LITERATURE REVIEW:

DEEP SYMBOLIC REGRESSION: RECOVERING MATHEMATICAL EXPRESSION FROM DATA VIA RISK-SEEKING POLICY GRADIENTS

NANCY LYU 02/10/2022

OUTLINE

- Motivation
- Symbolic regression
- Method
- Results .

MOTIVATION

- Understanding the underlying mathematical relationships among variables describing a dataset is a major task in the scientific process.[1]

[1] Brenden K. Petersen, Mikel Landajuela Larma, T. Nathan Mundhenk, Claudio P. Santiago, Soo K. Kim, Joanne T. Kim, Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients.

SYMBOLIC REGRESSION

- What is symbolic regression?
- Traditional approaches to symbolic regression vs. Deep symbolic regression

WHAT IS SYMBOLIC REGRESSION?

- Symbolic regression is the process of identifying mathematical expressions that fit observed output from a black-box process. It is a discrete optimization problem generally believed to be NP-hard. [1]

[1] T. Nathan Mundhenk, Mikel Landajuela, Ruben Glatt, Claudio P. Santiago, Daniel M. Faissol, Brenden K. Petersen, Symbolic Regression via Neural-Guided Genetic
Programming Population Seeding.

TRADITIONAL APPROACHES TO SYMBOLIC REGRESSION VS. DEEP SYMBOLIC REGRESSION

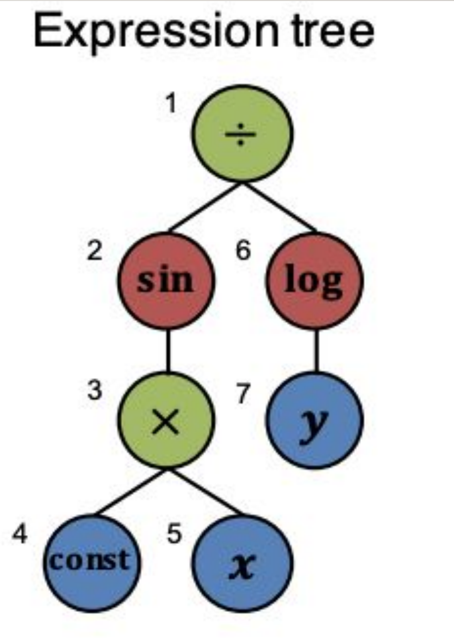
- The space of mathematical expressions is discrete (in model structure) and continuous (in model parameters).
- Traditional approaches to symbolic regression using evolutionary algorithms. In particular, genetic programming (GP) (Koza, 1992; Schmidt & Lipson, 2009; Back et al., 2018).
- GP exhibits high sensitivity to hyperparameters and scale poorly to larger problems.

TRADITIONAL APPROACHES TO SYMBOLIC REGRESSION VS. DEEP SYMBOLIC REGRESSION

- DSR framework: use a large model (i.e. neural network) to search the space of small models (i.e. symbolic expressions).

METHODS - Generate symbolic expression tree with RNN

- Binary tree
- Internal nodes are mathematical operators
- Terminal nodes are input variables or constants
- Pre-order traversal
- Uniqueness



Given a dataset (X, y), $X_i \in R^n$, $y_i \in R$.

$f : R^n \to R$

$L = \{+, -, \times, \div, \log, \sin, \cos, x, \text{const}\}$

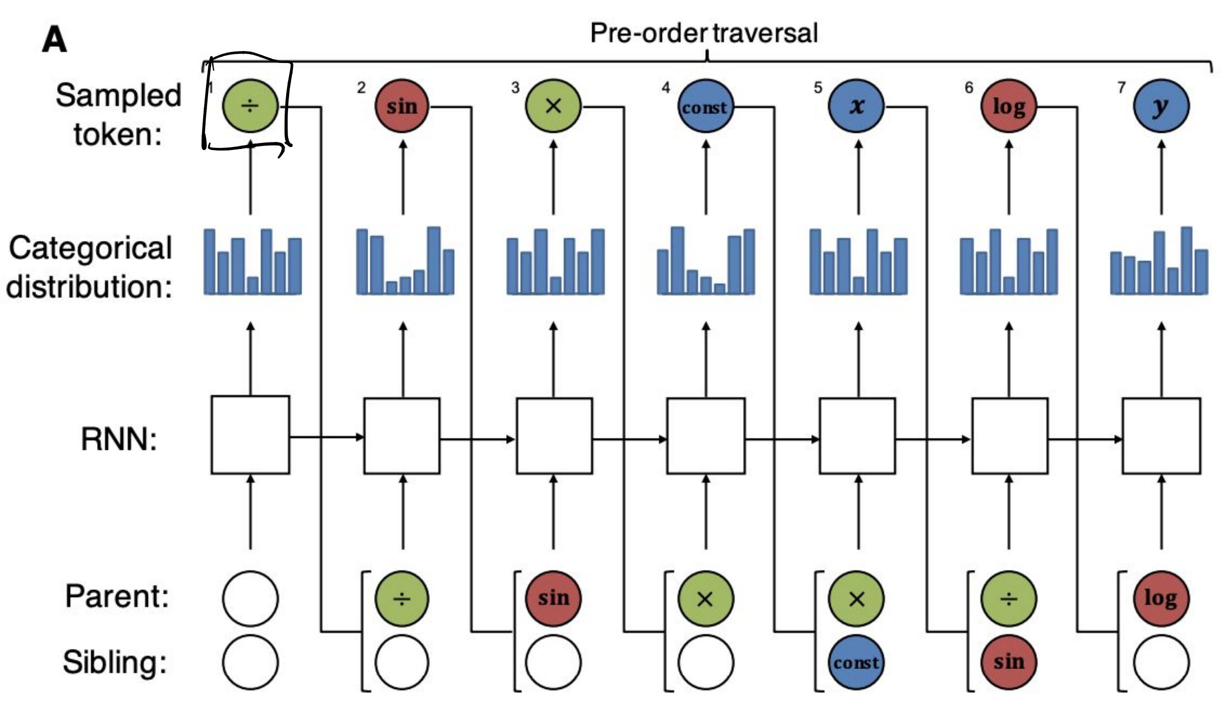
$\boxed{\sin(c\,\cancel{x}\,x_1) / \log \cancel{y}\,x_2}$

METHODS - Generate symbolic expression tree with RNN

- RNN emits a categorical distribution with parameter ψ over tokens.
- Ψ defines probabilities of selecting each token from library L.

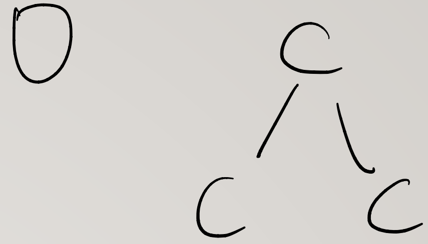
$$p(\tau_i|\tau_{1:(i-1)};\theta) = \psi^{(i)}_{\mathcal{L}(\tau_i)}$$

$$p(\tau|\theta) = \prod_{i=1}^{|\tau|} p(\tau_i|\tau_{1:(i-1)};\theta) = \prod_{i=1}^{|\tau|} \psi^{(i)}_{\mathcal{L}(\tau_i)}$$



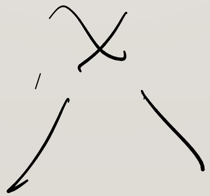
Providing hierarchical inputs to the RNN

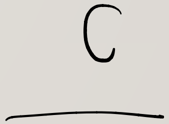
- Parent and sibling nodes
- Empty token for node does not have a parent or sibling



Constraining the search space

- Expressions are limited to a pre-specified minimum and maximum length.
- The children of an operator should not all be constants, as the result would simply be a different constant.
- The child of a unary operator should not be the inverse of that operator, e.g. log(exp(x)) is not allowed.
- Descendants of trigonometric operators should not be trigonometric operators, e.g. sin(x + cos(x))





Reward function

- Normalized root-mean-square error (NRMSE)
- Given a dataset (X, y) of size n and candidate expression f:

$$\text{NRMSE} = \frac{1}{\sigma_y}\sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - f(X_i)\right)^2} \simeq 0$$

$$R(\tau) = \frac{1}{1+NRMSE}$$

METHODS - training the RNN using policy gradients

- Using reinforcement learning to train the RNN to produce better-fitting expressions.
- $p(\tau|\theta)$ is like a policy
- Sampled tokens are like actions
- Standard policy gradient vs. Risk-seeking policy gradient

Standard policy gradient

- The expectation of a reward function R(τ) under expressions from the policy, $J_{std}(\theta)$.
- The standard REINFORCE policy gradient (Williams, 1992) can be used to maximize this expectation via gradient ascent:

N

$$\begin{aligned}\nabla_\theta J_{\text{std}}(\theta) &= \nabla_\theta \mathbb{E}_{\tau \sim p(\tau|\theta)}\left[R(\tau)\right] \\ &= \mathbb{E}_{\tau \sim p(\tau|\theta)}\left[R(\tau)\nabla_\theta \log p(\tau|\theta)\right]\end{aligned}$$

- Unbiased estimate of $\nabla_\theta J_{std}(\theta)$:

$$\nabla_\theta J_{\text{std}}(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} R(\tau^{(i)})\nabla_\theta \log p(\tau^{(i)}|\theta)$$

Risk-seeking policy gradient

- Optimizing the average performance or maximizing the best-case performance?
- Define $R_\varepsilon(\theta)$ as the $(1 - \varepsilon)$-quantile of the distribution of rewards under the current policy.

$$J_{\text{risk}}(\theta; \varepsilon) \doteq \mathbb{E}_{\tau \sim p(\tau|\theta)} \left[R(\tau) \mid R(\tau) \geq R_\varepsilon(\theta) \right]$$

ε

Risk-seeking policy gradient

- How to estimate this objective via Monte Carlo sampling?
- **Proposition 1**. *Let J_{risk} $(\theta; \varepsilon)$ denote the conditional expectation of rewards above the $(1-\varepsilon)$-quantile, as in Equation (1). Then the gradient of J_{risk} $(\theta; \varepsilon)$ is given by:*

$$\nabla_\theta J_{risk}(\theta;\varepsilon) = \mathbb{E}_{\tau\sim p(\tau|\theta)}[(R(\tau) - R_\varepsilon(\theta)) \cdot \nabla_\theta \log p(\tau|\theta) \mid R(\tau) \geq R_\varepsilon(\theta)]$$

$\tilde{R}_\varepsilon(\theta)$

Risk-seeking policy gradient

- A simple Monte Carlo estimate of the gradient from a batch of N samples:

$$\nabla_\theta J_{\text{risk}}(\theta;\varepsilon) \approx \frac{1}{\varepsilon N}\sum_{i=1}^{N}\left[R(\tau^{(i)}) - \tilde{R}_\varepsilon(\theta)\right]\cdot \mathbf{1}_{R(\tau^{(i)})\geq \tilde{R}_\varepsilon(\theta)}\nabla_\theta \log p(\tau^{(i)}|\theta).$$

- Two differences from the standard reinforcement MC estimate:

1. it suggests a specific baseline, Rε(θ), whereas the baseline for standard policy gradients is non-specific, chosen by the user;.
2. Only the top ε fraction of samples from each batch are used in the gradient computation.

RESULTS

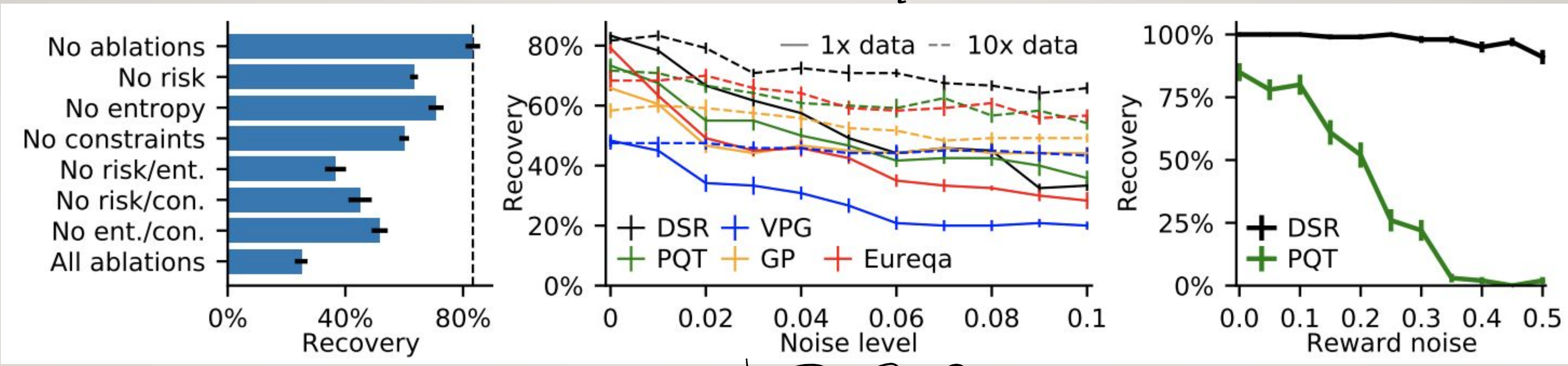
Benchmark	Expression	DSR	PQT	VPG	GP	Eureqa	Wolfram
Nguyen-1	$x^3 + x^2 + x$	100%	100%	96%	100%	100%	100%
Nguyen-2	$x^4 + x^3 + x^2 + x$	100%	99%	47%	97%	100%	100%
Nguyen-3	$x^5 + x^4 + x^3 + x^2 + x$	100%	86%	4%	100%	95%	100%
Nguyen-4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	100%	93%	1%	100%	70%	100%
Nguyen-5	$\sin(x^2)\cos(x) - 1$	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%	–

RESULTS

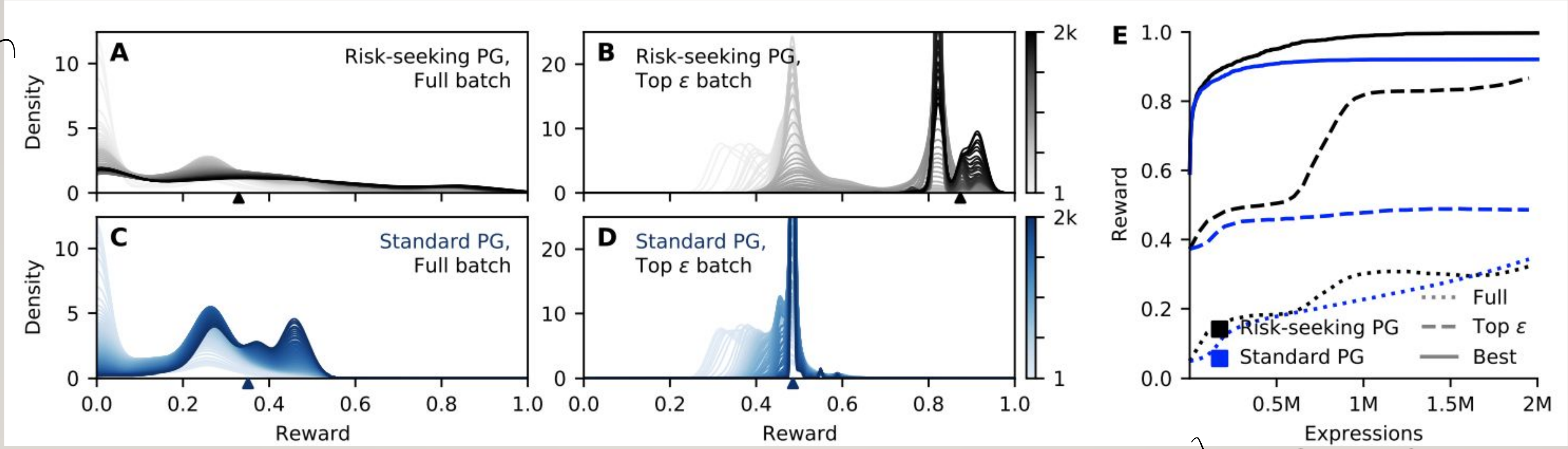
$R'(\tau) = R(\tau) + N(0, \sigma)$

SymPy

Recovery vs dataset noise



RESULTS



A
Risk-seeking PG,
Full batch
B
Risk-seeking PG,
Top ε batch
C
Standard PG,
Full batch
D
Standard PG,
Top ε batch
E
Density
Reward
Expressions
2k
1
0.5M
1M
1.5M
2M
Risk-seeking PG
Standard PG
Full
Top ε
Best