Multi-Level Symbolic Regression: Function Structure Learning for Multi-Level Data

Kei Sen Fong
National University of Singapore

Mehul Motani
National University of Singapore

Abstract

Symbolic Regression (SR) is an approach which learns a closed-form function relating the predictors to the outcome in a dataset. Datasets are often multi-level (MuL), meaning that certain features can be used to split data into groups for analysis (we refer to these features as levels). The advantage of viewing datasets as MuL is that we can exploit the high similarity of data within a group. SR is well-suited for MuL datasets, in which the learnt function structure serves as 'shared information' between the groups while the learnt parameter values capture the unique relationships within each group. In this context, this paper makes three contributions: (i) We design an algorithm, Multi-level Symbolic Regression (MSR), which runs multiple parallel SR processes for each group and merges them to produce a single function structure. (ii) To tackle datasets that are not explicitly MuL, we develop a metric termed MLICC to select the best feature to serve as a level. (iii) We also release MSRBench, a database of MuL datasets (synthetic and real-world) which we developed and collated, that can be used to evaluate MSR. Our results and ablation studies demonstrate that MSR achieves a higher recovery rate and lower error on MSRBench compared to SOTA methods for SR and MuL datasets.

1 INTRODUCTION

Symbolic Regression (SR) is the task of learning a closed-formed expression to learn the relationship between features of a dataset, most frequently used in machine learning (ML) to learn the relationship between a set of input features to a single output variable (Koza, 1992). Consider, a dataset with input features, X, and output variable, y. SR algorithms aim to find $f^* = \arg\min_{f\in\mathcal{F}} L(f(X), y)$, where $\mathcal{F}$ and L are the search space of closed-form functions and the loss function respectively. More loosely, the practical task of SR is to find an f such $f(X) \approx y$. The obtained function tends to be more explainable than the black-box models used in traditional ML, making it a first-class algorithm in fields such as Physics (Udrescu and Tegmark, 2020) and Material Sciences (Wang et al., 2019; Sun et al., 2019). However, while there has been ample research on SR for general tabular datasets (La Cava et al., 2021), little to no work has been done to apply SR on multi-level (MuL) datasets.

Multi-level (MuL) datasets contain data in which certain features (called levels), usually categorical, are used to partition the dataset into smaller subsets (or groups) of data for analysis. These datasets are common in the real-world (Lee, 2022; Hamaker and Muthén, 2020). As an example of MuL data, consider a dataset of students' exam scores along with 3 predictors: time spent in school (x_1), number of classes taken (x_2) and school (x_3). Traditional ML approaches would treat all 3 predictors as input features. This is a single-level approach in which the only level present is the student-level. In contrast, a MuL approach would partition the dataset into groups based on the schools the students are enrolled in and analyze the students in each group using the remaining 2 features, x_1 and x_2. This is a 2-level approach, in which we call the school-level the higher level and the student-level the individual level. By analyzing and modeling the dataset from a MuL perspective, we acknowledge that data (e.g., a student's examination marks) within the same group (e.g., school) are more likely to be similar to each other than to data of a different group. Models learnt under the MuL approach allow for increased complexity of the model without reducing explainability. In our work, we develop an approach that allows SR to be used with MuL datasets.

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

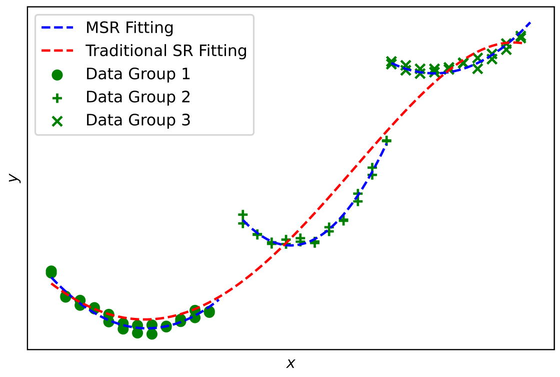

Figure 1: In this example, the best solution picked by traditional SR fits all the data without regard to the level, failing to capture the true parabolic relation of y with respect to x within each group.

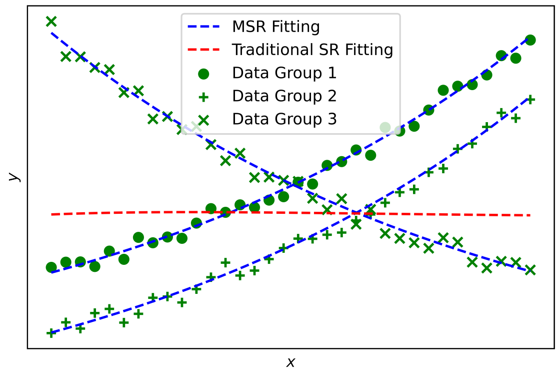

Figure 2: In this example, the overlapping range of x for the data from each group makes it difficult for traditional SR algorithms to learn a function for the whole set of datapoints.

Multi-Level Symbolic Regression – We take advantage of the synergy between MuL datasets and SR, and propose an algorithm called Multi-Level Symbolic Regression (MSR). Using the notation introduced earlier, recall that the task in traditional SR is to obtain f such $f(X) \approx y$. If we separate the numerical parameters in f from the function structure, then we can rewrite the task as obtaining f_θ and $\boldsymbol{\theta}$ such $f_\theta(X;\boldsymbol{\theta}) = f(X) \approx y$, where $\boldsymbol{\theta}$ is the vector of numerical parameters found in f and f_θ is the function structure of f. For example, in the function $f(X) = 3x_1 + 4^{x_2}$, we can separate f into $f_\theta(X;\boldsymbol{\theta}) = \alpha x_1 + \beta^{x_2}$, and $\boldsymbol{\theta} = [\alpha, \beta] = [3, 4]$.

In a MuL dataset with a higher level that has G groups, we can partition the dataset (X, y) as follows: $\{(X_j, y_j) \mid j \in \mathbb{G}\}$, where $\mathbb{G} = \{1, 2, \cdots, G\}$. The task of MSR is to obtain f_θ and $\{\boldsymbol{\theta}_j \mid j \in \mathbb{G}\}$ such that $f_\theta(X_j;\boldsymbol{\theta}_j) \approx y_j, \forall j \in \mathbb{G}$. Note that only a single function structure f_θ is learnt while G vectors of numerical parameters, $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_G\}$ are learnt.

MSR works well because the single function structure learnt, f_θ, serves as 'shared information' between the groups while the numerical parameters learnt for each group, $\boldsymbol{\theta}_j$, tailors to the unique relationship between data of each group. This is in contrast to applying traditional SR to the dataset as a whole and ignoring levels. Figures 1 & 2 gives examples of datasets in which traditional SR significantly underperforms MSR.

Another alternative to MSR is to treat the groups as multiple datasets and run traditional SR on each, ignoring the interaction between groups. In this approach, there is no interaction between the models of each group, thus the multiple individual models are each not benefiting from the full set of data collected, and may result in an over-fitted model per group. Also, the different function structure per group greatly reduces explainability, which is critical in various applications such as clinical equations in medicine, where a single f_θ across all groups is greatly valued (Cockcroft and Gault, 1976; Levey et al., 2009; Inker et al., 2021).

The application of MSR extends beyond MuL datasets. Even for tabular datasets without a preselected or given higher level, we can still utilize feature variables to partition the data into groups. We can think of MuL datasets as *explictly* MuL datasets and most other tabular datasets as *implicitly* MuL datasets. In this paper, we propose a metric, termed MLICC, to select a feature to serve as a level for datasets that are not explicitly MuL.

Finally, we point out a scenario that benefits from a MuL perspective with MSR. Many useful data in the real world are heterogeneous and come from various sources. If we isolate our analysis based on the data source, then we fail to use 'shared information' between the datasets. At the same time, there are real-world constraints preventing the merging of the data from different sources. We illustrate this later in the results section through an empirical example.

The main **contributions** of this paper are as follows:

1. We design an algorithm, MSR, which runs multiple parallel SR processes for each group and merges them to produce a single function structure.
2. To tackle datasets that are not explicitly MuL, we develop a metric, Mean Local Intraclass Correlation Coefficient (MLICC), to select the best feature to serve as a level.
3. We also release MSRBench, a database of MuL datasets (synthetic and real-world) which we developed and collated to evaluate MSR.
4. Our results and ablation studies demonstrate that MSR achieves a higher recovery rate and lower error on MSRBench compared to SOTA methods for SR and MuL datasets.

2 RELATED WORK

2.1 DistilSR

Most SR algorithms utilize traditional genetic programming to navigate through a large search space of function structures, resulting in a high degree of randomness in the algorithm (Orzechowski et al., 2018). This means that that while the search space of function structures is defined, the actual function structures that are evaluated during the runtime of the algorithm are a random subset of the full search space. In contrast, DistilSR (Fong and Motani, 2023) is an SR algorithm that exhaustively evaluates all function structures of a well-defined constrained space, $\mathcal{F}_\theta$. By utilizing gene expression programming's *K-Expressions* (Ferreira, 2002) of a fixed pre-selected length, a set of function structures of varying length can be obtained. This set forms $\mathcal{F}_\theta$, which is exhaustively evaluated by DistilSR, producing the best parameters for each function structure. We can say that $\{\arg\min_{\boldsymbol{\theta}\in\mathbb{R}^n} L(f_\theta(X;\boldsymbol{\theta}),y)|\hat{f}_\theta \in \mathcal{F}_\theta\} \leftarrow DistilSR(X,y)$. The benefit of having a fixed $\mathcal{F}_\theta$ which is exhaustively searched through is that we can now compare the performance for each $f_\theta \in \mathcal{F}_\theta$ across different sets of data. Specifically, in MSR, running DistilSR in parallel for each group, $\{(X_j, y_j) \mid j \in \mathbb{G}\}$, is useful to find f_θ and $\{\boldsymbol{\theta}_j \mid j \in \mathbb{G}\}$ such that $f_\theta(X_j;\boldsymbol{\theta}_j) \approx y_j, \forall j \in \mathbb{G}$.

2.2 Other SR Algorithms

Other successful SR algorithms exist for traditional SR tasks (Jin et al., 2019; Arnaldo et al., 2014), but they are not intentionally designed to handle MuL data and do not perform well on them. Though still not meant for MuL data, PS-Tree (Zhang et al., 2022) is an SR algorithm that divides the feature space into several subregions as one of the steps. In this paper, we compare our work to PS-Tree and demonstrate the uniqueness of MSR in successfully handling MuL data.

2.3 Multi-Level Modeling

To handle MuL data, statistical models have been built to capture the random effects of each group (Bryk and Raudenbush, 1992). In particular, under their multilevel modeling assumption of linearity, the random intercepts and slopes model (RISM) has become a popular out-of-the-box solution, which we use as a MuL benchmark method in this paper. In contrast to the simple linear models, there can be more complex models, featuring non-linear functions (Lee, 2022). However, these require hand-picked functions which require domain knowledge and expertise to build proper priors with respect to the dataset. In our paper, we propose MSR, which comes as an out-of-the-box solution for MuL datasets that can learn non-linear function structures without requiring a hand-picked function.

2.4 Intraclass Correlation Coefficient (ICC)

The ICC is a descriptive statistic that is commonly used by researchers in multi-level modeling as an indicator of the suitability of a multi-level model (Bliese, 1998; Shieh, 2012). An alternative to ICC is concordance correlation (CCC), but CCC is limited to partitions that allow only 2 groups (Liu et al., 2016), thus we adopted the more versatile ICC metric since it allows 2 or more groups. This includes assessing the suitability of a feature to be used as a level. In this paper, we use the notation $ICC(y;l)$ to indicate the ICC of the output variable, y, given that the input feature l is used as a level. The ICC models the output variable of the model as $y_{ij} = \mu + \gamma_j + \varepsilon_{ij}$, for $i = \{1,2,\cdots,N_j\}$, $j = \{1,2,\cdots,G\}$, where N_j is the number of data in the j-th group and G is the number of groups. γ_j and ε_{ij} are independent random variables which are distributed as follows: $\gamma_j \sim N\left(0,\sigma_\gamma^2\right)$, and $\varepsilon_{ij} \sim N\left(0,\sigma_\varepsilon^2\right)$. Here, σ_γ^2 is interpreted as the between-group variance and σ_ε^2 the within-group variance. Finally, the ICC, ρ, is defined as $\rho = \sigma_\gamma^2/\left(\sigma_\gamma^2+\sigma_\varepsilon^2\right)$. The ICC ranges from 0 to 1, with 0 suggesting that the total variance is attributed to the within-group variance and hence strongly not supporting MuL analysis and 1 suggesting that the total variance is attributed to the between-group variance and hence strongly supporting MuL analysis. In this paper, to estimate ICC, we use the 'ICC(1) formulation' (Shieh, 2012). We utilize ICC as a tool to select a feature variable to act as level on datasets where a level is not explicitly known. This enables us to extend the application MSR beyond *explicitly* MuL datasets to general tabular datasets (*implicitly* MuL datasets). In this paper, we also propose a modification to ICC to better identify non-linear within-group relationships.

3 METHODOLOGY

3.1 MSR Algorithm

In Algorithm 1, we introduce the pseudo code for MSR on MuL datasets (a MuL dataset is represented as $\{(X_j, y_j) \mid j \in \mathbb{G}\}$). DistilSR is first utilized to optimize the parameters for all function structures in $\mathcal{F}_\theta$. In our experiments, we select a K-Expression length of 7 and a primitive set of $\{+,-,*,/,**\}$, representing addition, subtraction, multiplication, division and power. These settings were chosen in the DistilSR paper to achieve short concise equations which were argued to be explainable, making them suitable for real-

Algorithm 1: MSR Pseudo Code

```
Input: $\{(X_j, y_j) \mid j \in \mathbb{G}\}, \mathcal{F}_\theta$
/* $\mathcal{F}_\theta$ is selected by $DistilSR$ */
Output: $f_\theta, \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \cdots, \boldsymbol{\theta}_G\}$
1 do in parallel
2     $\{\boldsymbol{\theta}_{j|\hat{f}_\theta} | \hat{f}_\theta \in \mathcal{F}_\theta\} \leftarrow DistilSR(X_j, y_j), \forall j \in \mathbb{G}$
3 end
4 for $\hat{f}_\theta \in \mathcal{F}_\theta$ do
5     $V_{\hat{f}_\theta} \leftarrow H\left(\left\{MSE\big(\hat{f}_\theta(X_j; \boldsymbol{\theta}_{j|\hat{f}}), y_j\big) | j \in \mathbb{G}\right\}\right)$
          /* $H$ denotes harmonic mean */
6 end
7 $\hat{f}^*_\theta \leftarrow \arg\min_{\hat{f}_\theta \in \mathcal{F}_\theta}(V_{\hat{f}_\theta})$
8 return $\hat{f}^*_\theta, \{\boldsymbol{\theta}_{j|\hat{f}^*_\theta} | j \in \mathbb{G}\}$
```

world application (Fong and Motani, 2023). In Steps 1 to 3 of Algorithm 1, note that DistilSR processes can be run in parallel, which allows for faster computation. In the context of SR, this is an important feature since the limitation of most SR algorithms is its relatively long runtime when compared against most other ML algorithms. In Steps 4 to 6 of Algorithm 1, we use the optimized parameters for each function structure to compute the mean-squared-error (MSE) as the loss (other types of loss can be used). We then assign a score, $V_{\hat{f}_\theta}$, to every function structure, $\hat{f}_\theta$, based on the harmonic mean of MSE computed using the function structure (and its parameters optimized by DistilSR) across all the G groups of data. The harmonic mean is given by $H(x_1, x_2, \ldots, x_n) = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$. Finally, in Steps 7 to 8 of Algorithm 1, the function structure with the best score is selected. Though using arithmetic mean for the scores might seem more natural, we realized that doing so heavily biases the results to the noisiest group, leading to lower prediction performances. We found the harmonic mean to be more robust to noisy groups of data, which we elaborate more in the results and discussion section.

3.2 Mean Local Intraclass Correlation Coefficient (MLICC)

To tackle datasets that do not have a pre-selected level, we develop MLICC to select the best feature to serve as a level. While ICC has been used in multi-level modeling to determine if a feature can serve as a level, it has 2 limitations. First, the ICC only takes into account the distribution of the output variable, irrespective of input features. Second, the ICC is primarily designed for the case where the relationship of the input features to output variable within the groups is linear.

Algorithm 2: MLICC Pseudo Code

```
Input: $X, y$, number of clusters: $K$,
chosen candidate feature to serve as a level: $l$
Output: $MLICC(X, y, K; l)$
1 $\{X_k \mid k \in \{1, \cdots, K\}\} \leftarrow KMeans_B(X, K)$
/* $KMeans_B$ denotes balanced K-means */
2 for $k \in \{1, \cdots, K\}$ do
3     $y_k \leftarrow FindCorrespondingY(X_k)$
4 end
5 return $Mean(\{ICC(y_k; l) | k \in \{1, \cdots, K\}\})$
```

For a MuL dataset with an output variable that has a non-linear within-group relationship with respect to the input features, the within-group variance of the output variable can be exceptionally high, resulting in low values of ICC, contrary to expectations.

To address these limitations, we modify ICC to be sensitive to non-linear relationships between the input features and output variables within each group, as outlined in Algorithm 2. Based on the observation that most relationships are locally linear (Miranda Filho et al., 2020), we utilize a clustering algorithm to create multiple 'local regions', via partitioning the dataset based on their proximity in the input feature space as shown in Step 1 of Algorithm 2. In Steps 2 to 4 of Algorithm 2, each of the partitions of the input features ('local regions') are then used to partition the output variables of the dataset, y. We then apply ICC to each partition of the output variable and compute the mean across the partitions. In order to generate partitions of sufficient size, we utilize the balanced k-means clustering algorithm, an special case of constrained k-means which guarantees equal-sized clusters (Malinen and Fränti, 2014). In our work, we select $K = \lfloor datasize/15 \rfloor$, which we found to strike a balance between creating sufficiently small 'local regions', but yet possessing sufficient samples to provide an informative coefficient value. Other values of K are possible, but one should note that setting K too large assigns too few points per cluster, while setting K too small creates too few clusters. In cases where the small 'local regions' possess data from only one group (using the candidate feature serving as a level), the ICC is not well-defined but we consider the ICC value to be 1, the maximum possible value. This is because we intend to use MLICC in the same way ICC is used, ranging from 0 (strong opposition for MuL analysis) to 1 (strong proposition for MuL analysis). If a 'local region' possesses only data from one group, then we strongly lean towards recommending MuL analysis since this is evidence in support of intra-group difference. Finally, the feature selected to be used as a level is found via $\arg\max_{l \in features} MLICC(X, y, K; l)$.

Table 1: Synthetic Function Structures

NAME	EXPRESSION
S1	$\{x_1/x_2 - x_3^{\alpha_j} \| j \in \mathbb{G}\}$
S2	$\{(x_1 - x_2)^{(x_3-\alpha_j)} \| j \in \mathbb{G}\}$
S3	$\{(x_1 + x_2) * (x_3 - \alpha_j) \| j \in \mathbb{G}\}$
S4	$\{(x_1 * x_2)/(x_3 * \alpha_j) \| j \in \mathbb{G}\}$
S5	$\{x_1^{x_2} * (x_3 - \alpha_j) \| j \in \mathbb{G}\}$
S6	$\{(x_1 + x_2) * x_3 * \alpha_j \| j \in \mathbb{G}\}$
S7	$\{(x_1 + x_2) - x_3 * \alpha_j \| j \in \mathbb{G}\}$
S8	$\{(x_1^{x_2} * \alpha_j)/x_3 \| j \in \mathbb{G}\}$
S9	$\{x_1 * x_2 * x_3^{\alpha_j} \| j \in \mathbb{G}\}$
S10	$\{(x_1/x_2)^{(x_3/\alpha_j)} \| j \in \mathbb{G}\}$
S11	$\{x_1/\alpha_j - x_2^{\beta_j} \| j \in \mathbb{G}\}$
S12	$\{(x_1 - \alpha_j)^{(x_2-\beta_j)} \| j \in \mathbb{G}\}$
S13	$\{(x_1 + \alpha_j) * (x_2 - \beta_j) \| j \in \mathbb{G}\}$
S14	$\{(x_1 * \alpha_j)/(x_2 * \beta_j) \| j \in \mathbb{G}\}$
S15	$\{x_1^{\alpha_j} * (x_2 - \beta_j) \| j \in \mathbb{G}\}$
S16	$\{(x_1 + \alpha_j) * x_2 * \beta_j \| j \in \mathbb{G}\}$
S17	$\{(x_1 + \alpha_j) - x_2 * \beta_j \| j \in \mathbb{G}\}$
S18	$\{(x_1^{\alpha_j} * \beta_j)/x_2 \| j \in \mathbb{G}\}$
S19	$\{x_1 * \alpha_j * x_2^{\beta_j} \| j \in \mathbb{G}\}$
S20	$\{(x_1/\alpha_j)^{(x_2/\beta_j)} \| j \in \mathbb{G}\}$

3.3 MSRBench, A Benchmark for MSR

To evaluate MSR, we developed synthetic MuL datasets and collated real-world datasets to form MSRBench.

Synthetic Datasets – We first define a collection of 20 function structure, from S1 to S20 as tabulated in Table 1. Let $U(a, b)$ denote uniform random sampling from lower limit, a, to upper limit, b. All variables, x_1, x_2, x_3 and parameters α, β are sampled from $U(1, 3)$, with a specified random seed. We create datasets from each function structure. The dataset names follow the following convention 'Dataset-{rows of data for each group}-{function structure}-{random seed number}'. For example, in the dataset named 'Dataset-(25,125)-S1-8', 25 and 125 rows of (x_1, x_2, x_3) are sampled (using random seed 8) for group 1 and 2 respectively, then we sample the parameters (using random seed 8), α_1 (for group 1), α_2 (for group 2), and substitute these parameters into S1. The substituted S1 is then provided the input features, (x_1, x_2, x_3), to generate $25 + 125 = 150$ output variables.

We create 6 experiment set-ups, each consisting of 100 datasets as follows (more details are provided in Appendix A.1):

(i) **Synthetic-3Var-Default:** 'Dataset-(100,100,100)-$\{M\}$-$\{N\}$', $\forall M \in \{\text{S1}, \cdots, \text{S10}\}, N \in \{1, \cdots, 10\}$.

(ii) **Synthetic-2Var-Default:** 'Dataset-(100,100,100)-$\{M\}$-$\{N\}$', $\forall M \in \{\text{S11}, \cdots, \text{S20}\}$, $N \in \{1, \cdots, 10\}$.

(iii) **Synthetic-3Var-VaryingNoise:** Same as Synthetic-3Var-Default, but output variables of the 3 groups have different Gaussian noise with variance of 10, 0.1 and 0.001 respectively.

(iv) **Synthetic-2Var-VaryingNoise:** Same as Synthetic-2Var-Default, but output variables of the 3 groups have different Gaussian noise with variance of 10, 0.1 and 0.001 respectively.

(v) **Synthetic-3Var-VaryingSize:** 'Dataset-(25,125)-$\{M\}$-$\{N\}$', $\forall M \in \{\text{S1}, \cdots, \text{S10}\}, N \in \{1, \cdots, 10\}$.

(vi) **Synthetic-2Var-VaryingSize:** 'Dataset-(25,125)-$\{M\}$-$\{N\}$', $\forall M \in \{\text{S11}, \cdots, \text{S20}\}, N \in \{1, \cdots, 10\}$.

Real-World Datasets – We use 7 experiment set-ups, with the three broad categories of MuL data (with a provided level), non-MuL tabular data (no pre-selected level, MLICC will be applied to determine a level) and multi-sources data. The 7 experiement set-ups are (more details are provided in Appendix A.2):

(i) **Real-World-MuL1:** MuL data from 'pupil educational attainment' (Paterson, 1991).

(ii) **Real-World-MuL2:** MuL data from 'malignant melanoma mortality' (Langford et al., 1998).

(iii) **Real-World-Tabular1:** Non-MuL data from PMLB '1029_LEV' (Romano et al., 2021).

(iv) **Real-World-Tabular2:** Non-MuL data from PMLB '1030_ERA' (Romano et al., 2021).

(v) **Real-World-Tabular3:** Non-MuL data from PMLB 'sleuth_case2002' (Romano et al., 2021).

(vi) **Real-World-MultiSource1:** Data from different geographical sources: Chicago, Amsterdam and Boston (Harrison and Rubinfeld, 1978).

(vii) **Real-World-MultiSource2:** Data for the US Government bond yields sourced from different time periods.

Datasets per set-up are made available in csv format in the Supplementary Materials. The detailed process of generating the synthetic data, and training and testing processes for datasets are also found in the Supplementary Materials.

4 RESULTS AND DISCUSSION

4.1 Synthetic Data Experiments

The performance of MSR is measured in terms of *recovery rate of function structure.* For each dataset, the top function structure selected by MSR must match the ground-truth function structure used to generate

Table 2: Recovery Rate of Function Structure (**Higher rates are better**)

	MSR (Ours)	MSR-A (Ours)	MSR-U (Our)	DSO-SR	RISM	DistilSR	PS-Tree	gplearn-SR
Synthetic-3Var-Default	**87%**	79%	76%	0%	10%	0%	0%	0%
Synthetic-2Var-Default	**80%**	75%	65%	0%	10%	0%	0%	0%
Synthetic-3Var-VaryingNoise	**76%**	54%	16%	0%	10%	0%	0%	0%
Synthetic-2Var-VaryingNoise	**73%**	41%	10%	0%	10%	0%	0%	0%
Synthetic-3Var-VaryingSize	**89%**	78%	88%	0%	10%	0%	0%	0%
Synthetic-2Var-VaryingSize	**80%**	76%	66%	0%	10%	0%	0%	0%

the data (see Table 1) to qualify as a recovery. This is implemented by using simplification from SymPy followed by a manual check. To illustrate a recovery, an example is in S11, where MSR recovers the exact function for all $j \in \{1, 2, 3\}$, whereas another algorithm, which failed to recover, produced $x_1/\alpha_j - x_2{}^{\beta_j}$ for $j \in \{1, 2\}$ and $x_1{}^{\alpha_j} - x_2{}^{\beta_j}$ for $j \in \{3\}$. In Table 2, the performance of MSR is recorded, along with the following 7 methods (resources and hyperparameters for the methods are provided in Appendix B):

(i) **MSR-Arithmetic (MSR-A)**, a variant of MSR in which Step 5 of Algorithm 1 uses an arithmetic mean instead.

(ii) **MSR-Unconstrained (MSR-U)**, a variant of MSR in which the function structure for each group can be different, losing the 'shared information' effect. Both MSR-A and MSR-U will be used for ablation analysis.

(iii) **Deep Symbolic Optimization Symbolic Regression (DSO-SR)**, a state-of-the-art SR algorithm that demonstrated the best performance among other SR algorithms for recovery of synthetic equations (Mundhenk et al., 2021; Kim et al., 2021; Petersen et al., 2019).

(iv) **RISM**, introduced in related works, serves as a benchmark to demonstrate the potential MuL analysis despite using a simple linear model.

(v) **DistilSR**, introduced in related works, will be used for ablation analysis since it is in essence, MSR without the MuL component.

(vi) **PS-Tree**, introduced in related works, serves as an SR algorithm designed for traditional SR tasks but has some design elements that can be useful for MuL data.

(vii) **gplearn-SR**, a widely-used vanilla SR algorithm (Koza, 1992).

Why is MSR the only method that is able to recover a significant number of function structures? While it would have been possible for DSO-SR, DistilSR and gplearn-SR to recover the function structures, it is unrealistic in practice. Indeed, the solutions could have evolved to be long enough to capture the group level difference (i.e. for S1 in Synthetic-3Var-Default, $y = x_1/x_2 - x_3^{\mathbb{1}_{j=1}(j)\alpha_1+\mathbb{1}_{j=2}(j)\alpha_2+\mathbb{1}_{j=3}(j)\alpha_3}$). However, searching the space of such long equations would increase the runtime of the algorithm exponentially. PS-Tree does not perform well as the splits do not occur on a sole feature and the groups are not designed to share the same function structure. Finally, RISM is a form of MuL analysis, but the linear assumption in RISM drastically restricts its complexity. The 10% value in RISM for Synthetic-3Var is accounted for by the 10 datasets generated using S7. Likewise, the 10% value in RISM for Synthetic-2Var is accounted for by the 10 datasets generated using S17.

Ablation: Why does MSR outperform MSR-A? Even though MSR-A performs well relative to the other methods, it is consistently outperformed by MSR in all 6 experiment set-ups. The reason is that for MSR-A, which uses an arithmetic mean in Step 5 of Algorithm 1, a high noise in any group will bias the algorithm towards picking the best function structure that optimizes well for that group. This means that the performance of the remaining groups (lower noise) is barely consequential to the algorithm because it is always overshadowed in magnitude by the 'worst-group performance'. In other words, MSR-A tends to pick the function structure with the best 'worst-group performance' and neglects the 'best-group performance'. This is supported by empirical results in Table 2, in which the recovery rate decreases from 79% and 75% to 54% and 41% respectively, when noise of varying variance is added to the output variable. In both set-ups with varying noise, the MSE of data in the group with Gaussian noise of variance 10 dominated in magnitude over the other 2 groups with Gaussian noise of variance 0.1 and 0.001. The optimization process of MSR-A in these 2 set-ups hence became reliant on the function structure which fits the noisy group the best, becoming a single-group analysis. This is the reason why our main algorithm, MSR, chose to

Table 3: Rate of Obtaining the Best Prediction Performance Among 6 Methods (**Higher rates are better**)

	MSR (Ours)	DSO-SR	RISM	DistilSR	PS-Tree	gplearn-SR
Synthetic-3Var-Default	**78%**	6%	16%	0%	0%	0%
Synthetic-2Var-Default	**81%**	3%	16%	0%	0%	0%
Synthetic-3Var-VaryingNoise	**79%**	14%	7%	0%	0%	0%
Synthetic-2Var-VaryingNoise	**82%**	16%	2%	0%	0%	0%
Synthetic-3Var-VaryingSize	**92%**	6%	2%	0%	0%	0%
Synthetic-2Var-VaryingSize	**94%**	5%	1%	0%	0%	0%

use the harmonic mean. The harmonic mean has the property that: $\min(x_1, \cdots, x_n) \leq H(x_1, \cdots, x_n)$ and $H(x_1, \cdots, x_n) \leq n \min(x_1, \cdots, x_n)$. The upper and lower bound of the harmonic mean of MSE is dependent on the minimum MSE achieved in all groups, this helps to increase the reliance of the score given to each function structure, $V_{\hat{f}_\theta}$, on the 'best-group performance'. Meanwhile, the score is still allowed to float freely between the upper and lower bound, which is dependent on the performance of the other groups, including the 'worst-group performance'. Thus, the harmonic mean helps to strike a balance in which the 'worst-group performance' still affects the score of a function structure, but the 'best-group performance' is no longer neglected. When comparing MSR to MSR-A results, it is also clear that the greatest difference occurs in the set-ups with varying noise, confirming the weakness of the arithmetic mean of MSE.

Ablation: Why does MSR outperform MSR-U? When the groups do not share a common function structure, as in the case of MSR-U, the recovery performance drops, as seen in Table 2. On inspection of the MSR-U solution for each dataset, we notice that MSR-U tends to overfit to the noise in the datasets since it is now given the flexibility choose the function structure of each group regardless of the other groups. This is verified by seeing the greatest difference in performance between MSR and MSR-U for the datasets with varying noise. In the groups with more noise, MSR-U overfits to the noise, obtaining a function structure for that group that is different from the ground-truth function structure. In a sense, the shared function structure in MSR doubles as a regularization mechanism, in addition to its function as 'shared information' and the practical advantages (i.e. a singular function structure is preferred for explainability).

Ablation: What happens if we remove the MuL aspect of MSR? MSR without enforcing the grouping is essentially DistilSR, which demonstrates much poorer performance, with a 0% recovery in all experiments as seen in Table 2.

Limitation: Why is MSR unable to recover 100% of the time? We found that in cases where MSR fails to recover the function structure, some of parameters for the groups, $\{\boldsymbol{\theta}_j \mid j \in \mathbb{G}\}$, were stuck at local optima points. This meant that the numerical optimization used in DistilSR became a limiting factor of the recovery rate. By swapping BFGS optimization (Broyden, 1970) used in DistilSR to other numerical optimization methods, we were unable to achieve consistently better results even at the cost of a longer runtime for some of these methods. This is consistent with other SR work which also found BFGS to be most effective in their work (Petersen et al., 2019).

In Table 3, the rate of obtaining the best prediction performance (in terms of MSE) among MSR, DSO-SR, RISM, DistilSR, PS-Tree and gplearn-SR is tabulated. The sum of percentages of each row in Table 3 is 100%.

Why is MSR most frequently the best performer? Since MSR had a high function structure recovery rate, we expect it to be the best performer especially since the other models were unable to recover most function structures, if any at all.

Why does MSR relative performance get even better under varying noise and size? All the other methods weight the error of each group equally. In the case of varying noise, the other methods tend to pick the best model that overfits to the noisiest group of data. The explanation for this is provided in the ablation discussion above. In the case of varying size, the other methods tend to pick the best model that overfits to the smallest-sized group. The explanation for this is that it is hard to learn a model that generalizes well from the smallest group of data, leading to high errors on data from the smallest-sized group. These high errors tend to dominate in magnitude over the errors from other groups of data, leading to overfitting on the smallest-sized groups. On the other hand, with the use of a harmonic mean of MSE for MSR, a better metric that balances of the performance on the groups is achieved.

Table 4: Average prediction performance (MSE) rank (SD in brackets) of MSR using various dataset features as levels. For both datasets, the prediction performance ranks correlate better with MLICC than ICC. **MLICC is a better selector** of a suitable level than ICC since the feature with best MLICC is more often the best choice to be used as a level for *implict* MSR. (Note: Results beyond the 3rd highest omitted)

	Synthetic Datasets		Real-World Datasets	
	MLICC	ICC	MLICC	ICC
Highest Coefficient Feature	**1.09 (0.29)**	1.36 (0.32)	**1.50 (0.52)**	1.95 (0.43)
2nd Highest Coefficient Feature	2.55 (0.45)	2.81 (0.67)	2.09 (0.53)	2.55 (0.59)
3rd Highest Coefficient Feature	3.64 (0.44)	3.45 (0.57)	2.91 (0.53)	3.36 (0.65)

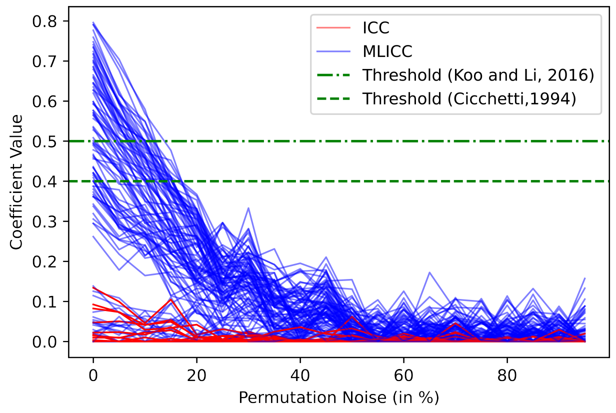

Figure 3: Spaghetti plot of MLICC & ICC values against permutation noise across multiple datasets, with varying level of permutation on the feature variable selected as a level. The ideal coefficient is above the threshold at 0% permutation noise and goes below the threshold as permutation noise increases. **MLICC demonstrates this property better than ICC.**

Outlier: For Synthetic-3Var-Default, why is MSR's rate in Table 3 (78%) lower than the recovery rate in Table 2 (87%)? In Synthetic-3Var-Default set-up, 10 of the 100 datasets were generated from the function structure, S7, provided in Table 1. The function structure in S7 can be fitted perfectly by RISM. When MSR recovers the function structure for these 10 datasets, it occasionally loses to RISM in terms of numerical optimization of parameters.

4.2 MLICC Experiments

Can MLICC identify features to be used as levels better than ICC? We compute MLICC and ICC on the synthetic and real-world datasets by using different features as levels. Then, we run the MSR algorithm by using each of these features as levels and measure the prediction performance (in terms of MSE). For MLICC and ICC to be a good identifier, they should be higher for a feature if using the feature as a level allows MSR to obtain a lower MSE (compared to other candidate features). In other words, MLICC and ICC should correlate well with the rank of the prediction performance. In Table 4, we demonstrate that MLICC shows a good correlation, and most importantly, the top feature selected by MLICC is more often the feature that leads to the lowest MSE, as compared to ICC. Thus, in the context of MSR, MLICC is a more effective identifier.

Is the absolute value of MLICC more useful than ICC? Although not used in the context of multi-level datasets, 2 other works have provided guidelines of thresholds for ICC: Poor:< 0.4 (Cicchetti, 1994) and Poor: < 0.5 (Koo and Li, 2016). We ran experiments across all synthetic datasets, and applied a partial random permutation on the feature that serves as a level (we call this permutation noise), ranging from 0% to 100%. We then calculated the ICC and MLICC of each of these datasets and plot the relationship of the coefficient values with the permutation noise in the spaghetti plot in Figure 3. Since ICC ranges from a value of 0 to 1, the mean of ICCs, MLICC, also ranges from 0 to 1. From Figure 3, it can be seen that even when synthetic MuL data is used, the ICC value is extremely low in most cases, close to 0. MLICC, on the other hand, accurately reports a sufficiently high value when there is 0 permutation noise. As more permutation noise is added, MLICC is also lowered, and is in fact below both thresholds. In accordance to the thresholds of ICC in other applications, the absolute value of MLICC provides a better indication of whether a feature serves as a level.

4.3 Real-World Data Experiments

Does MSR perform well on real-world datasets as well? To evaluate MSR's performance, we measure the normalized root-mean-squared-error

Table 5: Average NRMSE of MSR and 5 other methods on 6 real-world experiment set-ups (SD in brackets). MSR demonstrates the best performance (lowest NRMSE) in each set-up. (**Lower NRMSE is better**)

	MSR (Ours)	DSO-SR	RISM	DistilSR	PS-Tree	gplearn-SR
RealWorld-MuL1	**0.642 (0.171)**	1.07 (0.19)	0.775 (0.19)	0.995 (0.18)	0.849 (0.12)	1.18 (0.56)
RealWorld-MuL2	**0.672 (0.014)**	1.00 (0.018)	0.700 (0.0093)	1.00 (0.031)	0.700 (0.091)	0.731 (0.022)
RealWorld-Tabular1	**0.651 (0.042)**	0.708 (0.063)	0.681 (0.051)	0.82 (0.036)	0.674 (0.044)	0.812 (0.070)
RealWorld-Tabular2	**0.736 (0.022)**	0.828 (0.055)	0.807 (0.23)	0.851 (0.022)	0.787 (0.021)	0.833 (0.041)
RealWorld-Tabular3	**0.754 (0.069)**	1.03 (0.098)	0.804 (0.16)	1.04 (0.095)	0.869 (0.075)	0.801 (0.12)
RealWorld-MultiSource1	**0.501 (0.061)**	N.A.	0.668 (0.055)	N.A.	N.A.	N.A.
RealWorld-MultiSource2	**0.207 (0.049)**	0.904 (0.026)	0.447 (0.11)	0.894 (0.019)	0.288 (0.057)	0.771 (0.22)

(NRMSE) performance against DSO-SR, RISM, DistilSR, PS-Tree and gplearn-SR through the real-world experiments. We use the formulation NRMSE $= \frac{RMSE}{\sigma_y}$, given in SOTA SR work (Petersen et al., 2019). We tabulate our results in Table 5, which shows that MSR outperforms all other methods, regardless of whether the datasets are (i) *explicitly* MuL, (ii) generic tabular (no pre-selected level) in which MLICC is used to select a level (*implictly* MuL), or (iii) multi-source in which we use the source as a level.

Why is the value associated with RealWorld-MultiSource1 missing for 3 other methods? In RealWorld-MultiSource1, we use housing price data from 3 sources: Chicago, Amsterdam and Boston. Since the 3 datasets do not share a common currency, it is not realistic to combine the dataset for analysis. Selecting a fixed exchange rate for each currency does not accurately represent the currency as exchange rates fluctuate with time. On the other hand, by faithfully using the time-varying market exchange rate, a value that was once stationary in the original currency will now fluctuate. Both methods distort the data. By treating the source of the data as a level, and hence the multiple datasets as an MuL problem, MSR is able to avoid such complications by fitting to each dataset separately, yet benefit from the 'shared information' via the common function structure.

How does MSR's discovered function structure compare to hand-crafted function structures? In RealWorld-MultiSource2, US Government bond yields data is used. This real-world problem has been successfully modeled with an equation and has a well-known function structure that is used widely, with the parameters published on a daily basis (Gürkaynak et al., 2007). The function structure can be represented as $f_\theta(X;\boldsymbol{\theta}) = \beta_0 + \beta_1 \exp(-x_1/\tau_1) + \beta_2 (x_1/\tau_1) \exp(-x_1/\tau_1) + \beta_3 (x_1/\tau_2) \exp(-x_1/\tau_2)$, where β, τ are parameters and x_1 (the only feature of X) is the maturity in years. This achieves an NRMSE of 0.381 on the entire dataset. From MSR in Table 5, we obtained a much lower NRMSE of 0.207, in which we found the function structure, $f_\theta(X;\boldsymbol{\theta}) = ({x_1}^2+\beta_0 x_1+\beta_1)/(\beta_2 {x_1}^2+\beta_3 x_1+\beta_4)$. Interestingly, this is equivalent to a Padé approximant of order [2,2], which to our best knowledge, has been studied to be a possible alternative in modeling bond yields (Fowe, 2007). Though the well-known function structure by Gürkaynak et al. (2007) is still here to stay for legacy and domain-specific reasons, MSR's discovered function structure requires one less parameter and yet achieves a better NRMSE, serving as a testimony to MSR's ability to discover competitive function structures.

5 CONCLUSION

In this paper, we introduce the MuL approach to SR, and propose MSR, a parallelizable algorithm which learns a common function structure across groups (partitioned on a level) which serves as both 'shared information' and regularization. At the same time, the learnt parameter values for each group capture the unique within-group relationships. To tackle general tabular datasets without a pre-selected level, we propose a new metric, MLICC, to select the best feature to serve as a level. We then demonstrate the increased effectiveness of MLICC over ICC on empirical experiments. Then, we release MSRBench, a database of MuL datasets which we developed and collated. Finally, through the rigorous testing of MSR on both synthetic and real-world datasets in MSRBench, we consistently demonstrate that MSR achieves a higher recovery rate and lower error on MSRBench compared to SOTA methods for SR and MuL datasets. We believe that MSR can serve as an important automated tool in various fields, such as medicine and finance, where the MuL nature of many clinical equations and financial models makes MSR a perfect fit.

Acknowledgments

This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG3-PhD-2023-08-052T), and A*STAR, CISCO Systems (USA) Pte. Ltd and National University of Singapore under its Cisco-NUS Accelerated Digital Economy Corporate Laboratory (Award I21001E0002).

References

Arnaldo, I., Krawiec, K., and O'Reilly, U.-M. (2014). Multiple regression genetic programming. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 879–886.

Bliese, P. D. (1998). Group size, icc values, and group-level correlations: A simulation. *Organizational research methods*, 1(4):355–373.

Broyden, C. G. (1970). The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90.

Bryk, A. S. and Raudenbush, S. W. (1992). *Hierarchical linear models: applications and data analysis methods.* Sage Publications, Inc.

Cicchetti, D. V. (1994). Guidelines, criteria, and rules of thumb for evaluating normed and standardized assessment instruments in psychology. *Psychological assessment*, 6(4):284.

Cockcroft, D. W. and Gault, H. (1976). Prediction of creatinine clearance from serum creatinine. *Nephron*, 16(1):31–41.

Ferreira, C. (2002). *Gene Expression Programming in Problem Solving*, pages 635–653. Springer London, London.

Ferreira, J., Pedemonte, M., and Torres, A. I. (2019). A genetic programming approach for construction of surrogate models. In *Computer Aided Chemical Engineering*, volume 47, pages 451–456. Elsevier.

Fong, K. S. and Motani, M. (2023). Distilsr: A distilled version of gene expression programming symbolic regression. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 567–570.

Fowe, T.-k. (2007). Pade approximants and one of its applications. *Master's thesis, University of Central Florida, https://stars.library.ucf.edu/etd/3160.*

Gürkaynak, R. S., Sack, B., and Wright, J. H. (2007). The us treasury yield curve: 1961 to the present. *Journal of monetary Economics*, 54(8):2291–2304.

Hamaker, E. L. and Muthén, B. (2020). The fixed versus random effects debate and how it relates to centering in multilevel modeling. *Psychological methods*, 25(3):365.

Harrison, D. and Rubinfeld, D. L. (1978). Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102.

Inker, L. A., Eneanya, N. D., Coresh, J., Tighiouart, H., Wang, D., Sang, Y., Crews, D. C., Doria, A., Estrella, M. M., Froissart, M., et al. (2021). New creatinine-and cystatin c–based equations to estimate gfr without race. *New England Journal of Medicine*, 385(19):1737–1749.

Jin, Y., Fu, W., Kang, J., Guo, J., and Guo, J. (2019). Bayesian symbolic regression. *arXiv preprint arXiv:1910.08892.*

Kim, J. T., Larma, M. L., and Petersen, B. K. (2021). Distilling wikipedia mathematical knowledge into neural network models. *arXiv preprint arXiv:2104.05930.*

Koo, T. K. and Li, M. Y. (2016). A guideline of selecting and reporting intraclass correlation coefficients for reliability research. *Journal of chiropractic medicine*, 15(2):155–163.

Koza, J. R. (1992). Genetic programming. on the programming of computers by means of natural selection. *Complex adaptive systems.*

La Cava, W., Orzechowski, P., Burlacu, B., de França, F. O., Virgolin, M., Jin, Y., Kommenda, M., and Moore, J. H. (2021). Contemporary symbolic regression methods and their relative performance. *Neurips Track on Datasets and Benchmarks.*

Langford, I. H., Bentham, G., and McDonald, A.-L. (1998). Multi-level modelling of geographically aggregated health data: a case study on malignant melanoma mortality and uv exposure in the european community. *Statistics in medicine*, 17(1):41–57.

Lee, S. Y. (2022). Bayesian nonlinear models for repeated measurement data: An overview, implementation, and applications. *Mathematics*, 10(6):898.

Levey, A. S., Stevens, L. A., Schmid, C. H., Zhang, Y., Castro III, A. F., Feldman, H. I., Kusek, J. W., Eggers, P., Van Lente, F., Greene, T., et al. (2009). A new equation to estimate glomerular filtration rate. *Annals of internal medicine*, 150(9):604–612.

Liu, J., Tang, W., Chen, G., Lu, Y., Feng, C., and Tu, X. M. (2016). Correlation and agreement: overview and clarification of competing concepts and measures. *Shanghai archives of psychiatry*, 28(2):115–120.

Malinen, M. I. and Fränti, P. (2014). Balanced k-means for clustering. In *Structural, Syntactic, and*

Statistical Pattern Recognition: Joint IAPR International Workshop, S+ SSPR 2014, Joensuu, Finland, August 20-22, 2014. Proceedings, pages 32–41. Springer.

Miranda Filho, R., Lacerda, A., and Pappa, G. L. (2020). Explaining symbolic regression predictions. In *2020 IEEE congress on evolutionary computation (CEC)*, pages 1–8. IEEE.

Mundhenk, T. N., Landajuela, M., Glatt, R., Santiago, C. P., Faissol, D. M., and Petersen, B. K. (2021). Symbolic regression via neural-guided genetic programming population seeding. *arXiv preprint arXiv:2111.00053*.

Orzechowski, P., La Cava, W., and Moore, J. H. (2018). Where are we now? a large benchmark study of recent symbolic regression methods. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1183–1190.

Paterson, L. (1991). Socio-economic status and educational attainment: a multi-dimensional and multi-level study. *Evaluation & Research in Education*, 5(3):97–121.

Petersen, B. K., Larma, M. L., Mundhenk, T. N., Santiago, C. P., Kim, S. K., and Kim, J. T. (2019). Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *The International Conference on Learning Representations*.

Romano, J. D., Le, T. T., La Cava, W., Gregg, J. T., Goldberg, D. J., Chakraborty, P., Ray, N. L., Himmelstein, D., Fu, W., and Moore, J. H. (2021). Pmlb v1.0: an open source dataset collection for benchmarking machine learning methods. *arXiv preprint arXiv:2012.00058v2*.

Shieh, G. (2012). A comparison of two indices for the intraclass correlation coefficient. *Behavior research methods*, 44:1212–1223.

Sun, S., Ouyang, R., Zhang, B., and Zhang, T.-Y. (2019). Data-driven discovery of formulas by symbolic regression. *MRS Bulletin*, 44(7):559–564.

Udrescu, S.-M. and Tegmark, M. (2020). AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631.

Wang, Y., Wagner, N., and Rondinelli, J. M. (2019). Symbolic regression in materials science. *MRS Communications*, 9(3):793–805.

Zhang, H., Zhou, A., Qian, H., and Zhang, H. (2022). Ps-tree: A piecewise symbolic regression tree. *Swarm and Evolutionary Computation*, 71:101061.

A DATASET DETAILS – MSRBench

MSRBench is a collection of synthetic MuL datasets we developed and real-world datasets we collated.

A.1 Synthetic Datasets

Table 6: Synthetic Function Structures

NAME	EXPRESSION	NAME	EXPRESSION
S1	$\{x_1/x_2 - x_3^{\alpha_j} \| j \in \mathbb{G}\}$	S11	$\{x_1/\alpha_j - x_2^{\beta_j} \| j \in \mathbb{G}\}$
S2	$\{(x_1 - x_2)^{(x_3-\alpha_j)} \| j \in \mathbb{G}\}$	S12	$\{(x_1 - \alpha_j)^{(x_2-\beta_j)} \| j \in \mathbb{G}\}$
S3	$\{(x_1 + x_2) * (x_3 - \alpha_j) \| j \in \mathbb{G}\}$	S13	$\{(x_1 + \alpha_j) * (x_2 - \beta_j) \| j \in \mathbb{G}\}$
S4	$\{(x_1 * x_2)/(x_3 * \alpha_j) \| j \in \mathbb{G}\}$	S14	$\{(x_1 * \alpha_j)/(x_2 * \beta_j) \| j \in \mathbb{G}\}$
S5	$\{x_1^{x_2} * (x_3 - \alpha_j) \| j \in \mathbb{G}\}$	S15	$\{x_1^{\alpha_j} * (x_2 - \beta_j) \| j \in \mathbb{G}\}$
S6	$\{(x_1 + x_2) * x_3 * \alpha_j \| j \in \mathbb{G}\}$	S16	$\{(x_1 + \alpha_j) * x_2 * \beta_j \| j \in \mathbb{G}\}$
S7	$\{(x_1 + x_2) - x_3 * \alpha_j \| j \in \mathbb{G}\}$	S17	$\{(x_1 + \alpha_j) - x_2 * \beta_j \| j \in \mathbb{G}\}$
S8	$\{(x_1^{x_2} * \alpha_j)/x_3 \| j \in \mathbb{G}\}$	S18	$\{(x_1^{\alpha_j} * \beta_j)/x_2 \| j \in \mathbb{G}\}$
S9	$\{x_1 * x_2 * x_3^{\alpha_j} \| j \in \mathbb{G}\}$	S19	$\{x_1 * \alpha_j * x_2^{\beta_j} \| j \in \mathbb{G}\}$
S10	$\{(x_1/x_2)^{(x_3/\alpha_j)} \| j \in \mathbb{G}\}$	S20	$\{(x_1/\alpha_j)^{(x_2/\beta_j)} \| j \in \mathbb{G}\}$

To generate synthetic datasets, we first define a collection of 20 function structures, from S1 to S20 as tabulated in Table 6. These function structures were selected randomly (with random seed 0) from all possible K-Expressions (Ferreira et al., 2019) of length 7 with a primitive set of $\{+, -, *, /, **\}$, representing addition, subtraction, multiplication, division and power.

Then we generate the parameters, α, β, which were sampled uniformly from a range of 1 to 3, inclusive. These parameter values were then substituted into the equations to generate the 'ground truth'. Only after the 'ground truth' is selected, the values of the variables, x_1, x_2, x_3, are selected, again from a uniform random sampling (with a specified random seed) from a range of 1 to 3. This allows us to create datasets from each function structure.

To make the datasets name meaningful to the reader, we use the following convention in naming our datasets: 'Dataset-{rows of data for each group}-{function structure}-{random seed number}'. For example, in the dataset named 'Dataset-(25,125)-S1-8', we sample the parameters (using random seed 8), α_1 (for group 1), α_2 (for group 2), and substitute these parameters into S1. More specifically, by referring to our files for 'Dataset-(25,125)-S1-8', $y = x_1/x_2 - x_3^{2.94312717}$ for $j = 1$ and for $y = x_1/x_2 - x_3^{1.89899397}$ for $j = 2$.

The substituted S1 then forms the 'ground truth' for these dataset, which MSR and the other methods aims to recover. 25 and 125 rows of (x_1, x_2, x_3) are then sampled (using random seed 8) for group 1 and 2 respectively, which are provided to the 'ground truth' to generate the outputs.

We create 6 experiment set-ups, each consisting of 100 datasets as follows:

1. **Synthetic-3Var-Default:** 'Dataset-(100,100,100)-{M}-{N}', $\forall M \in \{\text{S1}, \cdots, \text{S10}\}, N \in \{1, \cdots, 10\}$.
2. **Synthetic-2Var-Default:** 'Dataset-(100,100,100)-{M}-{N}', $\forall M \in \{\text{S11}, \cdots, \text{S20}\}, N \in \{1, \cdots, 10\}$.

3. **Synthetic-3Var-VaryingNoise:** 'Dataset-(100,100,100)-$\{M\}$-$\{N\}$', $\forall M \in \{\text{S1}, \cdots, \text{S10}\}, N \in \{1, \cdots, 10\}$ but with output variables of the 3 groups having different Gaussian noise with variance of 10, 0.1 and 0.001 respectively.

4. **Synthetic-2Var-VaryingNoise:** 'Dataset-(100,100,100)-$\{M\}$-$\{N\}$', $\forall M \in \{\text{S11}, \cdots, \text{S20}\}, N \in \{1, \cdots, 10\}$ but with output variables of the 3 groups having different Gaussian noise with variance of 10, 0.1 and 0.001 respectively.

5. **Synthetic-3Var-VaryingSize:** 'Dataset-(25,125)-$\{M\}$-$\{N\}$', $\forall M \in \{\text{S1}, \cdots, \text{S10}\}, N \in \{1, \cdots, 10\}$.

6. **Synthetic-2Var-VaryingSize:** 'Dataset-(25,125)-$\{M\}$-$\{N\}$', $\forall M \in \{\text{S11}, \cdots, \text{S20}\}, N \in \{1, \cdots, 10\}$.

A.2 Real-World Datasets

We created 7 experiment set-ups as well, each using one real-world dataset:

1. **RealWorld-MuL1:** MuL data on 'malignant melanoma mortality' (Langford et al., 1998) with a pre-selected level.

2. **RealWorld-MuL2:** MuL data 'pupil educational attainment' (Paterson, 1991) with a pre-selected level.

3. **RealWorld-Tabular1:** Non-MuL (no pre-selected level) data from '1029_LEV' in PMLB database(Romano et al., 2021) (under MIT License). Using MLICC, x_2 was selected as a level.

4. **RealWorld-Tabular2:** Non-MuL (no pre-selected level) data from '1030_ERA' in PMLB database(Romano et al., 2021) (under MIT License). Using MLICC, x_2 was selected as a level.

5. **RealWorld-Tabular3:** Non-MuL (no pre-selected level) data from 'sleuth_case2002' in PMLB database(Romano et al., 2021) (under MIT License). Using MLICC, x_1 was selected as a level.

6. **RealWorld-MultiSource1:** Data from different sources that provides data for the geographical locations, Chicago, Amsterdam and Boston (Harrison and Rubinfeld, 1978) respectively (under CC0: Public Domain License).

7. **RealWorld-MultiSource2:** Data for the US Government bond yields sourced from 3 January 2022 to 30 December 2022.

B EXPERIMENT DETAILS

We used 7 algorithms for the experiments: MSR-A is a variant of MSR which uses an arithmetic mean instead. MSR-U is a variant of MSR in which the function structure for each group can be different. DistilSR, MSR and the MSR variants use K-Expressions of length 7 with a primitive set of $\{+, -, *, /, **\}$, representing addition, subtraction, multiplication, division and power, as these settings allowed for high prediction performance and yet remains explainable (Fong and Motani, 2023). Deep Symbolic Optimization Symbolic Regression (DSO-SR) is a state-of-the-art SR algorithm that demonstrated the best performance among other SR algorithms for recovery of synthetic equations (Mundhenk et al., 2021; Kim et al., 2021; Petersen et al., 2019). We selected the hyperparameters according to what was tuned to produce the best performance on the paper's recovery rate experiments, which is inclusive of the selection of primitive set of $\{+, -, *, /, sin, cos, exp, log\}$. RISM, which was introduced in the related works section, serves as a benchmark to demonstrate the potential MuL analysis despite using a simple linear model. PS-Tree used the default hyperparameters. gplearn-SR is a widely-used vanilla SR algorithm (Koza, 1992), which we tuned to find that $\{+, -, *, /, **\}$ to be the best primitive set for the synthetic data. The same hyperparameters were used throughout the synthetic and real-world experiments. All experiments were run on Intel(R) Xeon(R) CPU E5-2627 v4@2.30GHz and 128GB RAM.

All experiments used a 75-25 train-test split, with the exception of RealWorld-MultiSource2, in which a 82-18 train-test split in order to have sufficient training data for each group.

C EXPERIMENT FILES

The link for the files are: https://github.com/kentridgeai/MSR-Supplementary

Following the sequence of the Methodology subsections:

1. Link for MSR Algorithm– https://github.com/kentridgeai/MSR-Supplementary/tree/main/MSR.

   Contains Jupyter notebook example with MSR algorithm implementation and accompanied with supporting files.

2. Link for MLICC – https://github.com/kentridgeai/MSR-Supplementary/tree/main/MLICC.

   Contains Jupyter notebook example with MLICC implementation and accompanied with supporting files.

3. Link for MSRBench – https://github.com/kentridgeai/MSR-Supplementary/tree/main/MSRBench.

   Contains csv files used in experiments along with ground truth equations for synthetic datasets.