

Regression with small data sets: a case study using code surrogates in additive manufacturing

Chandrika Kamath¹  · Ya Ju Fan¹

Received: 31 August 2017 / Revised: 12 January 2018 / Accepted: 26 January 2018 /
Published online: 1 March 2018
© Springer-Verlag London Ltd., part of Springer Nature (outside the USA) 2018

Abstract There has been an increasing interest in recent years in the mining of massive data sets whose sizes are measured in terabytes. However, there are some problems where collecting even a single data point is very expensive, resulting in data sets with only tens or hundreds of samples. One such problem is that of building code surrogates, where a computer simulation is run using many different values of the input parameters and a regression model is built to relate the outputs of the simulation to the inputs. A good surrogate can be very useful in sensitivity analysis, uncertainty analysis, and in designing experiments, but the cost of running expensive simulations at many sample points can be high. In this paper, we use a problem from the domain of additive manufacturing to show that even with small data sets we can build good quality surrogates by appropriately selecting the input samples and the regression algorithm. Our work is broadly applicable to simulations in other domains and the ideas proposed can be used in time-constrained machine learning tasks, such as hyper-parameter optimization.

Keywords Regression · Sampling · Code surrogates · Small data sets

1 Introduction

The recent focus in data mining has been on the analysis of massive data sets in application domains where data are easy to collect, such as mining the web or online collections of text documents and images. At the other extreme are domains where generating each data point is very time consuming or expensive, resulting in much smaller data sets consisting of a few tens or hundreds of samples. An example is the building of *code surrogates* as a faster alterna-

✉ Chandrika Kamath
kamath2@llnl.gov

Ya Ju Fan
fan4@llnl.gov

¹ Lawrence Livermore National Laboratory, 7000 East Avenue, Livermore, CA 94551, USA

tive to computer simulations, which are increasingly being used to complement experiments and observations in domains such as additive manufacturing [21] and climate [1, 29]. These simulations, or “codes,” are computational models that, given the inputs of a physical phenomenon, called the “parameters” of the code, generate the corresponding outputs [35]. Scientists use simulations to select parameters for use in an experiment, to identify the important input parameters, and to understand how uncertainties in input parameters of an experiment affect the uncertainties in the outputs.

Computer simulations can range from the very simple, providing a fast but approximate solution to a problem, to the very complex, that give more accurate results, but take hours or days to run on massively parallel, high-performance systems. Since the number of simulations required to do a full sweep through the range of possible input parameters is exponential in the number of parameters, it is prohibitively expensive to run even a moderately complex simulation to gain an in-depth understanding of a physical phenomenon. To make the problem tractable, scientists often generate the simulation results at a few carefully selected sample points in the input parameter space and then build a data-driven model, called a *code surrogate* that relates the outputs to the inputs [15, 24]. The surrogate is then used to predict the outputs corresponding to the inputs for a new sample point, in effect serving as an interpolation algorithm. Traditionally, simple surrogates or response surfaces in the form of low-order models, such as first or second degree polynomials, were used. However, as the surfaces relating the outputs to the inputs become more complex, methods from machine learning are preferred.

Code surrogates have many uses as inexpensive alternatives to a simulation. Surrogates for simple, but approximate, simulations can be used to identify a viable region in the input parameter space, where an experiment is more likely to work. The points in this viable region could then be used in more complex and longer-running simulations [21], thus making better use of computer resources. If the simulation is more complex and provides results closer to experiments, the surrogates could be used directly in designing experiments. For data sets that are generated incrementally, a code surrogate could indicate the next sample points that should be run to provide the most benefit, in a manner similar to active learning [9]. Finally, surrogates play an important role in uncertainty analyses and sensitivity studies [13].

Code surrogates can be relatively inexpensive to build and use. But, to be effective, their predictions must be accurate enough for the task at hand. This is challenging when we can run the simulation at only a few sample points and the complexity of the function relating the simulation outputs to the inputs is unknown.

In this paper, we show how we can generate accurate surrogates using small data sets by appropriately selecting the sample points and the regression algorithm. We illustrate our work using the task of determining melt-pool characteristics in the domain of additive manufacturing. Our ideas are applicable to simulations in other domains, as well as the problem of hyper-parameter optimization for machine learning algorithms [6]. This problem is typically solved by sampling the potentially high-dimensional, input space of hyper-parameters and selecting the sample that results in the smallest value of an error metric, such as the leave-one-out error. If the calculation of this metric is expensive, which is the case for large data sets, or the number of hyper-parameters is large, the ideas presented in this paper could be applied to solve the problem within a fixed computational budget. Our contributions in this paper are as follows:

- We propose the use of a sampling technique from computer graphics that allows us to randomly place a given number of sample points that are well spread out so they cover the high-dimensional space adequately. We use this technique to generate sample points in

the input parameter space of simulations as well as to sample the hyper-parameter space of one of our regression algorithms to find the optimal hyper-parameters. This sampling can also be used to identify a subset of random-but-equispaced instances from a larger data set for use in time-constrained tasks such as evaluating the metric for hyper-parameter optimization [30].

- We use a practical problem in additive manufacturing to investigate how the accuracy of various regression algorithms changes as we reduce the size of the data sets. We demonstrate that even with a data set containing just tens of samples, there are some regression algorithms that perform quite well. Identifying a good algorithm for such small data sets is difficult as cross validation is an unreliable metric for evaluation; our results provide practitioners some guidance on a choice of algorithms.
- We propose an approach that combines simple, approximate simulations to first identify viable regions, followed by more accurate, but computationally expensive simulations that are run at select sample points, to obtain results at reduced cost. This is possible in many scientific domains, where inexpensive simulations, which do not include all of the science, are often available. This idea is especially applicable in hyper-parameter optimization, where we can use a simple metric to identify regions of viable hyper-parameters, followed by the evaluation of a more accurate metric on a select subset of viable points. We could even build a “surrogate” for the error metric to obtain the optimum hyper-parameters.

This paper is organized as follows: First, in Sect. 2, we briefly describe the application domain of additive manufacturing, the two simulations used in our work, and the sampling used to generate the data sets. In Sect. 3, we describe the algorithms used to build the code surrogates, followed by Sect. 4, where we present and discuss the performance of these algorithms using a variety of metrics. Related work is described in Sect. 5, and we conclude with a summary in Sect. 6.

2 Simulation data from additive manufacturing

We conduct our work using data from simulations in the domain of additive manufacturing, though the ideas presented are applicable to simulations in any domain. Additive manufacturing (AM), or 3-D printing, is a process for fabricating parts, layer-by-layer. In laser powder-bed fusion, each layer in a part is created by spreading a thin layer of powder and using a laser beam to selectively melt the powder in specific locations so that it blends into the layers below. A particular challenge in AM is to understand how the many parameters that control the process affect the properties and quality of a part. These parameters include not only the properties of the material such as thermal conductivity, but also the particle sizes and layer thickness of the powder, as well as the power and speed of the laser. Exploring this high-dimensional input parameter space using experiments is infeasible, and while simulations are a viable alternative, they too can be expensive if they incorporate all the physics involved in AM.

In our earlier application-focused work [21,23], we investigated how the choice of input parameters affected the characteristics (length, depth, and width) of the melt-pool that is formed when the laser melts a layer of powder. Our goal was to select input parameters that created melt pools deep enough to melt through the powder into the substrate below, but not so deep that we waste energy. We considered two models for the simulations:

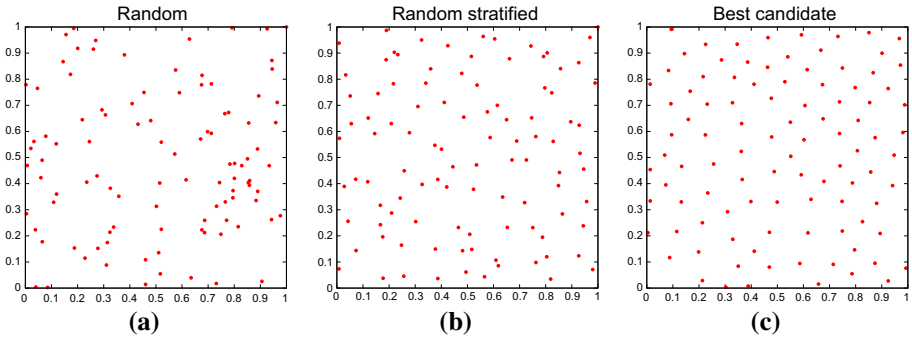


Fig. 1 Sampling in a two-dimensional space: 100 samples distributed using **a** random sampling, **b** stratified random sampling with ten levels along each axis, and **c** best-candidate sampling

- *Eagar–Tsai model* This very simple model [14] considers a Gaussian laser beam moving on a three-dimensional flat plate. The resulting temperature distribution is used to compute the melt-pool characteristics as a function of four input parameters: laser power, laser speed, beam size, and laser absorptivity of the powder, which indicates how much of the laser energy is absorbed by the powder. The model is computationally inexpensive, taking ≈ 1 minute to run on a laptop, enabling a dense sampling of the input parameter space.
- *Verhaeghe model* This is a more expensive physical model that considers various physical phenomena involved in the laser melting of powder on a substrate [37]. This model takes about 1–3 h of computational time (depending on the laser speed) on a small eight-processor cluster. The model has a larger number of parameters, such as the powder layer thickness, the void fraction in the powder, and various material properties, not all of which are known precisely.

We found that the Eagar–Tsai model, while fast, was not accurate enough for use in prediction, while the Verhaeghe model was more accurate, but its computational cost and larger number of parameters made its use prohibitively expensive. This naturally prompted us to consider code surrogates for these simulations.

2.1 Sampling the input space of simulations

To build a code surrogate, we first need to generate the training data. Our two simulation models each have a number of input parameters that we can vary to create sample points at which to run the simulations. As the number of sample points required to adequately sample a parameter space is exponential in the number of parameters [10], we need to select these sample points carefully.

When little is known about a simulation, the initial set of sample points are generated randomly and additional points are added as required. However, as seen in Fig. 1a for a two-dimensional space, a random sampling can result in regions that are over- or under-sampled. Improved sampling techniques have been extensively studied in two related fields—the traditional “design of (physical) experiments” [26] and the contemporary “design and analysis of computer experiments” [15], resulting in a wealth of sampling techniques currently in use [27, 28]. Since one of our physical models is computationally inexpensive and the other is moderately expensive, we consider two sampling schemes that were selected for their simplicity:

- *Stratified random sampling* In this scheme, we first divide each input parameter range into a number of levels and select a sample point randomly in each resulting cell (Fig. 1b). While the spatial distribution of sample points is improved in comparison with a purely random sampling, the number of samples, which is determined by the number of parameters and number of levels in each parameter, can be too large for use with expensive simulations.
- *Best-candidate sampling* The Mitchell's best-candidate sampling [25], originally proposed in computer graphics, places samples randomly as far apart from each other as possible. The method starts by placing the first point randomly. Then, for each new point, it randomly generates a pre-specified number of candidate points and selects as the next sample the candidate that has the largest nearest-neighbor distance to the current set of samples. The process continues until the desired number of samples have been generated. This method allows us to specify an arbitrary number of samples, incrementally add new samples to an existing set, and sub-sample from an existing set. As a result, we can start with a coarse sampling of the simulation input space or the hyper-parameters of an algorithm and then incrementally add more points in selected regions for a progressively denser sampling. We can also use the method to create a random subset of a larger data set, where the instances in the selected subset are far apart from each other, making the subset less biased.

2.2 Data used in the experiments

We generate three data sets generated from the two physical models and the two sampling schemes:

- *Eagar-Tsai-462* This data set was generated using the Eagar-Tsai model and the stratified random sampling. We selected the ranges and number of levels of the four input parameters to match our additive manufacturing machine and the material being used, resulting in 462 simulations [23]. Each data point in this data set consists of the four inputs — laser power, laser speed, laser beam size, and absorptivity of the material — and the corresponding values of melt-pool depth, width, and length obtained from the Eagar-Tsai model.
- *Eagar-Tsai-100* This data set was also generated using the Eagar-Tsai model, but used just 100 sample points generated using the best-candidate sampling scheme. It allows us to compare surrogates using the same, simple, Eagar-Tsai model when the size of the data set is reduced. Each sample point has the four inputs used in the simulation and only the corresponding value of melt-pool depth as it is the most important of the three outputs.
- *Verhaeghe-41* This data set was generated by considering the 41 points in the Eagar-Tsai-100 that had a melt-pool depth larger than 55 microns, a value that indicated a viable parameter set. The Verhaeghe model was then run at these sample points to obtain the melt-pool depth. Each of the 41 data points consists of the four inputs used in the simulation and the corresponding values of melt-pool depth. The remaining inputs were set to standard values for the material. We focused only on the depth as the Verhaeghe model does not include the physics required to reproduce the melt-pool width. This data set allows us to compare experimental results with the predictions from surrogates for the case where a complex simulation was run at a small number of sample points. It also illustrates the approach of combining simple and expensive simulations to reduce

the computational cost as we do not have to run the more expensive simulations at the non-viable points.

In addition to the data we generated from simulations, we also have experimental data from single-track experiments [38], where a single layer of powder is spread on a plate, and the laser is used to create single tracks at specific power and speed values. The plate is then processed to reveal the track cross section from which the depth and width are obtained. As the preprocessing and post-processing of the plate after the creation of the tracks is quite expensive, we have results for just 14 tracks [21,23] which we use in Sect. 4.3 to evaluate the quality of the surrogate built using the Verhaeghe-41 data set.

3 Description of the code surrogates

We next briefly describe the algorithms used to build the code surrogates in our study. They were chosen based on their accuracy, their simplicity and the resulting interpretability, as well as their successful use in various application domains.

3.1 Nearest-neighbor methods

The simplest among regression methods, this class of methods predicts the value at a point in input space as a function of the values of its nearest neighbors in the input space. In our work, we use locally weighted kernel regression (LWKR) [2], where we build a locally linear model, whose coefficients are obtained using a least squares fit. We first scale each input to lie between 0.0 and 1.0. To avoid having to select the number of near neighbors, we define distances between samples using a Gaussian kernel of bandwidth 0.1; this kernel distance reduces the contribution of samples that are farther away. In addition, we use a weighted Euclidean distance as an initial exploratory analysis of the training data using Analysis of Variance (ANOVA) (Sect. 4.1) indicated that this would improve the accuracy of the method.

3.2 Regression trees

For our regression tree algorithm [8], we split the data at a node of the tree using the mean-squared error, MSE, which for a split A on a certain input is defined as

$$\text{MSE}(A) = p_L \cdot s(t_L) + p_R \cdot s(t_R)$$

where t_L and t_R are the subset of samples that go to the left and right, respectively, by the split based on A , p_L , and p_R are the proportion of samples that go to the left and right, and $s(t)$ is the standard deviation of the $N(t)$ output values, c_i , of samples in the subset t :

$$s(t) = \sqrt{\frac{1}{N(t)} \sum_{i=1}^{N(t)} (c_i - \overline{c(t)})^2}$$

where $\overline{c(t)}$ is the mean of the values in subset t . The split at each node of the tree is the one that minimizes MSE across all features for the samples at that node. To avoid over-fitting, we stop growing the tree if the number of samples at a node is less than 5 or the standard deviation of the values of the output variable at a node has dropped below 5% of the standard deviation of the output variable of the original data set. Further, to improve the accuracy, we use an ensemble of 10 trees [20,31,32], where randomization is introduced at a node of the tree by using a random subset of the samples [22] to make the split at the node.

3.3 Multivariate adaptive regression splines

Multivariate adaptive regression splines (MARS) [16] is a regression model for high-dimensional data that combines separate linear models for segments within the input variables in a way that automatically models linear and nonlinear interactions among these variables. In the training phase, the data are partitioned into separate, piece-wise, linear segments of differing slopes, called linear splines, which are then connected smoothly together to build piece-wise curves, called basis functions. The MARS model is a linear combination of these basis functions and their interactions. Given two user-defined parameters—the maximum number of basis functions and the maximum interaction level—MARS first builds a complicated model that over-fits the data, followed by a phase where the model is simplified by removing basis functions that made the least contribution. The best model is determined using the generalized cross validation (GCV) criterion that trades off goodness-of-fit against model complexity:

$$\text{GCV} = \frac{\frac{1}{N} \sum_{i=1}^N [y_i - f(x_i)]^2}{1 - \frac{1}{N} [M + \delta \times \frac{M-1}{2}]},$$

where $f(x_i)$ and y_i are the predicted and actual values at point x_i ; M is the number of basis functions; $\frac{M-1}{2}$ is the number of knots; N is the number of training samples; and δ is a penalty that is about 2 or 3. The numerator is the average of the residual sum of squares. The denominator contains both the number of basis functions and the number of knots, which results in less complex models being selected.

3.4 Support vector regression

The ε -SVR method [36] uses the training data to find a decision function that has at most ε deviation from all of the actual target output values in the training data. In our work, we use the ν -support vector regression (ν -SVR) method [11, 34] that obtains the decision function by formulating an optimization problem in which the risk is reduced by minimizing both the model complexity and training error. A soft margin loss function [5] is used to cope with otherwise infeasible constraints. The decision function obtained by solving the optimization problem is then used to predict the output at a new sample point.

3.5 Gaussian processes

Gaussian process (GP) is a surrogate model that provides not just a prediction, but also an uncertainty on the prediction. A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution [30]. GPs can be considered to be an extension of multivariate Gaussian distributions to infinite dimensions. Since our training data with N samples can be thought of as a single point sampled from an N -variate Gaussian distribution, they can be partnered with a Gaussian process. The mean of this GP is often taken to be zero. The model that is fit to the data using a GP is controlled by the covariance function. We use a squared exponential that has two parameters—one for the maximum allowable covariance and the other a length parameter that determines the extent of influence of each point and therefore controls the smoothness of the interpolation. These parameters are calculated from the training data using a maximum likelihood approach. It is also possible to include a Gaussian noise component in the output variable, but in our current analysis, we have assumed the noise to be zero.

4 Experimental results

We conduct several experiments with the three data sets described in Sect. 2 to evaluate the different surrogates and understand their performance on a small training set. Specifically, we:

- perform an exploratory analysis of the data using ANOVA (Sect. 4.1),
- evaluate the accuracy of the different surrogate models on Eagar–Tsai-462 and Eagar–Tsai-100 data sets using cross validation and leave-one-out error metrics (Sect. 4.2), and
- describe how surrogate models built using a small training set perform in prediction (Sect. 4.3).

Recall that we use three data sets—Eagar–Tsai-462, Eagar–Tsai-100, and Verhaeghe-41—with 462, 100, and 41 sample points, respectively. There are four inputs in each data set: the laser power, the laser speed, the beam size, and the absorptivity of the material as a fraction. In the larger Eagar–Tsai data set, we consider three outputs: the melt-pool width, depth, and length, while in the remaining two data sets, we focus only on the depth as it is the most important of the three outputs. The depth is a relatively well-behaved function over most of the domain, though it increases rapidly in a small region where the laser power is high and the laser speed is low. Since this nonlinear region is quite small, it is possible that a small number of samples might not capture the variation in this region adequately. This explains why the Eagar–Tsai-462 data set has melt-pool depth values in the range 12–289 microns, while the smaller Eagar–Tsai-100 data set has depth values in the range 23–101 microns. This is an inherent problem whenever a small number of samples are used to model a rapidly varying function.

4.1 Exploratory analysis using ANOVA

Before presenting the results for the different surrogates described in Sect. 3, we discuss the results of Analysis of Variance (ANOVA) [26] applied to data derived from the Eagar–Tsai-462 data set. ANOVA is a very simple model based on mean values, and the deviation from mean values, used to explain data sets. It is related to linear regression, though it is not a surrogate in the strictest sense. We include ANOVA in this paper as it provides useful insight in improving the performance of nearest neighbor surrogates.

In the ANOVA approach [26], a physical or computational experiment, with a set of input variables (called factors), is run at a select set of values (called levels) of these inputs, and the resulting outputs are analyzed. To generate such a data set from the Eagar–Tsai-462 data, we first identified the minimum and the maximum values of each of the four inputs, then divided this range into four equal segments, and took the midpoint of each segment as the value at that level. These values are listed in Table 1. Next, we generated the output data (the melt-pool length, width, and depth) at these 256 points by using the MARS method applied to the Eagar–Tsai-462 data. We repeated this three times, each time using a random 90% subset of the data to build the MARS model, giving us a total of 768 predictions (three values at each of the 256 points) for each of the three output variables. We then performed a 4-way ANOVA on this data set, focusing on the main effects and first-order interactions of the four input factors. The F-statistic for these are summarized in Table 2, with the larger values indicated in bold. These results suggest that not all inputs contribute equally to the three outputs. Specifically:

Table 1 Input values used for each factor for the ANOVA experiment

Level	Speed	Power	Beam size	Absorptivity
1	0.335325	93.8875	26.125	0.325
2	0.882375	181.0625	28.375	0.375
3	1.429425	268.2375	30.625	0.425
4	1.976475	355.4125	32.875	0.475

The laser speed in m/s, the laser power in Watts, the beam size is the diameter of the Gaussian laser beam in micron, and the absorptivity of the material is a fraction indicating how much of the laser energy is absorbed

Table 2 F-statistic values for main effects (top) and 2-way interaction terms (bottom) from 4-factor ANOVA

Input Parameters	Output variables		
	Depth	Length	Width
Speed	4.11×10^2	4.50×10^{-2}	3.88×10^2
Power	1.12×10^2	2.06×10^3	1.15×10^2
Beam size	6.96×10^{-4}	1.26×10^{-5}	3.52×10^{-3}
Absorptivity	7.35	2.59×10^1	6.70
Speed and power	9.73×10^1	2.13×10^{-2}	1.49×10^2
Speed and beam size	2.20×10^{-2}	2.20×10^{-3}	5.08×10^{-3}
Speed and absorptivity	9.73×10^{-1}	2.49×10^{-3}	5.83×10^{-1}
Power and beam size	1.15×10^{-2}	9.96×10^{-5}	5.47×10^{-3}
Power and absorptivity	1.37×10^{-1}	5.90×10^3	3.38×10^{-2}
Beam size and absorptivity	3.26×10^{-3}	1.27×10^{-5}	9.67×10^{-4}

- The laser speed affects the melt-pool depth and width; the absorptivity affects the melt-pool length; and the laser power affects all three outputs.
- The laser power and laser speed have a strong interaction that affects the melt-pool depth and width, while the laser power and absorptivity interaction strongly affect the length.

We drew the same conclusions in our previous work using feature selection algorithms on the Eagar–Tsai-462 data set [21]. This suggested that to build good surrogate models, methods based on distances in the input space, such as the nearest-neighbor methods, would benefit by using a weighted distance. We, therefore, modified our implementation of the LWKR method to weight the features. The near-optimal feature weights were obtained by using the best-candidate method (Sect. 2.1) to generate random samples in the weight space in the range [0.0, 5.0] and selecting the sample that resulted in the minimum sum of absolute leave-one-out errors on the training set. Table 3 lists the near-optimal weights used in our LWKR model. Note that these weights, especially for the larger data sets, concur with the ANOVA analysis.

4.2 Evaluating the accuracy of surrogate models

We consider two different metrics to evaluate the accuracy of the predictive models. The first is k runs of m -fold cross validation, where the data are divided randomly into m parts, the

Table 3 Near-optimal feature weights for the LWKR surrogate model

Input Parameters	Eagar-Tsai-462			Eagar-Tsai-100	Verhaeghe-41
	Length	Depth	Width	Depth	Depth
Speed	0.93	4.17	4.90	1.23	1.47
Power	1.40	1.52	1.03	0.79	0.20
Beam size	0.03	0.05	0.05	0.10	0.24
Absorptivity	1.05	0.09	0.21	0.08	0.37

Table 4 Error rate of the different surrogate models using relative MSE (equivalent to $1 - R^2$) for five runs of fivefold cross validation for the Eagar-Tsai-462 and the Eagar-Tsai-100 data sets

Surrogate	Relative MSE for 5 runs of fivefold cross validation			
	Eagar-Tsai-462			Eagar-Tsai-100
	Length (%)	Depth (%)	Width (%)	Depth (%)
LWKR (no weighting)	0.04	9.02	9.76	6.40
LWKR (optimal weights)	0.02	0.34	0.19	0.25
Regression tree	1.92	8.95	8.91	27.32
Ensemble of 10 trees	0.85	3.89	4.19	12.43
MARS	0.02	0.72	0.62	0.42
SVR	0.01	10.24	11.08	1.52
GP	0.01	0.02	0.07	0.08

model is trained on $(m - 1)$ parts and evaluated on the part that is held out. This is repeated for each of the m parts. The process is repeated k times, each with a different random partition of the data. The final metric is the average of the accuracy for each of the $k \times m$ parts. We use the relative mean-squared error metric, defined as

$$\sum_{i=1}^n (p_i - a_i)^2 / \sum_{i=1}^n (\bar{a} - a_i)^2$$

where p_i and a_i are the predicted and actual values, respectively, of the i -th sample point in the test data consisting of n points, and \bar{a} is the average of the actual values in the test data. This is essentially the ratio of the variance of the residual to the variance of the target (that is, actual) values and is equal to $(1.0 - R^2)$, where R^2 is the coefficient of determination. The second metric is the leave-one-out (LOO) prediction, where a model, which is built using all but one of the sample points, is used to predict the value at the point that is held out. The plot of predicted versus actual values provides an estimate of the quality of the results.

We first present the results for the prediction of melt-pool length, depth, and width for the Eagar-Tsai-462 data set and the melt-pool depth for the Eagar-Tsai-100 data set. Table 4 summarizes the results of five runs of fivefold cross validation for the different surrogates, while Figs. 2 and 3 show the predicted vs. actual values using the leave-one-out method. We observe the following:

- The errors in the melt-pool depth and width tend to be similar and larger than the error in the length, which is the least important characteristic.
- A single regression tree is less accurate than an ensemble of 10 trees, which is expected. However, the cross validation error for an ensemble is still relatively high and the plots

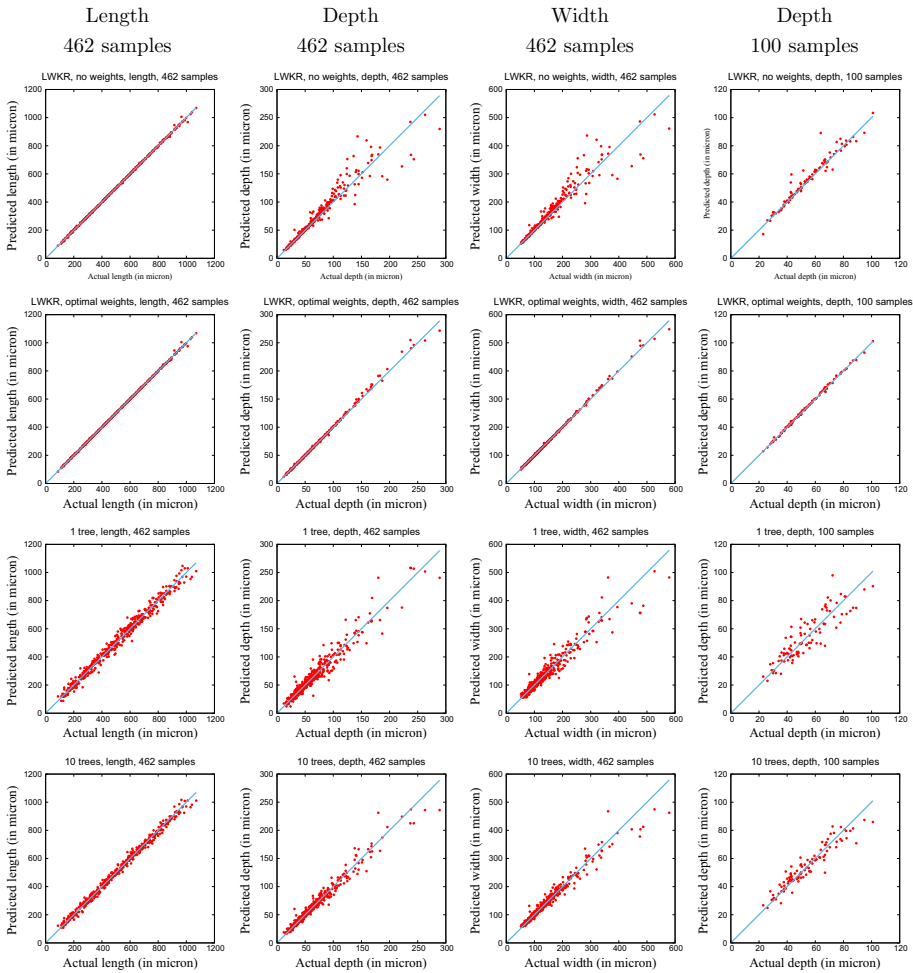


Fig. 2 Predicted versus actual values for the length, depth, and width for the Eagar–Tsai-462 data set and the depth using the Eagar–Tsai-100 data set for the different surrogate models using a leave-one-out approach. From top to bottom: locally weighted kernel regression with no feature weighting, locally weighted kernel regression with optimal feature weights, single regression tree, ensemble of 10 regression trees

of predicted vs. actual values show a good deal of scatter. As each node in the tree is required to have a minimum number of instances, the trees are very shallow and inaccurate, especially in the smaller Eagar–Tsai-100 data set.

- SVR has a much higher cross validation error than either LWKR with optimal weights, MARS, or GP. The predicted vs. actual value plots indicate that the error is mainly at the higher values, where the method under predicts both the depth and width. We suspect that the smaller number of samples at higher values results in a less-than-optimal choice of support vectors. The prediction is better for the Eagar–Tsai-100 data set that has fewer samples with large depth.
- As suggested by the ANOVA analysis, the cross validation results and the LOO prediction plots for the melt-pool depth and width are substantially improved if we use LWKR with optimal weights instead of weighting all inputs equally in calculating distances.

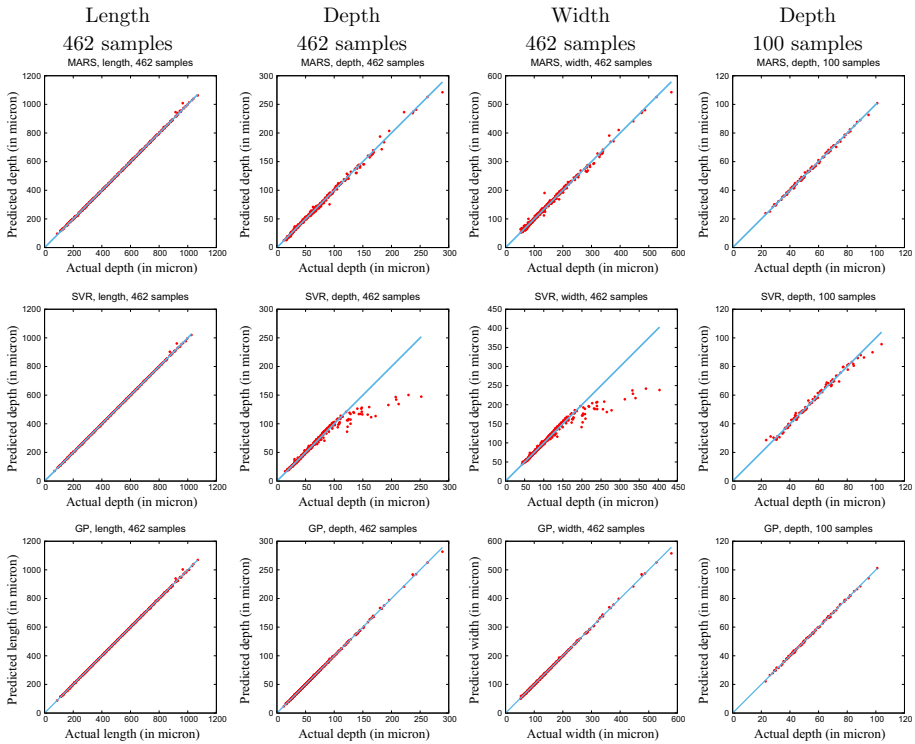


Fig. 3 Predicted versus actual values for the length, depth, and width for the Eagar–Tsai-462 data set and the depth using the Eagar–Tsai-100 data set for the different surrogate models using a leave-one-out approach. From top to bottom: *MARS* multivariate adaptive regression splines, *SVR* support vector regression, and *GP* Gaussian processes

- The accuracy of *MARS* and *LWKR* with optimal weights is comparable, while *GP* outperforms both. This is true for both data sets and all melt-pool characteristics.
- In surrogate models that perform well, there is a slightly more scatter at higher depth and width values in the Eagar–Tsai-462 data set. As mentioned earlier, there is a small region in design space where high laser power and low laser speed result in a rapid increase in the melt-pool depth and width. While the Eagar–Tsai-462 data set has some samples in this region, they are not enough to capture the variation, resulting in less accurate prediction. In contrast, this behavior is not seen in the Eagar–Tsai-100 data set as the smaller number of samples does not even capture this region.

For the remainder of the paper, we focus on the melt-pool depth as it is the most important of the three characteristics. Also, given their improved performance, we present results for *LWKR* only with optimal weights and for regression trees only with an ensemble of 10 trees.

4.3 Comparing surrogate models in practical applications

We have shown in Sect. 4.2 that several surrogate models give satisfactory prediction accuracy using the cross validation and the leave-one-out error metrics. We next compare how effective each surrogate is in its prediction of the melt-pool depth at new sample points especially as we reduce the size of the training data. These examples also illustrate the ways in which the

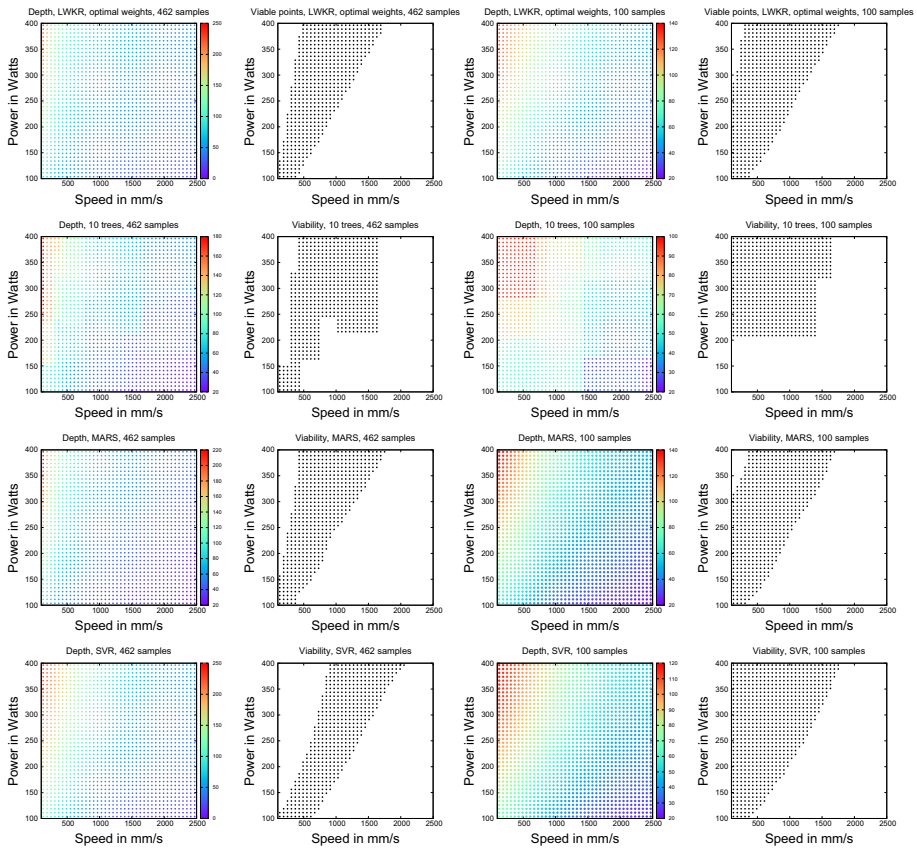


Fig. 4 Comparing the viability regions identified by different surrogates for the Eagar–Tsai simulation as the number of samples points is reduced. Columns 1 and 3 are the prediction of the melt-pool depth using Eagar–Tsai-462 and Eagar–Tsai-100, respectively, while columns 1 and 4 are the corresponding viability regions. The rows are from top to bottom—LWKR with optimal weights, an ensemble of ten regression trees, MARS, and SVR

surrogates can be used in a practical application and give us a way of evaluating surrogates on small data sets when metrics such as cross validation become unreliable [19].

4.3.1 Determination of viable region

In the first example, we use the surrogates to determine the viable region where the power and speed values result in sufficiently deep, but not too deep, melt pools. In our previous work [21], we found that for a powder layer thickness of 30 microns, a depth value from the simple Eagar–Tsai model could be considered viable if it was greater than or equal to 60 microns, but less than or equal to 120 microns.

Ideally, a good surrogate would predict similar viable regions with the smaller Eagar–Tsai-100 data set as with the larger Eagar–Tsai-462 data set. So, we generated viability plots by training the different surrogates on the two data sets and predicting the depth on a 40×40 grid of sample points in the power-speed space, keeping the values of the other two variables fixed at 52 microns for the beam size and 0.4 for the absorptivity. Figures 4 and 5 show the

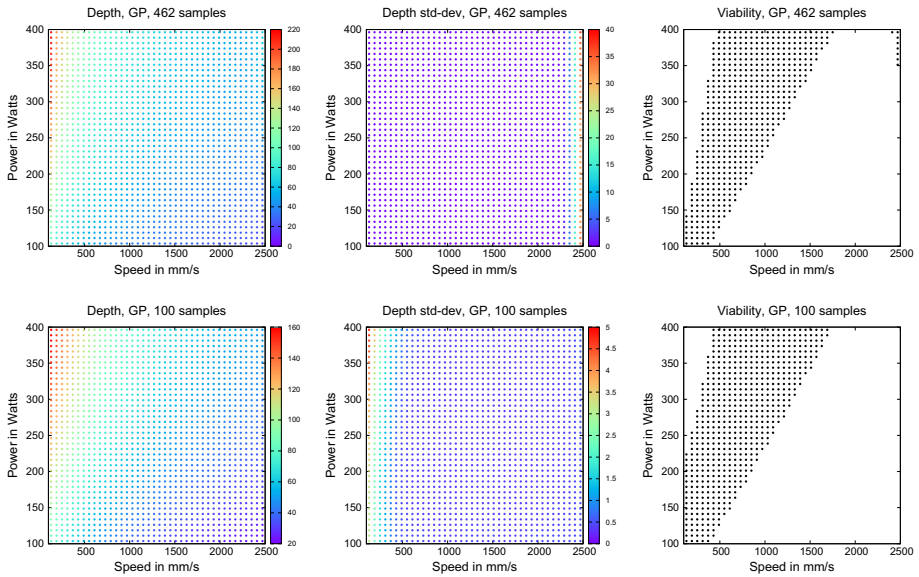


Fig. 5 Comparing the viability regions identified by and Gaussian process for the Eagar–Tsai simulation as the number of sample points is reduced. The columns are from left to right: the prediction of the melt-pool depth, the standard deviation in the prediction, and the viability region. Results for the top row are generated using the Eagar–Tsai-462 data set and for the bottom row, using the Eagar–Tsai-100 data set

predictions of melt-pool depth over this grid of 1600 sample points, along with the viability regions, for the two training data sets. For the GP surrogate, we also include the one standard deviation uncertainty in the prediction. We observe the following:

- The viable region for the ensemble of regression trees has the block structure typical of tree methods resulting from the axis-parallel cuts that divide the input space. The viable region is also very different when a smaller training set is used.
- Since SVR under predicts the depth at larger values, the viable region with the smaller training set completely fills the gap near the top left corner, where the melt-pools are very deep.
- For both LWKR with optimal weights and MARS, the viable region for the smaller training set is very similar to, but slightly larger than, the viable region for the larger training set.
- For the GP surrogate, the viable region for the smaller data set is closest to the viable region for the larger data set. Note that in the larger data set, there are a small number of viable points in the upper right corner at speed values of 2500 mm/s. These points are also associated with greater uncertainty in prediction as they lie outside the range of the training data, which has a maximum speed of 2250 mm/s. This availability of the associated uncertainty in the prediction is an added benefit of the GP surrogate.

4.3.2 Identifying parameters for constant depth

In additive manufacturing, to create a higher quality part, we would ideally like to keep the dimensions of the melt pool roughly constant as the laser traces out the part on each layer. Beuth and colleagues [7] have proposed the use of “process maps,” which are curves in

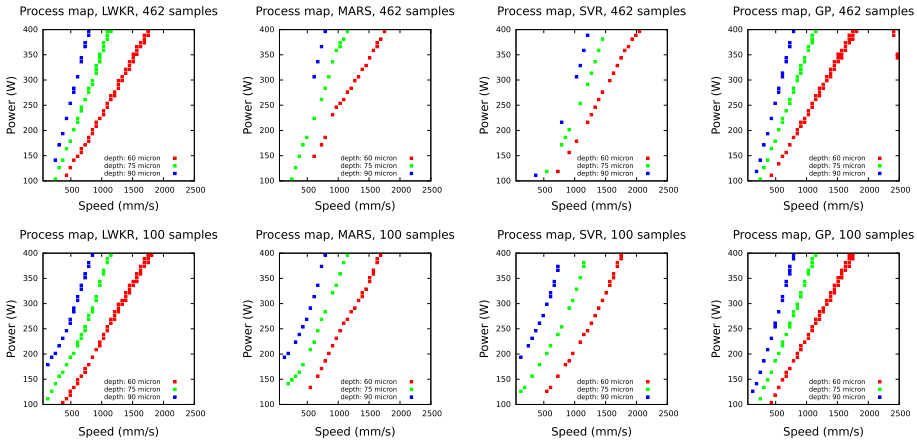


Fig. 6 Process maps indicating points at a constant depth in the power-speed space. The columns from left to right are LWKR with optimal weights, MARS, SVR, and GP. Results for the top row are generated using the Eagar–Tsai-462 data set and for the bottom row, the Eagar–Tsai-100 data set

power-speed space where melt-pool characteristics, such as depth, are constant. So, if the laser has to slow down to turn, it could change the power appropriately to keep the depth constant.

Code surrogates can be used to build these process maps. Figure 6 shows the curves with melt-pool depths of 60, 75, and 90 microns, predicted by the four surrogates using the Eagar–Tsai-462 and Eagar–Tsai-100 data sets. The points in these plots are within ± 1 micron of the depth for each curve. A good surrogate should give very similar curves with the smaller training data as with the larger training data. This is the case for LWKR with optimal weights and GP; though the latter has spurious points in the top right corner, it gives a better localization of the curves. In contrast, the location and completeness of the curves using MARS and SVR are very different when the training set is reduced in size.

4.3.3 Predictions for designing experiments

In earlier work [21], we showed that a Gaussian process model trained using the Verhaeghe-41 data set was able to predict, with reasonable accuracy, the actual depth obtained from an experiment where 14 single tracks were created on a plate. Unlike the Eagar–Tsai model, which is a simple approximation, the Verhaeghe model gives results that are much closer to the experiment. However, it is computationally very expensive, limiting the number of simulations we can run to fully understand the input design space. Code surrogates are an obvious alternative, but it is important to understand how well a surrogate would perform with such a small training set.

In Table 5, we present the predictions of the melt-pool depth for the 14 experimental tracks using the four surrogates. These results were obtained using the Verhaeghe-41 data set for training. We have also included the experimental value of the depth, mainly for comparison and to illustrate that more complex simulations can give results close to the experiment. The table lists the error in prediction of each of the surrogates relative to the depth from running the Verhaeghe simulation at the parameters for the 14 tracks. As a simple metric, we evaluated the sum of the absolute values of these errors for the four surrogates, which is 43.4, 47.9,

Table 5 Depth, in microns, from the single-track experiments, the Verhaeghe model, and the predictions using four code surrogates built with the Verhaeghe-41 data set

Track	Power	Speed	Expt	Verhaeghe	Surrogates			
					LWKR	MARS	SVR	GP
1	400	1800	105.0	105.0 (0.0)	105.6 (− 0.6)	101.5 (3.5)	105.0 (0.0)	105.0 (0.0)
2	400	1500	119.0	127.5 (− 8.5)	127.7 (0.2)	124.4 (3.1)	130.6 (− 3.1)	126.6 (0.9)
3	400	1200	182.0	163.5 (18.5)	157.8 (5.7)	158.0 (5.5)	160.5 (3.0)	155.5 (8.0)
4	300	1800	65.0	75.0 (− 10.0)	78.6 (− 3.6)	76.8 (− 1.8)	63.1 (11.9)	68.1 (6.9)
5	300	1500	94.0	90.0 (4.0)	90.6 (− 0.6)	91.1 (− 1.1)	87.5 (2.5)	89.4 (0.6)
6	300	1200	114.0	115.0 (− 1.0)	113.6 (1.4)	113.4 (1.6)	116.7 (− 1.7)	116.7 (− 1.7)
7	300	800	175.0	172.5 (2.5)	179.1 (− 6.6)	167.2 (5.3)	161.2 (11.3)	169.3 (3.2)
8	200	1500	57.0	55.0 (2.0)	52.3 (2.7)	57.8 (− 2.8)	41.7 (13.3)	57.9 (− 2.9)
9	200	1200	68.0	70.0 (− 2.0)	68.2 (1.8)	68.8 (1.2)	68.8 (1.2)	78.3 (− 8.3)
10	200	800	116.0	105.0 (11.0)	114.4 (− 9.4)	99.4 (5.6)	111.3 (− 6.3)	104.5 (0.5)
11	200	500	195.0	165.0 (30.0)	165.9 (− 0.9)	163.2 (1.8)	146.3 (18.7)	169.0 (− 4.0)
12	150	1200	30.0	47.5 (− 17.5)	45.0 (2.5)	46.4 (1.1)	45.3 (2.2)	59.5 (− 12.0)
13	150	800	67.0	72.5 (− 5.5)	81.6 (− 9.1)	65.5 (7.0)	86.1 (− 13.6)	69.5 (3.0)
14	150	500	120.0	112.5 (7.5)	115.3 (− 2.8)	110.5 (2.0)	120.1 (− 7.6)	113.1 (0.6)

Power values are in Watts and speed is in mm/s. For the Verhaeghe model, the value in parenthesis is the difference from the experiment, and for the four surrogates, it is the difference from the Verhaeghe model. The sum of the absolute error for LWKR, MARS, SVR, and GP is 47.9, 43.4, 96.4, and 52.6, respectively

52.6, and 96.4 microns for MARS, LWKR with optimal weights, GP, and SVR, respectively. This indicates that the MARS, LWKR, and GP surrogates tend to perform better than SVR.

4.4 Discussion of results

Our experiments provide some interesting insights into the use of regression algorithms with small data sets. Though we illustrate our work using the problem of surrogate models in the domain of additive manufacturing, our results extend to any domains and problems where generating the data is very expensive, leading to small data sets. By comparing algorithms in terms of how they would be used in a practical application, such as AM, we make our observations more applicable in the context of a real problem. The use of simulations to generate the data makes it easy to create data sets of different sizes for comparison. The variables to be predicted are reasonably well-behaved, without any discontinuities or large regions with rapidly varying values; this makes the results somewhat independent of the samples used in the simulations.

Of the five algorithms compared, the regression tree performs the worst. As the size of the data set decreases, the requirement that a leaf node has a minimal number of samples leads to a large variation in values at a leaf node and inaccurate predictions. SVR is not competitive on smaller data sets as it performs poorly in regions with large variation but few sample points to capture the variation. We suspect the support vectors do not adequately reflect the target values in such regions. The remaining three methods—LWKR with optimal weights, MARS, and GP—all perform well. Of these, GP provides the best performance, both in terms of prediction accuracy and use in practice. It also provides the uncertainty in prediction, which could be used to determine locations to add new sample

points when a data set is generated incrementally. Both LWKR and MARS are comparable in accuracy and practical use. The near-optimal weights obtained for the distance calculation in LWKR provides an indication of the relative importance of the different input dimensions.

Finally, we did not consider the computational time for each algorithm. As the data sets are small, the time required to build and apply the model is also small, and therefore less relevant in the comparison.

5 Related work

The work in prediction using small data sets has been driven mainly by two, somewhat related, factors: the high cost of collecting data in some problem domains and the need to draw conclusions from the data that has already been collected, while accounting for the fact that a small sample data set might not reflect the population as a whole. The early work in this topic focused on the sample sizes required to build good models with small data sets. In behavioral sciences, where simple linear regression models on predictor variables were used, the desire to avoid over-fitting led to several rules of thumb to determine appropriate sample sizes. Typical rules suggested a minimum of 10 to 15 observations per predictor variable or a minimum base sample size of 50 observations plus 8 additional observations per predictor [4]. With increasing availability of compute power, Monte Carlo simulations have been used to determine minimum sample sizes for estimating regression coefficients and for R^2 statistics in prediction [3]. The topic of selecting models for small sample sizes has been studied both theoretically for parametric regression [12] and for classification problems in high-dimensional simulated data sets in bioinformatics [18]. We found little work that compared how different methods performed on small data sets, both in terms of prediction accuracy, as well as practical use.

6 Conclusions

In this paper, we investigated the use of regression algorithms to build code surrogates for computer simulations. As the simulations are expensive, the training data set is small, in some cases consisting of just tens of samples. Despite this, we showed that we can build accurate surrogates by appropriately choosing both the regression algorithm and the samples in the input parameter space of the simulations. Our test problem, from additive manufacturing, was to predict characteristics of the melt pool formed when a laser beam, moving at a specific speed, melts a layer of powder into the substrate below. Our experiments indicated that ensembles of regression trees and support vector regression did not perform well, but locally weighted kernel regression and multivariate adaptive regression splines gave accurate results. Gaussian process models performed the best; they were not only accurate, but also provided a measure of confidence in the prediction results. Our work is broadly applicable to the task of building code surrogates for simulations in other domains such as climate and materials sciences, in building regression models with small data sets, and in time-constrained machine learning tasks such as hyper-parameter optimization. Finally, we observe that code surrogates, like any other interpolation techniques, work best with functions that are continuous and smoothly varying; discontinuities and sharp gradients in the function are unlikely to be accurately identified in the sampling.

Acknowledgements The results in this paper were generated using codes we developed for regression trees and LWKR, as well as public domain codes for MARS [33], SVR [11], and GP [17]. The Eagar–Tsai data were generated using a code developed by David Macknelly. We thank the anonymous reviewers for their feedback. This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

References

1. ACME (2016) Accelerated climate modeling for energy web page. <https://climatemodeling.science.energy.gov/projects/accelerated-climate-modeling-energy>
2. Atkeson C, Schaal SA, Moore AW (1997) Locally weighted learning. *AI Rev.* 11:75–133
3. Austin PC, Steyerberg EW (2015) The number of subjects per variable required in linear regression analyses. *J Clin Epidemiol* 68:627–636
4. Babyak MA (2004) What you see may not be what you get: a brief, non-technical introduction to overfitting in regression-type models. *Psychosom Med* 66:411–421
5. Bennett KP, Mangasarian OL (1992) Robust linear programming discrimination of two linearly inseparable sets. *Optim Methods Softw* 1(1):23–34
6. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. *J Mach Learn Res* 13:281–305
7. Beuth J et al (2013) Process mapping for qualification across multiple direct metal additive manufacturing processes. In: Bourell D (ed) International solid freeform fabrication symposium, an additive manufacturing conference. University of Texas at Austin, Austin, Texas, pp 655–665
8. Breiman L, Friedman JH, Olshen RA, Stone CJ (1984) Classification and regression trees. CRC Press, Boca Raton
9. Burl MC et al (2006) Automated knowledge discovery from simulators. In: Proceedings, Sixth SIAM international conference on data mining, pp 82–93
10. Carriera-Perpiñán MA (1996) A review of dimension reduction techniques. Tech. rep., Technical Report CS-96-09, Department of Computer Science, University of Sheffield, UK
11. Chang C-C, Lin C-J (2011) LIBSVM: A library for support vector machines. *ACM Trans Intell Syst Technol* 2:27:1–27:27. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>
12. Chapelle O, Vapnik V, Bengio Y (2002) Model selection for small sample regression. *Mach Learn* 48(1):9–23
13. Committee on Mathematical Foundations of Verification, Validation, and Uncertainty Quantification; Board on Mathematical Sciences and Their Applications, Division on Engineering and Physical Sciences, National Research Council (2012) Assessing the reliability of complex models: mathematical and statistical foundations of verification, validation, and uncertainty quantification. The National Academies Press, Washington
14. Eagar T, Tsai N (1983) Temperature-fields produced by traveling distributed heat-sources. *Weld J* 62:S346–S355
15. Fang K-T, Li R, Sudjianto A (2005) Design and modeling for computer experiments. Chapman and Hall/CRC Press, Boca Raton
16. Friedman JH (1991) Multivariate adaptive regression splines. *Ann Stat* 19, 1(03):1–67
17. GPy (2012) GPy: A Gaussian process framework in python. <http://github.com/SheffieldML/GPy>
18. Guo Y, Graber A, McBurney RN, Balasubramanian R (2010) Sample size and statistical power considerations in high-dimensionality data settings: a comparative study of classification algorithms. *BMC Bioinf* 11:447
19. Isaksson A, Wallman M, Goransson H, Gustafsson M (2008) Cross-validation and bootstrapping are unreliable in small sample classification. *Pattern Recogn Lett* 29:1960–1965
20. Kamath C (2009) Scientific data mining: a practical perspective. Society for Industrial and Applied Mathematics (SIAM), Philadelphia
21. Kamath C (2016) Data mining and statistical inference in selective laser melting. *Int J Adv Manuf Technol* 86:1659–1677
22. Kamath C, Cantú-Paz E (2001) Creating ensembles of decision trees through sampling. In: Proceedings of the 33-rd symposium on the interface: computing science and statistics
23. Kamath C, El-dasher B, Gallegos GF, King WE, Sisto A (2014) Density of additively-manufactured, 316L SS parts using laser powder-bed fusion at powers up to 400 W. *Int J Adv Manuf Technol* 74:65–78
24. Kleijnen JPC (2008) Design and analysis of simulation experiments. Springer, New York

25. Mitchell DP (1991) Spectrally optimal sampling for distribution ray tracing. *Comput Graph* 25(4):157–164
26. Oehlert GW (2000) A first course in design and analysis of experiments. W. H. Freeman. <http://users.stat.umn.edu/~gary/Book.html>
27. Owen AB (2003) Quasi-Monte Carlo sampling. Course notes from Siggraph course. <http://www-stat.stanford.edu/~owen/reports/>
28. Owen AB (1998) Latin supercube sampling for very high-dimensional simulations. *ACM Trans Model Comput Simul* 8(1):71–102
29. Qian Y et al (2016) Uncertainty quantification in climate modeling and projection. *Bull Am Meteorol Soc* 97(5):821–824
30. Rasmussen CE, Williams CKI (2006) Gaussian processes for machine learning. MIT Press, Cambridge
31. Rokach L (2010) Pattern classification using ensemble methods. World Scientific Publishing, Singapore
32. Rokach L, Maimon O (2014) Data mining with decision trees: theory and applications. World Scientific Publishing, Singapore
33. Rudy J (2013) Py-earth. <https://contrib.scikit-learn.org/py-earth/>
34. Schölkopf B, Smola AJ, Williamson RC, Bartlett PL (2000) New support vector algorithms. *Neural Comput* 12(5):1207–1245
35. Shiflet AB, Shiflet GW (2006) Introduction to computational science: modeling and simulation for the sciences. Princeton University Press, Princeton
36. Vapnik VN (1995) The nature of statistical learning theory. Springer, New York
37. Verhaeghe F, Craeghs T, Heulens J, Pandalaers L (2009) A pragmatic model for selective laser melting with evaporation. *Acta Mater* 57:6006–6012
38. Yadroitsev I, Gusarov A, Yadroitsava I, Smurov I (2010) Single track formation in selective laser melting of metal powders. *J Mater Process Technol* 210:1624–1631



Chandrika Kamath is a researcher at Lawrence Livermore National Laboratory, where she is involved in the analysis of data from scientific simulations, observations, and experiments. Her research interests include signal and image processing, machine learning, pattern recognition, and statistics, as well as the application of these techniques to solve practical problems. Chandrika holds six patents in data mining, received the 2006 R&D 100 award, and is a founding editor-in-chief of the *Statistical Analysis and Data Mining* journal. She is actively involved in the SIAM International Conference on Data Mining.



Ya Ju Fan received her M.Sc. degree in Decision Science from the University of Wisconsin-Madison and her PhD in Industrial and Systems Engineering from Rutgers University. She is a computational scientist at the Center for Applied Scientific Computing in Lawrence Livermore National Laboratory. Her research interest is to support decision making by developing and applying optimization algorithms and statistical methods for the analysis of data. She has experience with analyzing a wide variety of datasets, including but not limited to medical data, gene expression profiles, images and multivariate time series.