High Fidelity Approximation of Slow Simulators Using Machine Learning for Real-time Simulation/Optimization

Sudip Regmi
Hinman Box 2999
Dartmouth College
Hanover, NH 03755
Sudip.Regmi@Dartmouth.edu

Larry M. Deschaine, PE
Science Application International Corporation/Chalmers
Suite 200, 360 Bay Street
Augusta, GA 30901
Larry.M.Deschaine@alum.mit.edu

Sharad R. Regmi, PE
Science Applications International Corporation
11251 Roger Bacon Drive
Reston , VA 20190
Sharad.R.Regmi@saic.com

Keywords: Simulation, Machine Learning, Optimization

Abstract

Simulation and optimization of industrial processes is cost effective and profit productive. Often, high fidelity models require extensive resources to code and require long execution times. In this work, we examine using machine learning techniques to replace simulation models with high fidelity approximations. We test linear genetic programming, linear regression, and machine learning paradigms. The results show that high fidelity approximations (R^2 of 0.99) are possible that execute in a fraction of the time required by the original simulator. These solutions are coded into web services so that a plant manager can input standard information into a user friendly web page, but produce results in a few milliseconds as opposed to hours. This advantage allows for real-time dynamic planning and optimization on the plant floor.

INTRODUCTION

We investigate four industrial strength machine learning paradigms: Multiple Linear Regression, Classification and Regression Trees, Multivariate Adaptive Regression Splines, and Linear Genetic Programming. The best model will be chosen from among them on the basis of R-squared measure.

The purpose for doing this is to turn this chosen model into a real time dynamic planning and optimization system. Here, we use an industrial data set, but the idea is extensible, without much change, to many optimization process required elsewhere. In an industrial plant, with these capabilities we can envision a supervisor making instant decisions for inputs of various goods for maximum profit according to the parameters which the optimizer would deliver.

MACHINE LEARNING

Machine learning is the study of computer algorithms that manipulate models improving them automatically through generations of experience to produce a single model highly effective in modeling the data provided. Models are necessary in all fields. Thus, at any time when a data set is available and the relationships are unknown, this method will help us to produce satisfying models. So, the method can be used to meet the demands of many industrialized or financial process. The machine learning methods we use have no capability of distinguishing between the physical meanings of each and every input variable.

So, the technique can be applied to any process to optimize the output variable. The machine learning functions calibrate themselves as they work and produce a relationship between the given set of data to produce models that have an impressive correlation beyond the example data provided to it. The model that we generate from the machine learning process is then an effective tool to optimize any number of inputs in the range of our needs. We will show the flexibility we have in using these optimizing models. As an example, we deployed the model in a web-based, user-friendly interface.

MULTIPLE LINEAR REGRESSION

Multiple Linear Regression (MLR) can be regarded as a basic machine learning technique. We have used MLR in our analysis in this paper for two reasons. One, it is a powerful statistical technique which is capable of discovering the underlying linearity in the data set, producing good results in relationships like those that arise in physical sciences and economics. Furthermore, MLR will form the basic for comparison with other advanced methods that we will describe later. All of the methods share the statistical measure: R-squared.

MLR seeks to find a linear relationship between the output and a given number of input variables. If Y is the output and X1, X2 … XN are the input variables, then multiple linear regression finds the constants in the following relationship:

$$Y = C1(X1) + C2(X2) + \dots + CN (XN)$$

Therefore the sum of squares of the differences between the actual output and the predicted output is minimized. This is the R-square correlation measure. An R-square of at least 0.6 is considered to begin to show a statistically valid relationship.

To illustrate the meaning of R^2, a two-dimensional example with $R^2 = 0.92$ is shown in Figure 1.

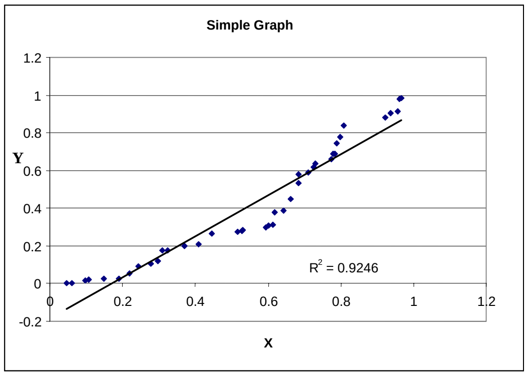


Figure 1. Illustration of the R^2 Measure of Fitness.

CART

Classification and Regression Trees (CART) is one the methods that we have used in this analysis. CART software, available from Salford Systems, makes use of decision trees in order to discover the relationships that exist in our data. CART performs the analysis in two major steps:

- Draws out an overgrown decision tree, and
- Prunes back on the branches of this huge tree, sacrificing little accuracy while getting rid of subtrees that contribute the least to its accuracy.

Maximal Tree

After a training data is fed to the system, it works by going through this data and creating numerous splitting points in each of the variables. At each node of the tree which the system grows, it uses splitting decisions to classify the data into either the right child or the left child. Then, regarding each of its children as the root of a new tree, it finds the maximal tree through recursion. This brute force method will produce a huge decision tree and will uncover any relationship that might exist between the predictor variables and the output. In the next step, the program prunes back the tree to find the optimal one.

Optimal Tree

After making this overgrown and complicated tree, the algorithm works backward pruning all the subtrees that contribute least to the accuracy of the model. In effect, it will create a sizeable tree that will model our data well. In regression, each of the terminal nodes of the tree is assigned a mean value and all records that make up the leaves are then given this value as output. Though this method of classifying data (splitting them and giving them values) is a simple process, it is not feasible to perform this without the help of a computer program. Thus, CART software is particularly helpful. Its strength is its simplicity in modeling the automatically. As we will see later, it has a very high degree of accuracy to boast.

Through this exhaustive search process, not only is the model ready but it also tells us how each of the variables contribute to the output variable.

Figure 2 is an example of a complex regression tree. Each of the terminal nodes represents a data value; each record is classified into one of the terminal node through the decisions made at the non-terminal nodes that lead from the root to that leaf. It was generated by CART.

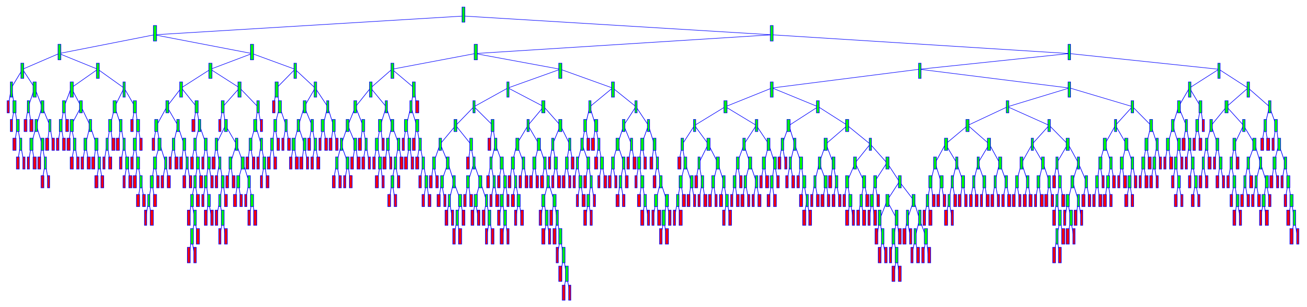


Figure 2. Tree Generated by the CART Algorithm.

MARS

Multivariate Adaptive Regression Splines (MARS), is a machine learning technique that the Salford System created for data-modeling and data-mining purposes. MARS has a Graphic User Interface-(GUI) and is easy and quick to develop good models for data with complicated relationships inferred.

Similar to the methodology of CART, the MARS method also uses a similar, two-step process to generate the model:

1. Generate a large number of basis functions.
2. Prune many of these basis functions without the loss of accuracy to the model.

Basis Functions

The multidimensional data is first scanned through in a brute force manner to find several short intervals where linear regression lines are used to fit the data, these are written as basis functions. The choice of the interval is through brute force and linear regression lines are then used to model the data in between these variables.

Pruning

After having created numerous of these regression splines the method then prunes back on these interval dissection and smoothes them out either by creating larger intervals or finding a compromise for the two endpoints in the data.

Testing

For testing the model that the software system has generated, we generally have two options given to us. We could either specify the file separately as a test input file or we can specify a certain portion of the file for testing. These two methods of testing will ensure that the model that we get will be one that is genuine and isn't just over fitting our data.

LINEAR GENETIC PROGRAMMING

Linear Genetic Programming (LGP) is the controlled evolution of computer models (e.g., programs, instruction) that have the ability to predict the output given the various input using simple mathematical relationships among them (Deschaine 2001, Francone 2003, RMLT 2003). Inspired from nature the methods extensively use genetic crossover, mutations and fitness-based selections to come up with the best model among billions of available ones. The strength of LGP lies in evolution and the sheer number of programs developed by the fast machine-code level algorithm and evolutionary processes. It comes up with the one that is the most suitable for our purpose.

The basic idea of LGP is the evolution of models that are progressively adaptive at each successive generation to the data provided. LGP, unlike the methods of CART and MARS, can be continued infinitely. The user must decide what accuracy is suitable. The three major steps in this process are:

1. Create an initial randomized pool of executable programs.
2. Measure of the fitness of some of these programs to map the provided data accurately. "Winners" and "losers" are selected at this stage from this fitness test.
3. Modify the winning programs by mutation and crossover (techniques inspired by nature) to produce a new generation of candidate programs which replace those that are deemed "losers" in Step 2. The "winners" from Step 2 together with their offspring are then mixed into the population pool to repeat Step 2.

Ultimately, this process, illustrated in Figure 3, produces a whole population fit for modeling our data. Each successive generation will introduce new programs that will perform the task better on the basis of natural selection.

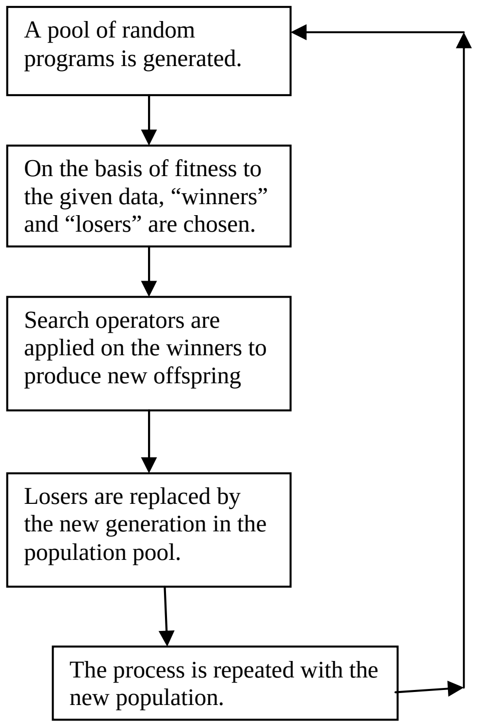

A pool of random programs is generated.
On the basis of fitness to the given data, "winners" and "losers" are chosen.
Search operators are applied on the winners to produce new offspring
Losers are replaced by the new generation in the population pool.
The process is repeated with the new population.

Figure 3: Linear Genetic Programming.

Major Controllable Search Parameters of Genetic Programming

The three major search operators that are employed by Discipulus™ and the ones that we will discuss here are:

- Mutation Rate
- Crossover Rate
- Reproduction Rate

These three are the major methods in which the LGP algorithm will generate the new offspring that will be capable of producing better results.

Each program that is selected as a "winner" in the run of a LGP algorithm is either mutated, reproduced, or program instructions are exchanged with the other winner to create an entirely new program which is put back into the pool. We can vary the rate at which each of these search operators are applied by varying the mutation rate, the reproduction rate, and the cross over rate. These are generally kept at high, and the multiple run option in Discipulus™ software usually takes care of varying each of the parameters and to produce better and better results.

DATA SET

The data set, developed as described in (Deschaine, et. Al, 2002), held 7547 records. With six different variables predicting the output, this was one vertical column of data, and thus we expected that the models that we obtain from the learning methods that are given above will be very accurate. Among the 7547 data, 4967 or 66% of the available data records had zero as its output. The output variable, rapid at first, slows to increase to a little beyond one.

The output variable with just the record number in an ascending order is graphed in Figure 4. Thus, we can see the output is nonzero for only 34% of the available data. This presents us with a challenge in modeling the data. However, we will see that the machine learning technique does well in modeling this data and is able to predict the output accurately throughout the range of the data.

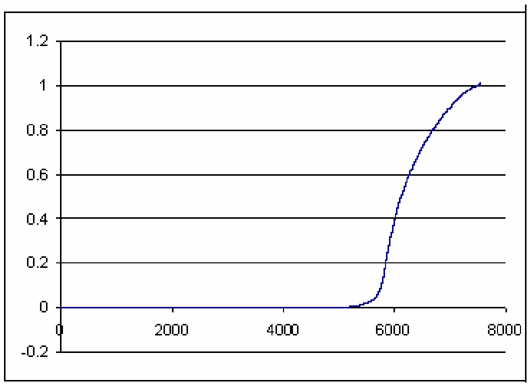


Figure 4. Sorted Depiction of the Output from the Physical Simulator.

Now we will present the resulting accuracy of the models that we obtained from the various methods. We will follow this order:

1. Multiple Linear Regression,
2. Classification and Regression Trees.
3. Multivariate Adaptive Regression Splines, and
4. Linear Genetic Programming.

The data is always divided into thirds. For all methods sans LGP, two-thirds of the set is allocated for training, while the rest is used in testing the model after the methods have completed. However, for LGP, which we modeled using the software Discipulus™, we divided data into three equal parts: the training set, the validation set, and the testing set.

The testing set is always the blind data set for each of the methods and is the best measure of the models produced. With Discipulus™, we also analyzed in two parts: first using the zeros and then without the zeros. Using the zeros produced a better result. Therefore, for the rest of the analysis we only focus on the whole data set. Table 1 summarizes the analysis and the accuracy of the models. The models themselves are available either in the software or exportable as computer programs.

Table 1. Summary of Analysis and Model Accuracy

Methods Used	FITNESS [R^2]	With Zeros	Without Zeros
Linear Genetic Programming	Training	0.99077	0.97844
	Validation	0.99061	0.97928
	Applied	0.98863	0.97264
Multiple Linear Regression	Training	0.4900873402	
	Testing	0.49563209	
Classification and Regression Trees	Training	0.977	
	Testing	0.937435855	
Multivariate Adaptive Regression Splines	Training	0.983	
	Testing	0.922904196	

Table 1 shows the fitness of the models that were developed. Most of the machine learning methods also provide extra information in that they tell which variables are the most important ones.

THE BEST MODEL

The best model is chosen on the basis of the R^2 for the testing set of data. The best model from each method is shown below in the Table 2.

Table 2. Best Model Analysis

Method Used	**R^2 of the Best Model Obtained**
Linear Regression (with zeros)	0.489445672
Discipulus (with zeros)	0.99077
Discipulus (without zeros)	0.97844
CART Analysis (with zeros)	0.937435855
MARS Analysis (with zeros)	0.922904196

The chosen model was the model generated using LGP. This model was then converted into Java™ code using the facility provided by the Discipulus™ software.

Excerpt from the model code is given below:

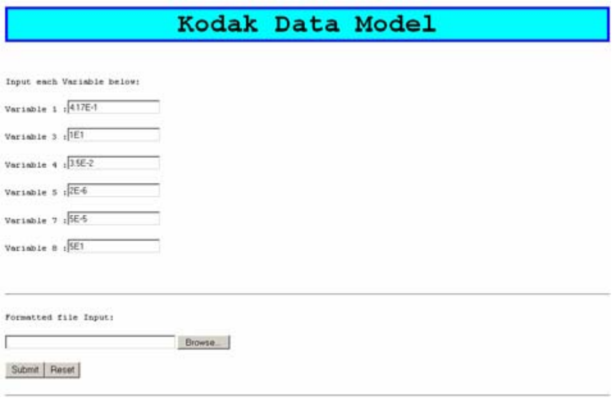
```
f[0] is the output and the f[i] are the inputs:
double DiscipulusJavaFunction(double  [ ] v )
{
    double [] f=new double[8];
    double tmp = 0;
    boolean cflag = false;

    f[0]=f[1]=f[2]=f[3]=f[4]=f[5]=f[6]=f[7]=0;

    f[0]+=v[3];
    f[2]+=f[0];
    f[0]*=f[0];
    f[0]/=v[1];
    f[0]/=1.530829906463623f;
    f[0]=Math.sqrt(f[0]);
    f[0]/=v[2];
    f[1]+=f[0];
    f[0]+=v[3];
    f[0]+=v[3];
    f[0]=Math.sqrt(f[0]);
    f[0]+=f[0];
    f[0]*=f[0];
    f[0]-=f[1];
    f[0]*=v[2];
```

(Note: The entire model is not shown here.)

Web-based Deployment

For ease of model usage, data was converted into a Java™ web-based model. It was then possible to use this model as a service from the web and parameterize the output variable.

Note in Figure 5, we can type input directly into the variable fields that are presented or we can input a whole file, which has formatted input present. In either case, the output generated is in a tabular form.


Kodak Data Model

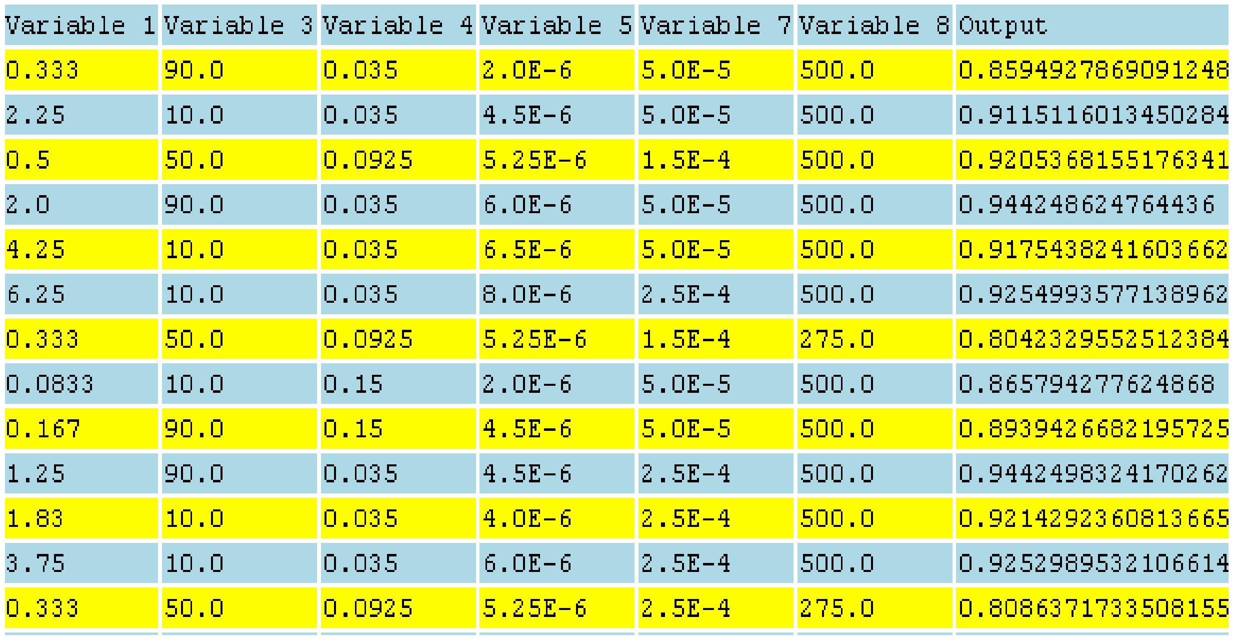
Input each Variable below:

Variable 1 : 4.17E-1
Variable 3 : 1E1
Variable 4 : 3.5E-2
Variable 5 : 2E-6
Variable 7 : 5E-5
Variable 8 : 5E1

Formatted file Input:

Figure 5: Input Screen for the Kodak Data Model

Instantly with the web-based deployment, the result is displayed, as seen in Figure 6.



Variable 1	Variable 3	Variable 4	Variable 5	Variable 7	Variable 8	Output
0.333	90.0	0.035	2.0E-6	5.0E-5	500.0	0.8594927869091248
2.25	10.0	0.035	4.5E-6	5.0E-5	500.0	0.9115116013450284
0.5	50.0	0.0925	5.25E-6	1.5E-4	500.0	0.9205368155176341
2.0	90.0	0.035	6.0E-6	5.0E-5	500.0	0.944248624764436
4.25	10.0	0.035	6.5E-6	5.0E-5	500.0	0.9175438241603662
6.25	10.0	0.035	8.0E-6	2.5E-4	500.0	0.9254993577138962
0.333	50.0	0.0925	5.25E-6	1.5E-4	275.0	0.8042329552512384
0.0833	10.0	0.15	2.0E-6	5.0E-5	500.0	0.865794277624868
0.167	90.0	0.15	4.5E-6	5.0E-5	500.0	0.8939426682195725
1.25	90.0	0.035	4.5E-6	2.5E-4	500.0	0.9442498324170262
1.83	10.0	0.035	4.0E-6	2.5E-4	500.0	0.9214292360813665
3.75	10.0	0.035	6.0E-6	2.5E-4	500.0	0.9252989532106614
0.333	50.0	0.0925	5.25E-6	2.5E-4	275.0	0.8086371733508155

Figure 6: Output Generated by the Kodak Data Model.

Note: The output can easily be copied from this file into spreadsheet programs like OpenOffice™, StarOffice™, and MS Excel™.

CONCLUSION

The above model smoothly executes in milliseconds when the value of the inputs were increased and decreased by 25% of the range values used to develop the LGP model. The initial idea was to link an optimization algorithm to the LGP-derived Java^{TM} code for real-time optimization of plant processes. The Java^{TM} code executes so quickly that simple enumeration provides a suite of optimal operational conditions in seconds. The solution and tool developed around it requires no more skill than is available from personnel who use any common internet browsers, such as Mozilla™ or MSExplorer™.

In the example, we have used the model only to display the output given the variables in a file or a single set of records, but this process could be extended to optimize a certain variable to within a given range.

Using these fast executing models, we can get quick approximation to complex processes if we have a data set available to train one of the machine learning techniques. The model most suitable for our purpose could then be used in the above mentioned fashion, for increased productivity.

REFERENCES

Deschaine, L. M., and Francone, F. D., Design Optimization Integrating the Outer Approximation Method with Process Simulators and Linear Genetic Programming, Joint Conference on Information Science, Research Triangle Park, NC, ISBN 0-9707890-1-7, pages 618-621, March, 2002.

Deschaine, L. M., Patel, J. J., Guthrie, R. G., Grumski, J. T., and Ades, M. J., Using Linear Genetic Programming to Develop a C/C++ Simulation Model of a Waste Incinerator. The Society for Modeling and Simulation International: Advanced Simulation Technology Conference, Seattle, WA, USA. ISBN: 1-56555-238-5, pages 41-48, April 2001.

Francone, F. D., and Deschaine, L.M., Extending the Boundaries of Design Optimization by Integrating Fast Optimization Techniques with Machine-Code-Based Linear Genetic Programming, Information Sciences Journal, Elsevier Press, Amsterdam, The Netherlands In-press: November, 2003.

Register Machine Learning Technologies, Discipulus users guide version 3.0. See http://**www.aimlearning.com** for more details.

Salford Systems. Product Manual Overview for CART and MARS. See **http://www.salford-systems.com** .