Improving Genetic Programming Based Symbolic Regression Using Deterministic Machine Learning

Ilknur Icke and Joshua C. Bongard
Department of Computer Science
The University of Vermont

IEEE Congress on Evolutionary Computation
CEC 2013

What is wrong with Genetic Programming Based Symbolic Regression (GP-SR) ?

- Easy to implement, but hard to analyze
- Works well on toy problems but struggles on high dimensional data

Vicious cycle of GP-SR:

- GP-SR effectively performs feature elimination *only when* sufficiently accurate models are evolved
- High dimensionality makes it difficult for GP-SR to evolve sufficiently accurate models!

What is wrong with Genetic Programming Based Symbolic Regression (GP-SR) ?

- Easy to implement, but hard to analyze
- Works well on toy problems but struggles on high dimensional data

Vicious cycle of GP-SR:

- GP-SR effectively performs feature elimination *only when* sufficiently accurate models are evolved
- High dimensionality makes it difficult for GP-SR to evolve sufficiently accurate models!

Our hypothesis: Hybridize and you shall win

A hybrid genetic programming (GP) and deterministic machine learning (ML) based symbolic regression algorithm out-performs the genetic programming based symbolic regression (GP-SR) algorithm alone

Method:

- Perform feature extraction using a deterministic, fast machine learning algorithm
- Pass the extracted features to GP-SR for model building

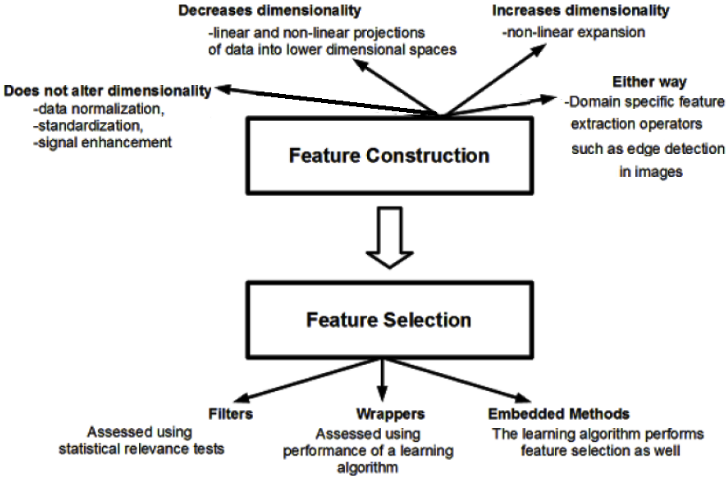
Our hypothesis: Hybridize and you shall win

A hybrid genetic programming (GP) and deterministic machine learning (ML) based symbolic regression algorithm out-performs the genetic programming based symbolic regression (GP-SR) algorithm alone

Method:

- Perform feature extraction using a deterministic, fast machine learning algorithm
- Pass the extracted features to GP-SR for model building

Feature Extraction as a Sequential Process



Feature selection

- Naive way: consider one feature at a time
- Subset Selection
  - *Filters*-independent of the learning algorithm
  - *Wrappers*-assess informativeness based on the performance of the learning algorithm
  - *Embedded Methods*-built in the learning algorithm: decision trees, regularization approach

Regularization for Linear Regression

Given a multivariate dataset $X = \{x_1, x_2, ..., x_N\}$ of observations, the response variable Y is defined as:

$$Y = f(X) = \beta_0 + \sum_{j=1}^{N} \beta_j * x_j$$

$$RSS = min_\beta(\sum_{i=1}^{N} y_i - \beta_0 - \sum_{j=1}^{N} \beta_j * x_{ij})^2$$

Learning algorithm applied to the training data $=>$ overfitting. An additional constraint on the coefficients is imposed in order to tame the coefficients ($\sum_{j=1}^{N} ||\beta_j||_1 \leq t$).

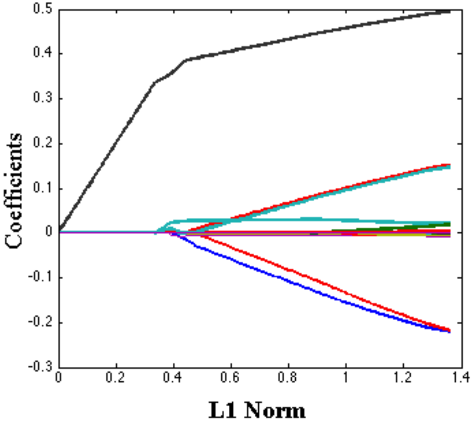
Regularization for Linear Regression cont'd

- LASSO-least absolute shrinkage and selection operator (constraint $l_1$norm)
- Ridge Regression (constraint l_2 norm)
- Elastic Net (hybrid of the two above)

$$Y = f(X) = \beta_0 + \sum_{j=1}^{N} \beta_j * x_j + \lambda_2 ||\beta||_2^2 + \lambda_1 ||\beta||_1$$

λ_1, λ_2 are balanced by a single *mixing* parameter $(0 \leq \alpha \leq 1)$ At the extreme values of α, elastic net behaves like purely lasso or purely ridge regression.

Elastic Net



Adding Nonlinearity-Generalized Linear Models

$$Y = f(X) = \beta_0 + \sum_{j=1}^{N} \beta_j * b_j(X)$$

where $b_j(X)$ are nonlinear basis functions applied to the input variables in order to construct new features.

The Fast Function Extraction: FFX Algorithm

The FFX Idea from McConaghy, 2011

Input $x_1, x_2, x_3, \ldots, x_N$

Stage 1: Feature Construction

Unary and Binary interactions

$\log(x_1)$, $\mathrm{sqrt}(x_1)$ ...

$x_1 * x_2$, $\log(x_1 * x_2)$ ...

Stage 2: Model Building

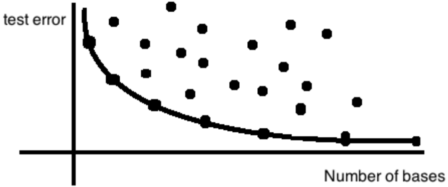
For a set of λs

$y = a1 * x_1 + b$

$y = a1 * x_1 + a2 * x_1 * x_3 + b$

.....

Stage 3: Model Selection

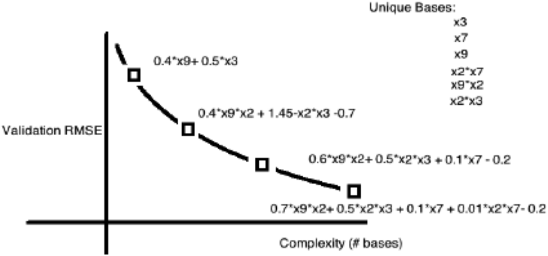


Improving GP-SR using Deterministic Machine Learning

Algorithm 3: The hybrid FFX/GP-SR algorithm

Input: V={v_1,v_2, ..., v_N}
Output: One best model with respect to the validation data error and complexity

```
nonDominatedModels = ffx(trainingDataset)
bases = extractBasisFunctions (nonDominatedModels,
                               validationDataset)
newDataset=createNewDataset(bases)
bestModel = GP-SR(newDataset)
```



Synthetic Benchmark Data Suite Generation

Systemically generated benchmark data suite:

- number of variables : 1-3 , 10 , 25
- order of polynomial: 1 -4
- number of basis functions: 1 -4

- 2500 training, 1250 validation and 1250 test data points
- for each type of polynomial, 30 different datasets

Evaluation Procedure

Experiments

For each dataset :

- FFX runs 1 time
- 30 runs of GP-SR
- 30 runs of FFX/GP-SR

Evaluation Select the model with lowest prediction error on validation set and report:

- prediction error on test set
- similarity to the correct functional form

Algorithm parameters

Parameter	Value
Basis Function Expansion	Exponents : 1 Interactions : Unary, Binary Operators : { }
Elastic Net	α : $\{0, 0.05, 0.1, ..., 1\}$ λ : 100 λ values calculated by glmfit based on α Maximum basis functions allowed : 250
Model Selection	Non-dominated models with respect to validation data error versus number of bases

Default FFX-variant parameters

Algorithm Parameters

Parameter	Value
Representation	GPTIPS Multigene syntax tree Number of genes: 1 Maximum tree depth: 7
Population Size Runtime Budget	500 1 minute
Selection Tournament Size	Lexicographic tournament selection 7
Crossover Operator Crossover Probability	Sub-tree crossover 0.85
Mutation Operator Mutation Probability	Sub-tree mutation 0.1
Reproduction Probability	0.05
Building Blocks	Operators: $\{+, -, *, \mathit{protected}/\}$ Terminal Symbols: $\{x_1, ..., x_N\}$
Fitness	$\frac{1}{N}\sqrt{\sum(y-\hat{y})^2}$
Elitism	Keep 1 best individual

Default GP-SR parameters

Results on 1D data

Example polynomials:

order 1 polynomial:

(1) $y = 0.288 * x_1 + 0.8446$

order 2 polynomials:

(1) $y = 0.14 * x_1^2 + 0.629$

(2) $y = 0.12 * x_1 + 0.03 * x_1^2 + 0.29$

order 3 polynomials:

(1) $y = -0.31 * x_1^3 - 0.11$

(2) $y = 1.35 * x_1^2 - 0.83 * x_1^3 + 0.139$

(3) $y = 0.13 * x_1 + 0.44 * x_1^2 + 0.34 * x_1^3 + 0.39$

order 4 polynomials:

(1) $y = 0.20 * x_1^4 + 0.13$

(2) $y = 0.24 * x_1^3 + 0.23 * x_1^4 + 0.39$

(3) $y = 0.75 * x_1^2 + 0.30 * x_1^3 + 0.35 * x_1^4 + 0.334$

(4) $y = 0.02 * x_1 + 0.13 * x_1^2 + 0.301 * x_1^3 + 0.32 * x_1^4 + 0.91$

Number of times correct form is discovered:

Standalone GP-SR (1 minute)

		Bases			
		1	2	3	4
Order of the Polynomial	1	30	-	-	-
	2	30	29	-	-
	3	30	27	19	-
	4	30	27	11	16

FFX runs with unary (x_i) and binary interactions ($x_i * x_j$) (average run time: 7 seconds)

		Bases			
		1	2	3	4
Order of the Polynomial	1	30	-	-	-
	2	30	30	-	-
	3	0	0	0	-
	4	0	0	0	0

FFX/GP-SR (1 minute GP run on FFX-generated dataset)

		Bases			
		1	2	3	4
Order of the Polynomial	1	30	-	-	-
	2	30	27	-	-
	3	30	26	19	-
	4	30	28	16	17

Correctness of the polynomial form

Results on 1D data

Example polynomials:

order 1 polynomial:

(1) $y = 0.288 * x_1 + 0.8446$

order 2 polynomials:

(1) $y = 0.14 * x_1^2 + 0.629$

(2) $y = 0.12 * x_1 + 0.03 * x_1^2 + 0.29$

order 3 polynomials:

(1) $y = -0.31 * x_1^3 - 0.11$

(2) $y = 1.35 * x_1^2 - 0.83 * x_1^3 + 0.139$

(3) $y = 0.13 * x_1 + 0.44 * x_1^2 + 0.34 * x_1^3 + 0.39$

order 4 polynomials:

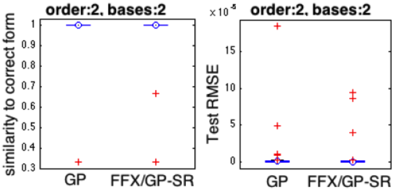
(1) $y = 0.20 * x_1^4 + 0.13$

(2) $y = 0.24 * x_1^3 + 0.23 * x_1^4 + 0.39$

(3) $y = 0.75 * x_1^2 + 0.30 * x_1^3 + 0.35 * x_1^4 + 0.334$

(4) $y = 0.02 * x_1 + 0.13 * x_1^2 + 0.301 * x_1^3 + 0.32 * x_1^4 + 0.91$

Similarity to correct form & Prediction Accuracy



Wilcoxon Rank Sum Test

Similarity to Correct Form & Prediction accuracy

Results on 2D data

Example polynomials:

order 1 polynomial:

(1) $y = 0.62 * x_2 - 0.854$

order 2 polynomials:

(1) $y = 0.22 * x_1^2 + 0.05$

(2) $y = 0.12 * x_1 - 0.25 * x_1 * x_2 + 0.4$

order 3 polynomials:

(1) $y = 1.67 * x_1^2 * x_2 + 0.46$

(2) $y = 0.17 * x_1 * x_2 + 0.369 * x_2^3 - 0.3$

(3) $y = 0.03 * x_2 - 0.36 * x_1^2 + 0.22 * x_2^3 + 0.42$

order 4 polynomials:

(1) $y = 2.88 * x_1^2 * x_2^2 + 0.15$

(3) $y = 0.4978 * x_1 * x_2^3 - 0.08 * x_1^4 + 0.36$

(3) $y = 2.19 * x_1 * x_2^2 - 0.87 * x_2^3 + 0.87 * x_1^2 * x_2^2 + 0.39$

(4) $y = 0.13 * x_2 - 1.313 * x_1 * x_2 - 0.1 * x_1^3$

$0.4926 * x_1^2 * x_2^2 + 0.19$

Number of times correct form is discovered:

Standalone GP-SR (1 minute)

Order of the Polynomial	Bases 1	2	3	4
1	30	-	-	-
2	30	29	-	-
3	30	22	15	-
4	30	20	10	2

FFX runs on with unary (x_i) and binary interactions ($x_i * x_j$) (average run time: 9 seconds)

Order of the Polynomial	Bases 1	2	3	4
1	30	-	-	-
2	30	16	-	-
3	0	0	0	-
4	0	0	0	0

FFX/GP-SR runs (1 minute GP run on FFX-generated dataset)

Order of the Polynomial	Bases 1	2	3	4
1	30	-	-	-
2	30	30	-	-
3	30	19	14	-
4	30	20	11	3

Correctness of the polynomial form

Results on 2D data

Example polynomials:

order 1 polynomial:

$$(1)\quad y = 0.62 * x_2 - 0.854$$

order 2 polynomials:

$$(1)\quad y = 0.22 * x_1^2 + 0.05$$

$$(2)\quad y = 0.12 * x_1 - 0.25 * x_1 * x_2 + 0.4$$

order 3 polynomials:

$$(1)\quad y = 1.67 * x_1^2 * x_2 + 0.46$$

$$(2)\quad y = 0.17 * x_1 * x_2 + 0.369 * x_2^3 - 0.3$$

$$(3)\quad y = 0.03 * x_2 - 0.36 * x_1^2 + 0.22 * x_2^3 + 0.42$$

order 4 polynomials:

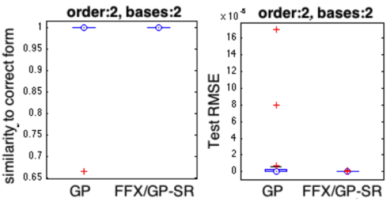
$$(1)\quad y = 2.88 * x_1^2 * x_2^2 + 0.15$$

$$(3)\quad y = 0.4978 * x_1 * x_2^3 - 0.08 * x_1^4 + 0.36$$

$$(3)\quad y = 2.19 * x_1 * x_2^2 - 0.87 * x_2^3 + 0.87 * x_1^2 * x_2^2 + 0.39$$

$$(4)\quad y = 0.13 * x_2 - 1.313 * x_1 * x_2 - 0.1 * x_1^3$$
$$0.4926 * x_1^2 * x_2^2 + 0.19$$

Similarity to correct form & Prediction Accuracy



Wilcoxon Rank Sum Test

Similarity to Correct Form & Prediction accuracy

Results on 3D data

Example polynomials:

order 1 polynomial:

$$(1)\quad y = 0.746 * x_3 + 0.8268$$

order 2 polynomials:

$$(1)\quad y = 0.54 * x_3^2 + 0.4$$

$$(2)\quad y = 0.8651 * x_1 - 0.61 * x_2^2 - 0.30$$

order 3 polynomials:

$$(1)\quad y = 0.84 * x_1 * x_2 * x_3 - 0.86$$

$$(2)\quad y = 0.93 * x_1 * x_2 - 0.46 * x_3^3 + 0.88$$

$$(3)\quad y = 0.04 * x_2 - 0.18 * x_2 * x_3 - 0.01 * x_1 * x_2^2 + 0.3$$

order 4 polynomials:

$$(1)\quad y = 0.20 * x_1 * x_2^3 + 0.91$$

$$(2)\quad y = 0.73 * x_1^2 * x_2 - 0.07 * x_1^2 * x_2 * x_3 + 0.39$$

$$(3)\quad y = 1.2 * x_1 * x_2 + 0.68 * x_1^2 * x_2 + 0.48 * x_1^2 * x_2 * x_3 + 0.41$$

$$(4)\quad y = 0.35 * x_3 - 0.32 * x_2 * x_3 - 0.35 * x_1 * x_2^2 - 0.39 * x_3^4 + 0.24$$

Number of times correct form is discovered:

Standalone GP (1 minute)

Order of the Polynomial	Bases 1	Bases 2	Bases 3	Bases 4
1	30	-	-	-
2	30	25	-	-
3	30	25	9	-
4	30	13	12	3

FFX runs with unary (x_i) and binary interactions ($x_i * x_j$) (average run time: 12 seconds)

Order of the Polynomial	Bases 1	Bases 2	Bases 3	Bases 4
1	30	-	-	-
2	29	16	-	-
3	0	0	0	-
4	0	0	0	0

FFX/GP-SR runs (1 minute GP run on FFX-generated dataset)

Order of the Polynomial	Bases 1	Bases 2	Bases 3	Bases 4
1	30	-	-	-
2	30	26	-	-
3	30	28	14	-
4	30	17	12	6

Correctness of the polynomial form

Results on 3D data

Example polynomials:

order 1 polynomial:

$$(1)\quad y = 0.746 * x_3 + 0.8268$$

order 2 polynomials:

$$(1)\quad y = 0.54 * x_3^2 + 0.4$$

$$(2)\quad y = 0.8651 * x_1 - 0.61 * x_2^2 - 0.30$$

order 3 polynomials:

$$(1)\quad y = 0.84 * x_1 * x_2 * x_3 - 0.86$$

$$(2)\quad y = 0.93 * x_1 * x_2 - 0.46 * x_3^3 + 0.88$$

$$(3)\quad y = 0.04 * x_2 - 0.18 * x_2 * x_3 - 0.01 * x_1 * x_2^2 + 0.3$$

order 4 polynomials:

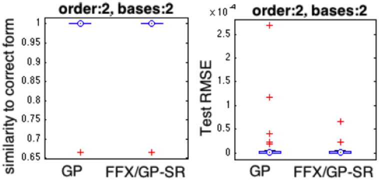
$$(1)\quad y = 0.20 * x_1 * x_2^3 + 0.91$$

$$(2)\quad y = 0.73 * x_1^2 * x_2 - 0.07 * x_1^2 * x_2 * x_3 + 0.39$$

$$(3)\quad y = 1.2 * x_1 * x_2 + 0.68 * x_1^2 * x_2 + 0.48 * x_1^2 * x_2 * x_3 + 0.41$$

$$(4)\quad y = 0.35 * x_3 - 0.32 * x_2 * x_3 - 0.35 * x_1 * x_2^2 - 0.39 * x_3^4 + 0.24$$

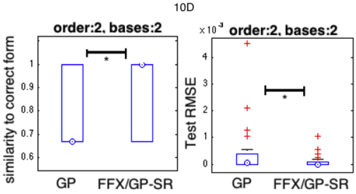
Similarity to correct form & Prediction Accuracy



Wilcoxon Rank Sum Test

Similarity to Correct Form & Prediction accuracy

Results on 10D & 25D data (2nd order polynomials)

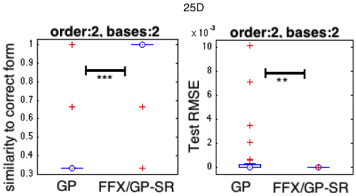


$$y = 0.7 * x_3 - 0.23 * x_9 * x_7 + 0.2$$

where

$$x_i \in x_1, \ldots, x_{10}$$

correctness of polynomial form:
FFX: 10/30, GP-SR: 14/30, FFX/GP-SR: 22/30



$$y = 0.7 * x_3 - 0.23 * x_9 * x_7 + 0.2$$

where

$$x_i \in x_1, \ldots, x_{25}$$

correctness of polynomial form:
FFX: 18/30, GP-SR: 1/30, FFX/GP-SR: 26/30

The hybrid: significantly more similar to the hidden true polynomials & significantly more predictive as dimensionality increases

Discussion

Results show:

- For low dimensional (1 -3) data, GP-SR is competitive with state-of-the-art deterministic ML on our benchmark data suite
- As dimensionality increases (10/25), the hybrid approach wins because hybrid takes advantage of feature extraction provided by the deterministic ML

Therefore, we prove:

- GP-SR is competitive on toy problems, but when dimensionality increases it struggles,
- Hybridize and you shall win!

Discussion

Results show:

- For low dimensional (1 -3) data, GP-SR is competitive with state-of-the-art deterministic ML on our benchmark data suite
- As dimensionality increases (10/25), the hybrid approach wins because hybrid takes advantage of feature extraction provided by the deterministic ML

Therefore, we prove:

- GP-SR is competitive on toy problems, but when dimensionality increases it struggles,
- Hybridize and you shall win!

Current and Future Work

- Real-world problems: 52 dimensional fMRI dataset (presented at GPTP 2013)
- Other ways of hybridizing the deterministic and GP-based methods for Symbolic Regression
- Comparing all three approaches

Acknowledgments

This research was supported by a DARPA grant #FA8650-11-1-7155

The authors acknowledge the Vermont Advanced Computing Core which is supported by NASA (NNX 06AC88G), at the University of Vermont for providing High Performance Computing resources that have contributed to the research results reported within this paper

Questions ?