Micro-Step Time-Series Regression

Insights from System Identification Using Symbolic Regression

Hengzhe Zhang, Alberto Tonda, Qi Chen, Bing Xue, Evelyne Lutton, Mengjie Zhang

February 28, 2025

Outline

Introduction

Time-Series Regression

- **Time-series regression** is fundamental for modeling real-world phenomena
- Standard approach: predict $\hat{y}_{t+1}$ using past observations:

$$\hat{y}_{t+1} = f(\Phi(y_t, y_{t-1}, \dots, y_{t-p+1}))$$

- Φ represents feature construction via genetic programming (GP)

Genetic Programming for Time-Series

- GP evolves interpretable models capturing complex temporal dependencies
- Applications include:
  - Quality of service forecasting [1]
  - Streamflow prediction [2]
  - Financial time-series analysis
- Key gap: Most approaches use fixed time steps determined by the problem

[1]Fanjiang et al., "Time series QoS forecasting for Web services using multi-predictor-based genetic programming," *IEEE Trans. Serv. Comput.*, 2020.

[2]Mehr and Gandomi, "MSGP-LASSO: An improved multi-stage genetic programming model for streamflow prediction," *Inf. Sci.*, 2021.

Dynamic Systems and ODEs

- Dynamic systems modeled by ODEs:

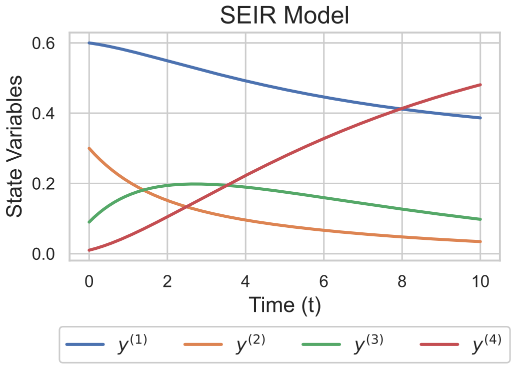
$$\frac{dy^{(i)}}{dt} = f_i(t, y^{(1)}, y^{(2)}, \ldots, y^{(n)})$$

- Euler method approximation:

$$y_{n+1} = y_n + h \cdot f(t_n, y_n)$$

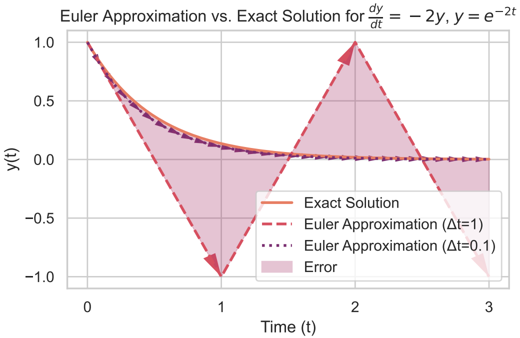
- Problem: First-order Euler method has limited accuracy for complex dynamics

Key Challenge: Large time steps inadequately capture continuous dynamics, leading to errors



Motivation: Euler Method Limitations

- Even with known ground truth ODEs, direct prediction over large intervals causes significant errors
- Example: ODE $\frac{dy}{dt} = -2y$
- Traditional prediction: $y_{t+1} = y_t - 2y_t$
- This large-step approximation diverges from the true solution



Our Solution: Decompose predictions into smaller intervals for better approximation

Contributions

1. **Analysis of step size impact**:
   - Demonstrated that large step sizes cause prediction errors
   - Showed that even with known ground truth, accuracy suffers
2. **Micro-step regression technique**:
   - Novel data augmentation using linear interpolation
   - Enriches training data for GP feature construction
   - Allows capture of finer temporal patterns
3. **Empirical validation**:
   - Tested on 100 datasets from M4 forecasting benchmark
   - Demonstrated significant improvements in accuracy
   - Evolved more compact, interpretable models

Motivation

Impact of Time Step Size

Table: Impact of Reduced Discretization Interval on R^2 Performance

System Id	**State**	**Trajectory**	R^2 **for** $\Delta_t \approx 0.06$	R^2 **for** $\Delta_t \approx 0.03$	R^2 **for** $\Delta_t \approx 0.015$
11	x_0	1	0.002478	0.769415	0.948090
		2	0.957637	0.990291	0.997697
26	x_0	1	0.531413	0.896984	0.976729
		2	0.838427	0.964003	0.991650
	x_1	1	0.670316	0.926977	0.983374
		2	0.884917	0.974147	0.993958

- Key insight: Even with known ground truth, large time steps lead to poor predictions
- Reducing the time step dramatically improves accuracy
- Limitation: In real applications, we can't simply collect data at finer intervals

Proposed Approach

Micro-Step Time-Series Regression

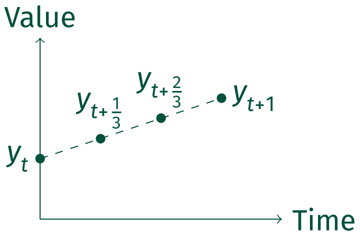
Key idea: Decompose predictions into smaller intervals

1. **Data augmentation:** Linear interpolation between points

$$y_{t+\frac{i}{k+1}} = y_t + \frac{i}{k+1} \cdot (y_{t+1} - y_t)$$

2. **Feature construction:** GP evolves features on augmented data
3. **Prediction:** Aggregate micro-step predictions

$$\hat{y}_{T+h} = y_T + \sum_{i=1}^{h} \sum_{j=1}^{k+1} \Delta\hat{\tilde{y}}_{T+i,j}$$



Evolutionary Framework

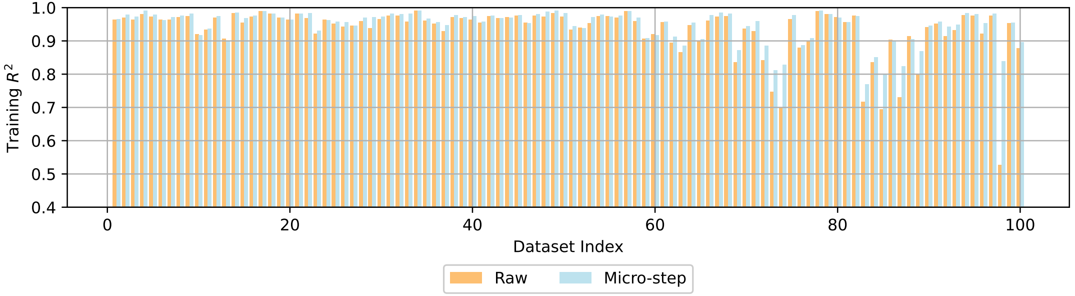
1. **Population Initialization**:
   - Multi-tree GP individuals
   - Ramped-half-and-half initialization
   - Function set: Mathematical operators
   - Terminal set: Lag features $y_{t-1}, \dots, y_{t-p}$
2. **Fitness Evaluation**:
   - Construct m features from each individual
   - Train ARIMA model on constructed features
   - Evaluate on augmented training data
3. **Selection and Variation**:
   - ϵ-Lexicase selection
   - Subtree crossover and mutation
4. **Archive Maintenance**:
   - Store best individual for final predictions

Experimental Results

Experimental Settings

- **Datasets**: 100 time-series from M4 benchmark
- **Baseline Methods**:
  - Raw GP (without micro-step)
  - ElasticNet, Decision Trees, Random Forests, XGBoost, SVR
  - ODEFormer (state-of-the-art for system identification)
- **Evaluation Protocol**:
  - Training/test splits from M4
  - Subsampled to 6-hour intervals
  - Forecast horizon: 8 steps (48 hours)
  - 30 repeated runs with different random seeds
- **Parameters**:
  - Population size: 200
  - Generations: 100
  - Augmentation parameter $k = 1$

Training Performance: Comparison



- Micro-step regression significantly improves training R^2 across datasets
- The improvement demonstrates that data augmentation effectively decomposes the regression task
- By predicting smaller changes, the model better captures underlying patterns

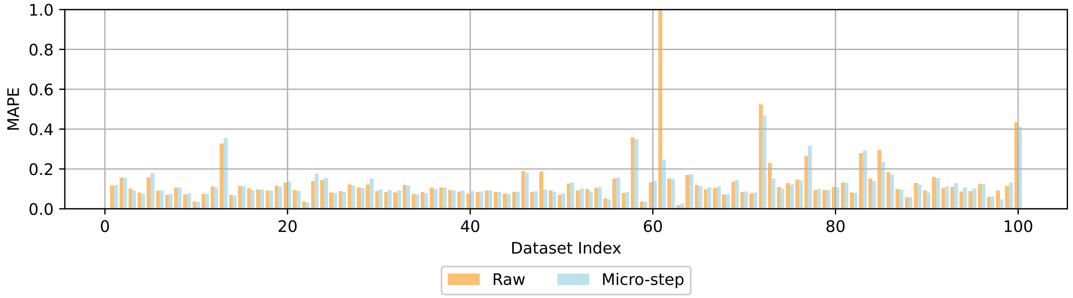
Training Performance: Statistical Analysis

Table: Statistical comparison of training R^2 across 100 datasets

	GP	ElasticNet	DT	RF	XGB	SVR
Micro-GP	73(+)/25(=)/2(-)	100(+)/0(=)/0(-)	0(+)/0(=)/100(-)	0(+)/0(=)/100(-)	0(+)/0(=)/100(-)	100(+)/0(=)/0(-)
GP	—	100(+)/0(=)/0(-)	0(+)/0(=)/100(-)	0(+)/0(=)/100(-)	0(+)/0(=)/100(-)	100(+)/0(=)/0(-)

- Micro-GP significantly outperforms GP on 73% of datasets
- Both GP variants outperform ElasticNet and SVR on all datasets
- Tree-based models (DT, RF, XGB) achieve higher R^2 by memorizing training points
- Tree models can precisely memorize by partitioning the feature space until each region contains only one point

Test Performance: Comparison



- Micro-step regression improves generalization on unseen data
- Lower MAPE values indicate better forecasting accuracy across the prediction horizon
- The improvement is particularly notable on datasets with fine-grained temporal dynamics

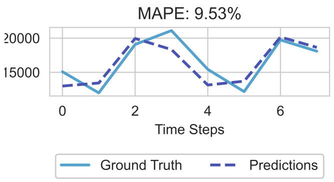
Test Performance: Statistical Analysis

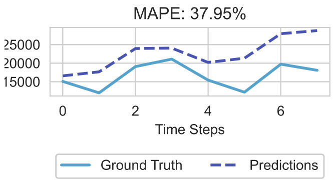
Table: Statistical comparison of test MAPE across 100 datasets

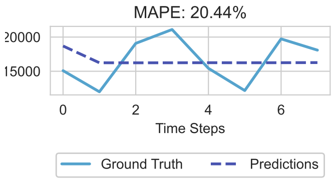
	GP	ElasticNet	DT	RF	XGB	SVR	ODEFormer
Micro-GP	21(+)/74(=)/5(-)	32(+)/44(=)/24(-)	51(+)/22(=)/27(-)	35(+)/27(=)/38(-)	38(+)/25(=)/37(-)	96(+)/3(=)/1(-)	98(+)/1(=)/1(-)
GP	—	17(+)/48(=)/35(-)	30(+)/48(=)/22(-)	20(+)/43(=)/37(-)	25(+)/34(=)/41(-)	74(+)/23(=)/3(-)	76(+)/22(=)/2(-)

- Micro-GP significantly outperforms GP on 21% of datasets
- Micro-GP dramatically outperforms ODEFormer (98%) and SVR (96%)
- Mixed results against other ML methods, suggesting dataset-specific performance
- ODEFormer struggles with real-world data despite being pretrained on synthetic systems

Prediction Visualization



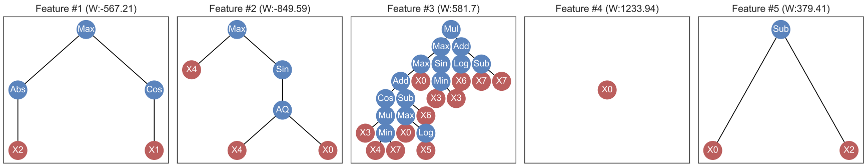




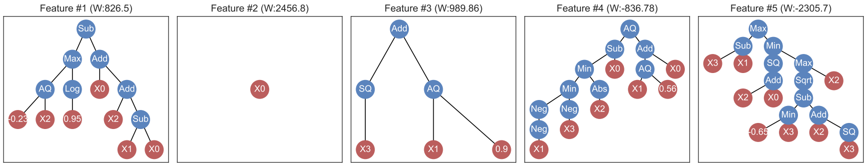
Micro-Step Regression — Raw Regression — ODEFormer

- Micro-step regression maintains accuracy across prediction horizon
- Raw regression deteriorates quickly after first step
- ODEFormer fails to capture underlying patterns

Evolved Models



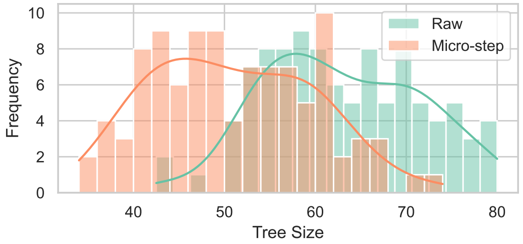
Micro-Step Features

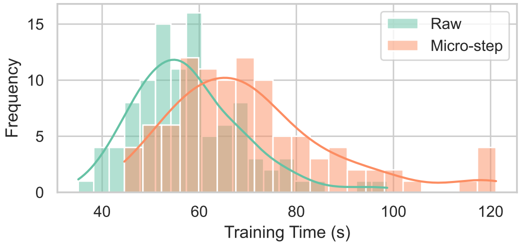


Raw Features

- Features evolved from micro-step data are simpler
- Raw data requires more complex features to capture larger changes
- Simpler features generalize better to unseen data

Model Size and Training Time





Model Size (Tree Nodes)

Training Time (seconds)

- Micro-step regression produces smaller trees
- Only modest increase in training time despite doubled data
- Simplicity of evolved models offsets computational costs

Conclusions

CONCLUSIONS

- **Key Insight**: Large time steps inadequately capture continuous dynamics
- **Contribution**: Micro-step time-series regression technique
  - Decomposes predictions into smaller intervals
  - Uses linear interpolation for data augmentation
  - Allows GP to evolve simpler, more effective features
- **Results**:
  - Improved accuracy on 21% of test datasets
  - Evolved smaller, more interpretable models
  - Outperformed specialized dynamics modeling approaches
- **Future Work**:
  - Automatic determination of optimal step size
  - Extension to multivariate time-series
  - Application to other domains with continuous dynamics

Thank You!

GitHub Project: https://github.com/hengzhe-zhang/EvolutionaryForest/blob/master/experiment/methods/MicroSR.py