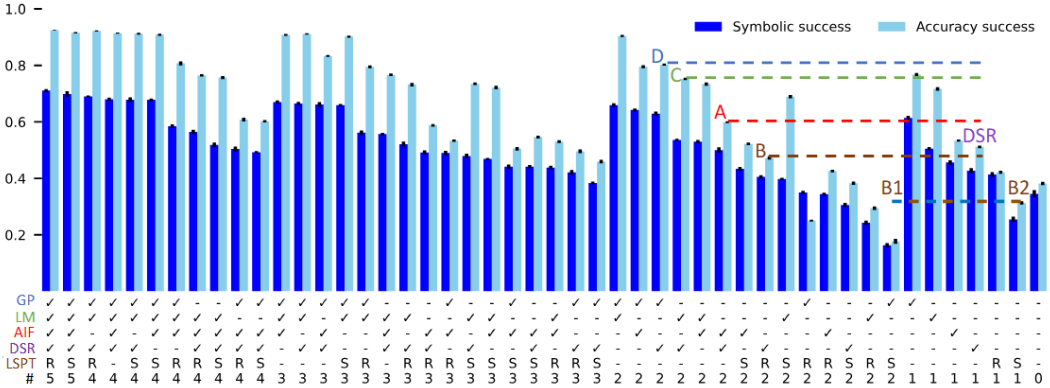
Unified Deep Symbolic Regression (NIPS 2022)



Unified Deep Symbolic Regression (NIPS 2022)



Unified Deep Symbolic Regression (NIPS 2022)



D: GP+DSR, DSR: Deep Symbolic Regression

Language Model-Based Symbolic Regression

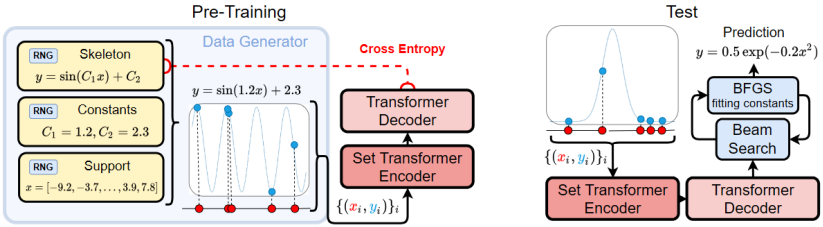
Methodology:

- Transformer-based architecture:
  - **Encoder:** Maps $\{(x_i, y_i)\}$ to a latent space z.
  - **Decoder:** Generates a skeleton e without numerical constants:

$$P(e_k|e_{<k}, z) = \text{softmax}(W_k h_k + b_k)$$

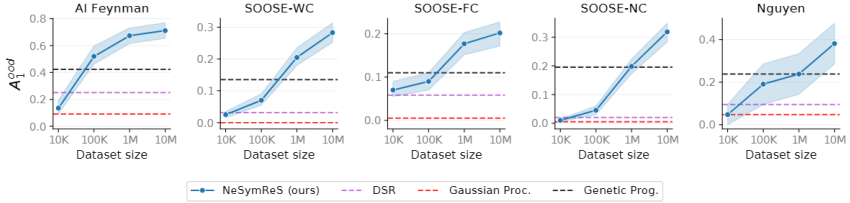
- Numerical constants are fit post-decoding using optimization methods (e.g., BFGS).
- Training uses a cross-entropy loss on skeletons:

$$L = -\sum_k \log P_\theta(e_k|e_{<k}, X, Y)$$



Key Results:

- A_{ood} accuracy on AI-Feynman: 82.5% (NeSymReS) vs. 65.3% (Deep Symbolic Regression).
- Efficient scaling with larger datasets (up to 10M equations) and test-time data points.
- Robust against unseen equation structures (e.g., strictly out-of-sample skeletons, SOOSE).
- Limitations:
  - Supports up to $D = 3$ input dimensions.
  - Different instances of the same skeleton can represent vastly different functions depending on constants (e.g., $\sin(10x)$ vs. $\sin(0.1x)$).



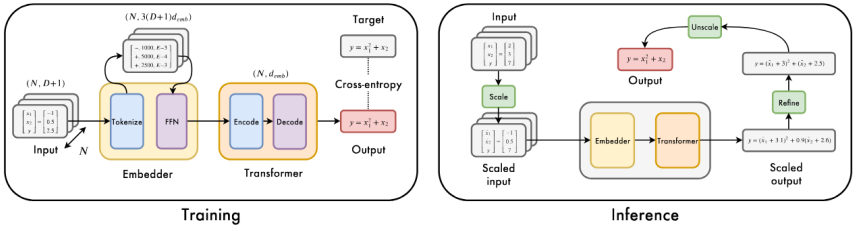
End-to-End Symbolic Regression with Transformers (NIPS 2022)

Methodology:

- End-to-end approach: Train Transformers to predict full expressions, including constants.
- Use symbolic-numeric tokenization:

$$f(x) = \sin(10x) \quad \Rightarrow \quad [\sin, *, 10, x]$$

- Refinement: Fine-tune predicted constants via BFGS with informed initialization.
- Scalable to larger input dimensions ($D = 10$) and datasets.

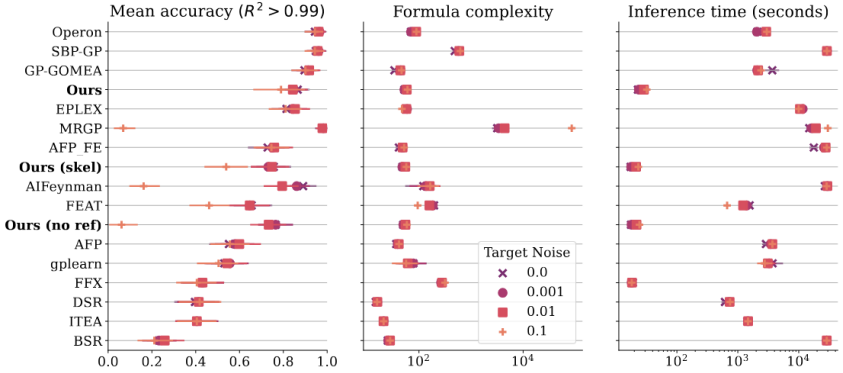


Key Results:

- Outperforms skeleton-based approaches in symbolic regression benchmarks:

$$R^2 = 0.68 \quad \text{(E2E with refinement)} \quad \text{vs.} \quad R^2 = 0.43 \quad \text{(Skeleton-based)}$$

- Significant inference speed-up compared to genetic programming.



Transformer-Based Symbolic Regression with Joint Supervision (ICLR 2023)

Motivation:

- Ill-posed problem: Different instances of the same skeleton behave differently.
- Poor feature extraction by previous encoders.

Proposed Method:

- **Feature Extraction:** Use a pure residual MLP framework:

$$O_i = \mathrm{POS}\left(\mathrm{MaxPool}\left(\mathrm{PRE}(f_{i,j}), j \in K\right)\right)$$

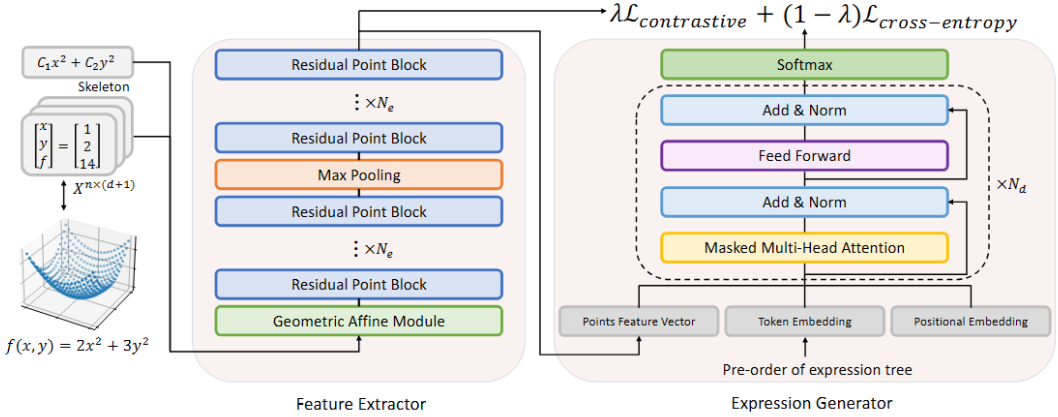
- **Joint Learning:** Combine Cross-Entropy (CE) and Contrastive Learning (CL) losses:

$$L = (1-\lambda)L_{\mathrm{CE}} + \lambda L_{\mathrm{CL}}$$

- **Expression Generation:** Transformer-based autoregressive decoder:

$$P(\tau|v) = \prod_{i=1}^{|\tau|} P(\tau_i|\tau_{<i}, v)$$

Transformer-Based Symbolic Regression with Joint Supervision



Results

	Ours	SymbolicGPT	NeSymReS	DSO	GP
Benchmark	$R^2 \uparrow$	$R^2 \uparrow$	$R^2 \uparrow$	$R^2 \uparrow$	$R^2 \uparrow$
Nguyen	**0.99999**	0.64394	0.97538	0.99489	0.89019
Constant	**0.99998**	0.69433	0.84935	0.99927	0.90842
Keijzer	**0.98320**	0.59457	0.97500	0.96928	0.90082
R	**0.99999**	0.71093	0.99993	0.97298	0.83198
AI-Feynman	**0.99999**	0.64682	**0.99999**	**0.99999**	0.92242
SSDNC	**0.94782**	0.74585	0.85792	0.93198	0.88913
Overall avg.	**0.98850**	0.67274	0.94292	0.97806	0.89049

Limitations of Language Model-Based Symbolic Regression

Challenges:

- Symbolic regression is an **NP-hard problem**.
- Language models alone are unlikely to efficiently solve NP-hard problems.

Solutions:

- **Combine pre-trained language models with optimization techniques**.
- Transformer-Based Planning for Symbolic Regression (TPSR).

Methodology:

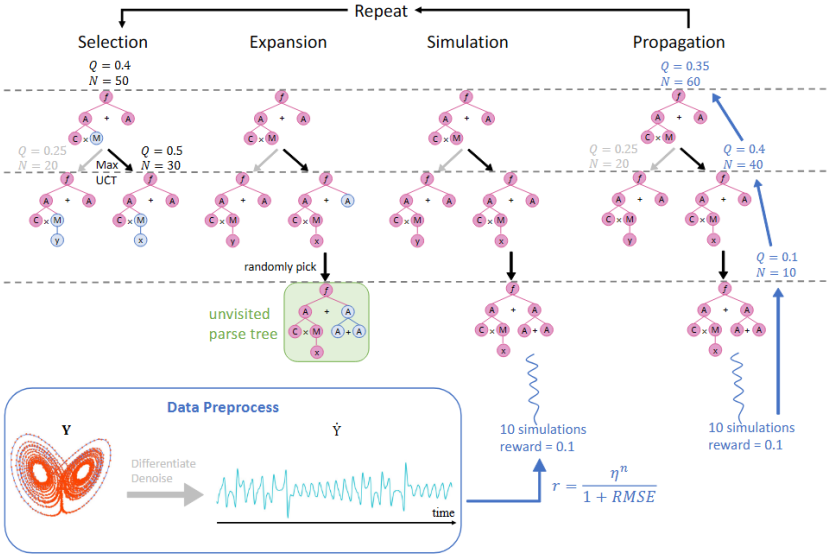
- Representation: Mathematical expressions are encoded as parse trees.
- MCTS Reward Function:

$$r = \frac{\eta^n}{1 + \sqrt{\frac{1}{N}\|Y_i - \tilde{f}(Y)\|^2}}$$

  where η: parsimony factor, n: production rules count, N: number of data points.
- Upper Confidence Bound for Trees (UCT):

$$\mathrm{UCT}(s,a) = Q(s,a) + c\sqrt{\frac{\ln N(s)}{N(s,a)}}$$

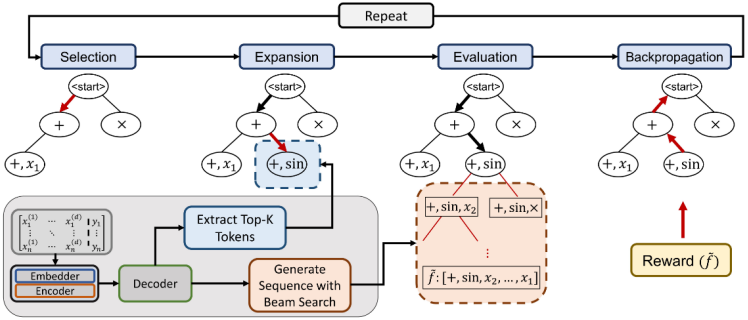
  - $Q(s,a)$: Estimated reward for taking action a in state s.
  - $N(s)$: Visit count for state s.
  - $N(s,a)$: Visit count for action a at state s.
  - c: Exploration coefficient controlling the balance between exploration and exploitation.


Repeat
Selection
Expansion
Simulation
Propagation
Q = 0.4
N = 50
Q = 0.25
N = 20
Max
UCT
Q = 0.5
N = 30
randomly pick
unvisited
parse tree
Q = 0.35
N = 60
Q = 0.25
N = 20
Q = 0.4
N = 40
Q = 0.1
N = 10
10 simulations
reward = 0.1
10 simulations
reward = 0.1
Data Preprocess
Y
Ẏ
Differentiate
Denoise
time
r = η^n / (1 + RMSE)

Transformer-Based Planning for Symbolic Regression (NIPS 2023)

TPSR Objective:

- Introduce Monte Carlo Tree Search (MCTS) into Transformer decoding to guide equation generation.
- Leverage pre-trained Transformer priors while incorporating feedback on:
  - Fitting accuracy.
  - Model complexity.



Methodology:

- Transformer-based SR model enhanced with MCTS:
  - s: Current state (partial equation sequence) in the Monte Carlo Tree Search (MCTS).
  - a: Action (next token) extending s to a new state s'.
  - $Q(s, a)$: Estimated quality (expected reward) of taking action a in state s.
  - β: Hyperparameter balancing exploration and exploitation.
  - $P_\theta(a|s)$: Probability of selecting action a in state s, predicted by the pre-trained Transformer model.
  - $N(s)$: Visit count for state s (times s has been explored).
  - $N(s')$: Visit count for the next state s', resulting from action a.

$$P\text{-UCB}(s, a) = Q(s, a) + \beta \cdot P_\theta(a|s) \cdot \sqrt{\frac{\ln(N(s))}{1 + N(s')}}$$

P-UCB(s, a)

$$P\text{-UCB}(s,a) = Q(s,a) + \beta \cdot P_\theta(a|s) \cdot \sqrt{\frac{\ln(N(s))}{1+N(s')}}$$

Key Components:

- **Exploit ($Q(s,a)$):** Prioritize high-quality actions.
- **Explore ($\sqrt{\frac{\ln(N(s))}{1+N(s')}}$):** Encourage actions leading to less-visited states.
- **Prior ($P_\theta(a|s)$):** Leverage pre-trained Transformer knowledge to bias promising actions.

Reward Function

Reward function combining accuracy and complexity:

$$r(f) = \frac{1}{1 + \text{NMSE}} + \lambda e^{-\frac{\ell(f)}{L}}$$

- NMSE: Normalized Mean Squared Error between predicted and actual values.
- $\ell(f)$: Length of the generated equation sequence.
- L: Maximum allowable sequence length.

SNIP: Bridging Symbolic and Numeric Domains (ICLR 2024)

SNIP: A pre-training framework for joint symbolic-numeric learning using contrastive loss.

- Dual-Encoder Architecture:

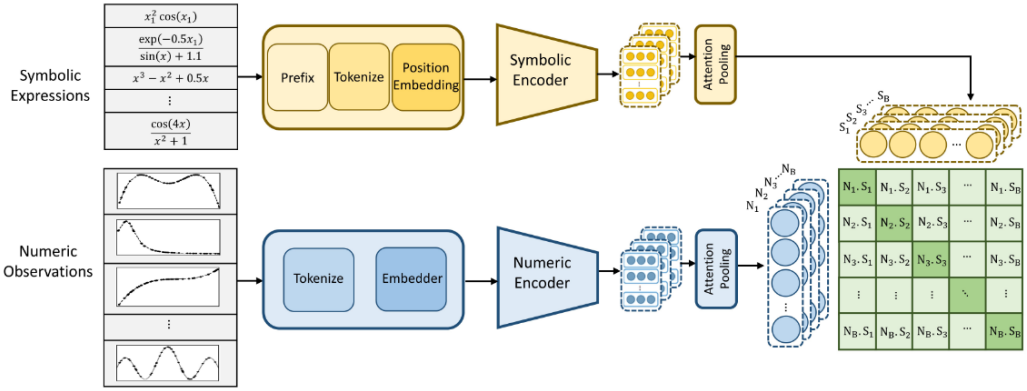
$$\text{Numeric Encoder: } Z_V = A_V \cdot V_{LV}, \quad \text{Symbolic Encoder: } Z_S = A_S \cdot S_{LS}$$

- Pre-training Objective:

$$L = -\sum_{(v,s)\in B} \left(\log \text{NCE}(Z_S, Z_V) + \log \text{NCE}(Z_V, Z_S)\right)$$

  where NCE aligns symbolic Z_S and numeric Z_V embeddings.
- Applications: Property prediction, symbolic regression via latent space optimization.

SNIP: Bridging Symbolic and Numeric Domains (ICLR 2024)



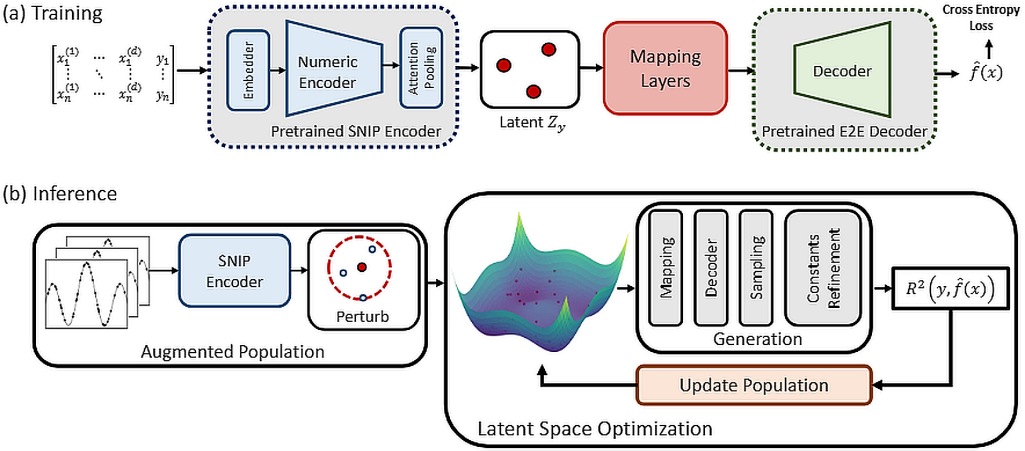
SNIP in SR Pipeline:

- Numeric encoder produces embeddings (Z_V).
- Integrated with a symbolic expression generation decoder (G_ω) via a mapping network.
- Training Objective: $L(f, f) = -\frac{1}{|f|} \sum_j \log P(t_j | t_1, \ldots, t_{j-1}; G_\omega)$

Latent Space Optimization (LSO):

- Explore equations in the learned embedding space using optimization.
- Refine constant values for equations via numerical solvers.

SNIP: Bridging Symbolic and Numeric Domains (ICLR 2024)



Sparse Learning-Based Symbolic Regression

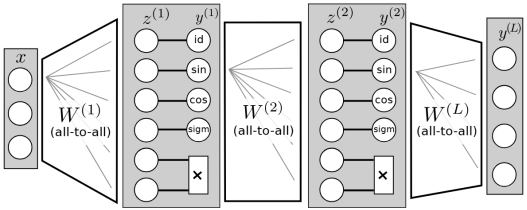
Equation Learner (EQL):

- EQL uses a feed-forward architecture with specialized units:

$$z^{(l)} = W^{(l)} y^{(l-1)} + b^{(l)}$$

$$y^{(l)} = (f_1(z_1), \ldots, f_u(z_u), g_1(z_{u+1}, z_{u+2}), \ldots)$$

- Unary Functions (f_i): $\mathrm{id}, \sin, \cos,$ sigmoid.
- Binary Functions (g_j): Multiplication of inputs.
- The final output is produced by a linear regression layer.



Training and Loss Function:

- Loss function with regularization:

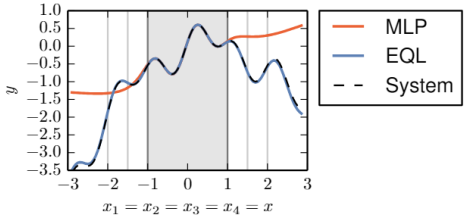
$$L(D) = \frac{1}{N} \sum_{i=1}^{N} \|\psi(x_i) - y_i\|^2 + \lambda \sum_{l=1}^{L} \|W^{(l)}\|_1$$

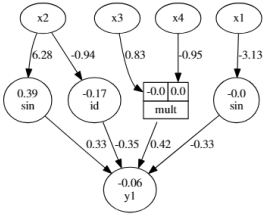
- Hybrid regularization: L1 for sparsity and L0 for fine-tuning.
- Regularization Strategy:
  - **Initial Phase ($t < t_1$):** No regularization ($\lambda = 0$), allowing free parameter exploration.
  - **Middle Phase ($t_1 \leq t \leq t_2$):** Apply L_1-regularization to encourage sparsity.
  - **Final Phase ($t > t_2$):** Disable L_1-regularization and fine-tune using L_0-constraints.
- Transition Points:

$$t_1 = \frac{1}{4}T, \quad t_2 = \frac{19}{20}T$$

EQL Results

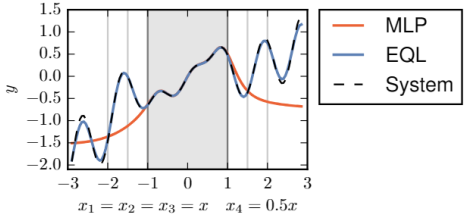
(a) F-1

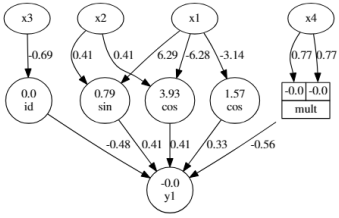




learned formula: $-0.33\sin(-3.13x_1) + 0.33\sin(6.28x_2 + 0.39) + 0.33x_2 - 0.056 - 0.33x_3x_4$

(b) F-2



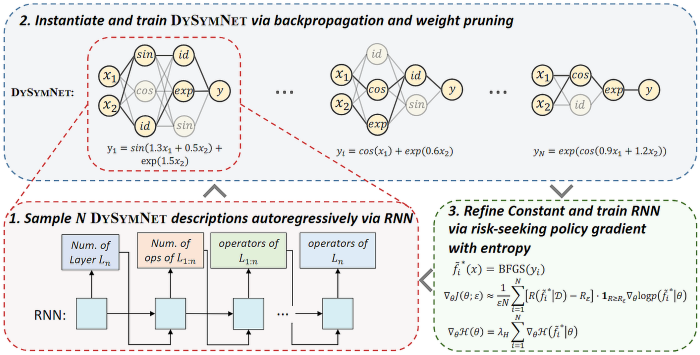


learned formula: $0.33\cos(3.14x_1 + 1.57) + 0.33x_3 - 0.33x_4^2 +$
$0.41\cos(-6.28x_1 + 3.93 + 0.41x_2) + 0.41\sin(6.29x_1 + 0.79 + 0.41x_2)$

DYSYMNET: Neural-Guided Dynamic Symbolic Regression (ICML 2024)

- **DYSYMNET**: A novel SR framework based on reinforcement learning (policy gradient) for architecture search.
- **Policy Gradient Objective for Risk-Seeking Optimization:**

$$J_{\text{risk}}(\theta_c; \epsilon) = \mathbb{E}[R(f^*) \mid R(f^*) \geq R_\epsilon(\theta_c)].$$



DYSYMNET RESULTS

Data Group	Benchmark	DYSYMNET $R^2 \uparrow$	uDSR $R^2 \uparrow$	NGGP $R^2 \uparrow$	NeSymReS $R^2 \uparrow$	EQL $R^2 \uparrow$
Standard ($d \leq 2$)	Nguyen	**0.9999**$_{\pm 0.0001}$	0.9989$_{\pm 0.0000}$	0.9848$_{\pm 0.0000}$	0.9221$_{\pm 0.0000}$	0.9758$_{\pm 0.0032}$
	Nguyen*	**0.9999**$_{\pm 0.0000}$	**0.9999**$_{\pm 0.0000}$	**0.9999**$_{\pm 0.0001}$	0.3523$_{\pm 0.0000}$	0.7796$_{\pm 0.0019}$
	Constant	**0.9992**$_{\pm 0.0000}$	0.9989$_{\pm 0.0002}$	0.9989$_{\pm 0.0002}$	0.7670$_{\pm 0.0000}$	0.9817$_{\pm 0.0088}$
	Keijzer	**0.9986**$_{\pm 0.0002}$	0.9968$_{\pm 0.0000}$	0.9912$_{\pm 0.0005}$	0.9358$_{\pm 0.0000}$	0.9446$_{\pm 0.0014}$
	Livermore	**0.9896**$_{\pm 0.0001}$	0.9670$_{\pm 0.0003}$	0.9665$_{\pm 0.0005}$	0.9195$_{\pm 0.0000}$	0.7496$_{\pm 0.0002}$
	R	0.9895$_{\pm 0.0012}$	**0.9999**$_{\pm 0.0000}$	**0.9999**$_{\pm 0.0000}$	0.9368$_{\pm 0.0000}$	0.8588$_{\pm 0.0061}$
	Jin	**1.0000**$_{\pm 0.0000}$	0.9864$_{\pm 0.0002}$	0.9942$_{\pm 0.0001}$	0.9987$_{\pm 0.0000}$	0.9921$_{\pm 0.0002}$
	Koza	**1.0000**$_{\pm 0.0000}$	0.9965$_{\pm 0.0000}$	0.9999$_{\pm 0.0000}$	0.9999$_{\pm 0.0000}$	0.7571$_{\pm 0.0022}$
Data Group	Benchmark	$R^2 \uparrow$	$R^2 \uparrow$	$R^2 \uparrow$	$R^2 \uparrow$	$R^2 \uparrow$
SRBench ($d \geq 2$)	Feynman	**0.9931**$_{\pm 0.0015}$	0.9806$_{\pm 0.0003}$	0.9190$_{\pm 0.0004}$	0.9234$_{\pm 0.0000}$	0.5641$_{\pm 0.0028}$
	Strogatz	**0.9968**$_{\pm 0.0031}$	0.9455$_{\pm 0.0003}$	0.9534$_{\pm 0.0003}$	0.8816$_{\pm 0.0000}$	0.6511$_{\pm 0.0012}$
	Black-box	**0.8908**$_{\pm 0.0028}$	0.6697$_{\pm 0.0010}$	0.6086$_{\pm 0.0021}$	0.4226$_{\pm 0.0000}$	0.4528$_{\pm 0.0107}$

Thanks for listening!

Email: Hengzhe.zhang@ecs.vuw.ac.nz

GitHub Project: https://github.com/hengzhe-zhang/EvolutionaryForest/