# PREDICTIVE MODELING OF I/O CHARACTERISTICS IN HIGH PERFORMANCE COMPUTING SYSTEMS

Thomas C. H. Lux

Dept. of Computer Science
Virginia Polytechnic Institute
& State University
Blacksburg, VA 24061
tchlux@vt.edu

Layne T. Watson

Dept. of Computer Science
Dept. of Mathematics
Dept. of Aerospace & Ocean Eng.
Virginia Polytechnic Institute
& State University

Tyler H. Chang
Jon Bernard
Bo Li

Dept. of Computer Science
Virginia Polytechnic Institute
& State University

Li Xu

Dept. of Statistics
Virginia Polytechnic Institute
& State University

Godmar Back
Ali R. Butt
Kirk W. Cameron

Dept. of Computer Science
Virginia Polytechnic Institute
& State University

Yili Hong

Dept. of Statistics
Virginia Polytechnic Institute
& State University

## ABSTRACT

Each of high performance computing, cloud computing, and computer security have their own interests in modeling and predicting the performance of computers with respect to how they are configured. An effective model might infer internal mechanics, minimize power consumption, or maximize computational throughput of a given system. This paper analyzes a seven-dimensional dataset measuring the input/output (I/O) characteristics of a cluster of identical computers using the benchmark IOzone. The I/O performance characteristics are modeled with respect to system configuration using multivariate interpolation and approximation techniques. The analysis reveals that accurate models of I/O characteristics for a computer system may be created from a small fraction of possible configurations, and that some modeling techniques will continue to perform well as the number of system parameters being modeled increases. These results have strong implications for future predictive analyses based on more comprehensive sets of system parameters.

*SpringSim-HPC, 2018 April 15-18, Baltimore, MD, USA;* 

**Keywords:** Regression, approximation, interpolation, performance modeling

## 1 INTRODUCTION

Performance tuning is often an experimentally complex and time-intense chore necessary for configuring HPC systems. The procedures for this tuning vary largely from system to system and are often subjectively guided by the system engineer(s). Once a desired level of performance is achieved, HPC systems often remain in that base configuration with all subsequent modifications being incremental. Future changes made are often associated with updates and job-specific customizations. In the case that a system has changing workloads or non-stationary performance objectives that range from maximizing computational throughput to minimizing power consumption and system variability, it becomes obvious that a more effective and automated tool is needed for configuring systems. This scenario presents a challenging and important application of multivariate approximation and interpolation techniques.

1. The value of multivariate Modeling
2. The data context
3. The proposed method for using multivariate models
4. The impact of effective models

This paper compares five multivariate approximation techniques that operate on inputs in $\mathbb{R}^d$ (vectors of $d$ real numbers) and produce predicted responses in $\mathbb{R}^1$. In order to provide coverage of the varied mathematical strategies that can be employed to solve the continuous modeling problem, three of the techniques are regression-based and the remaining two techniques produce interpolants. The sections below outline the mathematical formulations of each technique and provide computational complexity bounds.

### 1.1 Approximation

#### 1.1.1 Multivariate Adaptive Regression Splines

#### 1.1.2 Multi-Layer Perceptron Regressor

#### 1.1.3 Support Vector Regressor

### 1.2 Interpolation

#### 1.2.1 Delaunay

#### 1.2.2 Linear Shepard

The remainder of the paper is broken up into _ parts. *(will fill in the parts and descriptions once they are finalized)*

Table 1:

## 2 RELATED WORK

1. Not sure how much to include here? Shooting for thoroughness or simply necessary coverage? How much background should I expect the readers of this paper to have in the "multivariate modeling of systems" area?

## 3 METHODOLOGY

### 3.1 Data

### 3.2 Dimensional Analysis

This work utilizes an extension to standard k-fold cross validation that allows for a more thorough investigation of the expected model performance in a variety of real-world situations. Alongside randomized splits, two extra components are considered: the amount of training data provided, and the dimension of the input data. It is important to consider that algorithms which perform well with less training input also require less experimentation. Although, the amount of training data required may change as a function of the number of input dimensions and this needs to be studied as well.

1. Cycling the categorical settings
2. Selecting subsets of 1,2,3 up to 4 dimensions
3. Cycling different training : testing ratios (5:95 $\rightarrow$ 95:5)
4. Generating 200 random training : testing splits
5. Ensuring the testing points are on/inside the convex hull of the training.
6. Ensuring the training points are well-spaced.

### 3.3 Prediction

1. For each file generated from the dimensional analysis, train on the training data, evaluate at the testing data points

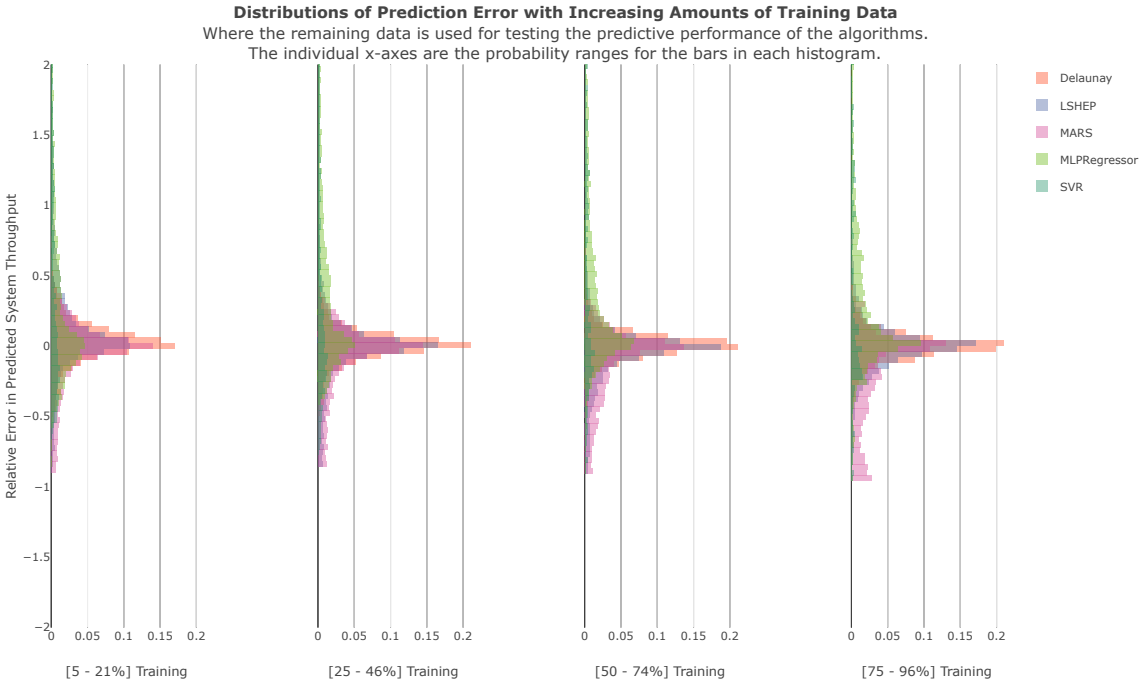


Figure 1: In this figure, it can be seen how the distribution of prediction errors for each algorithm is affected by the quantity of training data provided. Notice that for all amounts of training data, Delaunay has the largest likelihoods around 0 error.

| Algorithm | Input Dimension | Mean Absolute Error | Mean Relative Error |
|---|---|---|---|
| Delaunay | 1 | 2060267 | 0.03 |
| | 2 | 2916224 | 0.07 |
| | 3 | 3001794 | 0.09 |
| | 4 | 3134779 | 0.10 |
| LSHEP | 1 | 6353148 | 0.06 |
| | 2 | 13038882 | 1.52 |
| | 3 | 5446207 | 1.13 |
| | 4 | 5133648 | 1.07 |
| MARS | 1 | 2176368 | 0.03 |
| | 2 | 6213369 | 0.61 |
| | 3 | 4024964 | 0.65 |
| | 4 | 4497414 | 1.01 |
| MLPRegressor | 1 | 2461097 | 0.06 |
| | 2 | 8883580 | 0.81 |
| | 3 | 11190977 | 2.03 |
| | 4 | 11831721 | 2.20 |
| SVR | 1 | 19306731 | 0.96 |
| | 2 | 78213963 | 18.49 |
| | 3 | 97761266 | 31.27 |
| | 4 | 113551300 | 37.2 |

Table 2: Most models experience a decay in predictive performance as the dimension of the data increases. This is expected because higher dimension input has exponentially more possible patterns to identify. The MLP Regressor and SVR experience the worst decay in performance with increasing dimension.

| **Algorithm** | **Training Data** | **Mean Absolute Error** | **Mean Relative Error** |
|---|---|---|---|
| Delaunay | [5-21]% | 5274933 | 0.11 |
| | [25-46]% | 2898330 | 0.08 |
| | [50-74]% | 1518477 | 0.07 |
| | [75-96]% | 886346 | 0.08 |
| LSHEP | [5-21]% | 13218469 | 2.53 |
| | [25-46]% | 4936381 | 0.85 |
| | [50-74]% | 2251409 | 0.26 |
| | [75-96]% | 1415034 | 0.15 |
| MARS | [5-21]% | 6543153 | 0.61 |
| | [25-46]% | 4324916 | 0.80 |
| | [50-74]% | 3115943 | 0.95 |
| | [75-96]% | 2591451 | 0.74 |
| MLPRegressor | [5-21]% | 17348982 | 2.58 |
| | [25-46]% | 12504957 | 2.29 |
| | [50-74]% | 5292177 | 1.17 |
| | [75-96]% | 2996253 | 1.02 |
| SVR | [5-21]% | 170906944 | 49.45 |
| | [25-46]% | 99851846 | 31.28 |
| | [50-74]% | 51565405 | 19.63 |
| | [75-96]% | 27125130 | 12.6 |

Table 3: All models experience a reduction in error with increasing amounts of training data. This suggests that the data being modeled is pattern-dense, over-fitting is *not* occurring, and that oversimplifications will tend towards worse predictions.