Abstract—There are various ways to use machine learning to improve data storage techniques. In this paper, we study symbolic regression, a machine-learning method for recovering the symbolic form of a function from its samples. We present a new symbolic regression scheme that utilizes side information for higher accuracy and speed in function recovery. The scheme enhances latest results on symbolic regression that were based on recurrent neural networks and genetic programming. The scheme is tested on a new benchmark of functions for data storage.

I. INTRODUCTION

There are various approaches of using machine learning to improve data storage techniques. This work studies symbolic regression, a machine-learning method for recovering the symbolic form of a function from its samples. Specifically, let \( x = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^n \) be \( n \) variables. Let \( y = f(x) \in \mathbb{R} \) be a function of \( x \). Let \( S = \{(X_i, y_i) | i = 1, 2, \cdots, m\} \) be \( m \) samples of the function \( f \), where each point \( X_i \in \mathbb{R}^n \) is a sample of \( x \), and \( y_i = f(X_i) \) is the corresponding value of \( y \). Given such a dataset of samples \( S \), the goal of symbolic regression is to recover the symbolic form of the function \( f \) (such as \( y = -x_1 \log_2 x_1 - (1 - x_1) \log_2 (1 - x_1) \) or \( y = \sqrt{(x_1 - x_2)^2 + (x_3 - x_4)^2} \)). The recovered function should fit the samples well, and also be as simple as possible.

Symbolic regression can have numerous applications to data storage. One application is modeling physical storage devices, such as non-volatile flash memories, where it can provide functional insights into evolving cell threshold voltage distributions, spatial inter-cell interference effects, and page-level bit error counts as the memory ages over read/write cycles or loses charge with infrequent use. Symbolic regression can bridge experimental data with theoretical models. Another application is aiding the analysis of storage schemes, such as modeling the performance of error-correcting codes, wear-leveling schemes, etc. Here it can bridge simulation data with theoretical models and provide insights for further analysis.

Symbolic regression is a challenging problem in AI. The number of functions in the search space grows exponentially as the lengths of the functions increase. Existing symbolic-regression methods mainly use evolutionary algorithms or deep learning [3]. In particular, a recent method [1] (which we shall call DSR-GP) improves the Deep Symbolic Regression (DSR) approach [2] and achieves state-of-the-art performance by combining deep neural networks with genetic programming. In this work, we introduce a new symbolic regression approach that further improves DSR-GP by combining it with side information. Broadly speaking, side information can refer to any information that correlates with the ground-truth function \( f \), although in this paper, we shall focus on a narrower type: functions that resemble all or a part of \( f \). We show that the new scheme, which we shall call DSR-GP-SI, can notably improve the performance of symbolic regression.

II. SYMBOLIC REGRESSION USING SIDE INFORMATION

To represent symbolic functions, we need a set of operators \( S_{\text{op}} \), a set of variables \( S_{\text{var}} \), and a set of values for coefficients \( S_{\text{coeff}} \). For example, \( S_{\text{op}} = \{+,-,\times,\div,\sin,\cos,\log,\exp\} \), \( S_{\text{var}} = \{x_1, x_2\} \) and \( S_{\text{coeff}} = \mathbb{R} \), then \( \sin(x_1 - x_2) + 1.5 \exp(x_1) \) is a valid symbolic function. If we wish to represent all coefficients using a placeholder “\( \text{const} \)”, we can let \( S_{\text{coeff}} = \{\text{const}\} \). The three sets \( S_{\text{op}}, S_{\text{var}} \) and \( S_{\text{coeff}} \) together form the token library. A symbolic function can be represented by its Polish notation, which is a depth-first traversal of the symbols in the computation tree (where leaves are variables or constants, and internal nodes are operators) of the symbolic function. For example, for \( f(x_1, x_2) = \frac{x_1^{x_2}}{x_2 + 1}, \log x_2 - \sin(2x_1) \cos x_2, \) its Polish notation is \( //x_1^x_2/1x_2+1\times\log x_2\sin(2x_1)\cos(\times\log x_2\sin(2x_1)\cos x_2)\).

Let \( y = f(x) \) be the ground-truth function we look for. Let \( G_f \) be its computation tree. This work considers a function \( f_{SI} \) to be side information (SI) for \( f \) if the computation tree of \( f_{SI} \), \( G_{f_{SI}} \), can be obtained from a subtree of \( G_f \), \( G_{sub} \), where some subtrees in \( G_{sub} \) can be replaced by the “\( \text{const} \)” tokens. (For discussions on how side information can be obtained in practice, and how to extend the above SI to more general cases, please see the full paper [4].) The similarity between \( f \) and \( f_{SI}, \gamma(f,f_{SI}) \), can be measured as follows. Let \( L(f) \) and \( L(f_{SI}) \) be the number of symbols in \( f \) and \( f_{SI} \), respectively. Then \( \gamma(f,f_{SI}) = L(f_{SI})/L(f) \in [0, 1] \).

We present a new symbolic regression (SR) scheme DSR-GP-SI, which extends DSR [2] and DSR-GP [1] by incorporating side information. It searches for a function \( f \) that not only fits the samples in \( S \) well, but also is close to the provided side-information function \( f_{SI} \). To define the
“fitness” of $\hat{f}$, $F_{S,SI}(\hat{f})$, let us first define several notations: (1) Let $\mu_y = \frac{1}{|S|} \sum_{i=1}^{|S|} y_i$ and $\sigma_y = \sqrt{\frac{1}{|S|} \sum_{i=1}^{|S|} (y_i - \mu_y)^2}$, then the “normalized root-mean-square error” $\text{NRMSE}(\hat{f}) \equiv \frac{1}{|S|} \sum_{i=1}^{|S|} (\hat{f}(X_i) - y_i)^2$; (2) let $d_{lev}(\hat{f}, f_{SI})$ be the Levenshtein distance between the Polish notations of $\hat{f}$ and $f_{SI}$, then the “pre-order representation distance” $\text{PORD}(\hat{f}, f_{SI}) \equiv \frac{d_{lev}(\hat{f}, f_{SI})}{d_{lev}(f_{SI})}$; (3) let $\delta > 0$ be a constant parameter, then the “normalized distance to side-information function” $\text{NDSIF}(\hat{f}, f_{SI}) \equiv \frac{1}{|S|} \sum_{i=1}^{|S|} |\hat{f}(X_i) - f_{SI}(X_i)| + \delta |\hat{f}(X_i)|$. Then $F_{S,SI}(\hat{f}) \equiv \frac{\text{NRMSE}(\hat{f})}{w_1} + \frac{\text{PORD}(\hat{f}, f_{SI})}{w_2} + \frac{\text{NDSIF}(\hat{f}, f_{SI})}{w_3}$, where $w_1, w_2, w_3$ are constant parameters. The greater $F_{S,SI}(\hat{f})$ is, the better $\hat{f}$ is considered to be.

The DSR-GP-SI scheme is illustrated in Fig. 1. It searches for good functions in iterations, and it ends either when a found function matches the samples in $S$ sufficiently well, or when it exceeds a time budget. In each iteration, the scheme works as follows: (1) Let $f_{SI}$ be the given side-information function. We use a recurrent neural network (RNN) $\text{RNN}_{\text{enc}}$ to transform it to a latent vector $c_{SI}$, and use $c_{SI}$ to initialize the memory of another RNN $\text{RNN}_{\text{dec}}$. We then use $\text{RNN}_{\text{dec}}$ to generate $N_{\text{RNN}}$ new functions in an auto-regressive way (by generating the symbols in the Polish notation one by one), and select from them $n_{\text{RNN}}$ functions of the highest “fitness”. Let $F_{\text{RNN}}$ denote those $n_{\text{RNN}}$ functions. (2) Let $T_{GP}^0 = F_{\text{RNN}} \cup \{f_{SI}\}$ be the initial population for the genetic programming (GP) component. Use $\alpha$ generations (of GP operations including mutation, crossover and selection) to generate a new population $T_{GP}^\alpha$, and select from it $n_{GP}$ functions of the highest “fitness”. Let $F_{\text{GP}}$ denote those $n_{GP}$ functions. (3) Feed the functions in $F_{\text{RNN}} \cup F_{\text{GP}}$ to a persistent maximum reward priority queue (MRPQ), which saves the $n_{\text{RNN}}$ functions of the highest fitness it has seen. Then, the $n_{\text{RNN}}$ functions in MRPQ (denoted by $F_{\text{RNN}}$) are used as sample functions to train the two RNNs $\text{RNN}_{\text{enc}}$ and $\text{RNN}_{\text{dec}}$ together using supervised learning, so that they can learn to generate new functions that are similarly good.

We evaluate the performance of DSR-GP-SI, and compare it to DSR [2] and DSR-GP [1], which the DSR-GP-SI scheme evolved from. We compare their performance on two existing benchmarks: Nguyen [3] and Livermore [1]. To better evaluate the potential of symbolic regression for data storage, we also present a new benchmark of 20 functions (called the DS benchmark), shown in [4]. The performance of the three schemes for the DS benchmark is shown in Fig. 2. (Here the side-information functions are generated randomly. For details, see [4].) It can be seen that having side information can help improve the recovery rate (the fraction of times of recovering functions correctly) substantially; and generally, the more side information (measured by similarity between the ground-truth function and the SI function), the better. Furthermore, even when “similarity” is small, there often still exist (short) SI functions that can improve the recovery rate significantly. That can be seen from those gray circles near the top of the figure (whose recovery rates are close or equal to 1). Similar performance improvement by side information can be observed for the Nguyen and Livermore benchmarks [4].