LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-729017

Regression with Small Data Sets: A Case Study using Code Surrogates in Additive Manufacturing

C. Kamath, Y. J. Fan

April 12, 2017

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Regression with Small Data Sets: A Case Study using Code Surrogates in Additive Manufacturing

Chandrika Kamath and Ya Ju Fan

Lawrence Livermore National Laboratory
7000 East Avenue
Livermore, CA 94551, USA
kamath2,fan4@llnl.gov

April 6, 2017

Abstract

There has been an increasing interest in recent years in the mining of massive data sets whose sizes are measured in terabytes. While it is easy to collect such large data sets in some application domains, there are others where collecting even a single data point can be very expensive, so the resulting data sets have only tens or hundreds of samples. For example, when complex computer simulations are used to understand a scientific phenomenon, we want to run the simulation for many different values of the input parameters and analyze the resulting output. The data set relating the simulation inputs and outputs is typically quite small, especially when each run of the simulation is expensive. However, regression techniques can still be used on such data sets to build an inexpensive "surrogate" that could provide an approximate output for a given set of inputs. A good surrogate can be very useful in sensitivity analysis, uncertainty analysis, and in designing experiments. In this paper, we compare different regression techniques to determine how well they predict melt-pool characteristics in the problem domain of additive manufacturing. Our analysis indicates that some of the commonly used regression methods do perform quite well even on 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 application 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. However, we can still apply data mining techniques to these data to gain insight. An example of such a situation is the building of *code surrogates* as a faster alternative to computer simulations.

In many application domains, such as additive manufacturing [20] and climate [1, 29], computer simulations, sometimes referred to as "codes", are increasingly being used to complement experiments and observations. These simulations are essentially computational models that, given the inputs of a physical phenomenon (referred to as the parameters of the code), generate the corresponding outputs [36]. The insight provided by simulations benefits scientists in many ways; for example, it enables them to select parameters for use in an experiment, to determine which input parameters are more important than others, 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, with detailed physics that gives more accurate results. However, such complex simulations can 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 can be 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 (also referred to as a *code surrogate* or a *metamodel*) 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, when the surfaces relating the outputs to the inputs become more complex, alternatives such as neural networks are an option; such techniques arethe focus of the work presented in this paper.

Code surrogates can be used in many ways as they are essentially an inexpensive alternative to a simulation. When the simulations are simple, but approximate, surrogates can be used to identify a viable region in design space, where an experiment is more likely to work. The points in this viable region could then be considered as inputs to more complex and longer-running simulations [20], 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 [8]. Finally, surrogates play an important role in uncertainty analyses and sensitivity studies [12].

Code surrogates can be relatively inexpensive to build and use. However, to be effective, their predictions must be accurate enough for the task at hand. The predictive quality of a surrogate depends on several factors, including the number and placement of sample points in the simulation input parameter space, the complexity of the function relating the outputs to the inputs, and the algorithm used to create the surrogate model. For computationally-expensive simulations that result in small data sets, an obvious question to ask is the following - are some code surrogates better predictors then others when the number of sample points is small? In this paper, we attempt to address this question using a problem from additive manufacturing (AM) as a testbed.

This paper is organized as follows: First, in Section 2, we describe the application domain of additive manufacturing. We are interested in determining the characteristics of the melt-pool formed in laser powder-bed fusion, which is one of many AM processes currently in use. We describe two different simulations used in our work (Section 2.1) and two different sampling schemes used to generate sample points at which to run the simulations (Section 2.2). In Section 3, we describe the resulting data sets, followed in Section 4 by the algorithms used to build the code surrogates. Section 5 compares the performance of these algorithms using a variety of metrics. Related work is discussed in Section 6 and we conclude with a summary in 7.

2 The application domain of additive manufacturing

Additive manufacturing (AM), also referred to as 3-D printing, is a process for fabricating parts, layer-by-layer. In laser powder-bed fusion, which is one of many AM processes currently in use, a laser beam is used to fuse fine powders together. The process starts by slicing a three-dimensional model of the part into two-dimensional layers of a fixed thickness, typically a few tens of microns. To build the part, a thin layer of powder is then spread on a base plate and the first layer is created by selectively melting the powder in the locations indicated in the first slice of the part. The process is repeated for each successive layer and the part is built, layer by layer, with the power and speed of the laser selected so that the energy density is sufficient to melt the powder and the layers below it, thus integrating the new layer into the rest of the part.

AM enables the production of complex parts not possible with traditional manufacturing processes. But, it is a challenge to understand how the many parameters — nearly 130 by some estimates — that control an AM process affect the properties and quality of an additively-manufactured part. These parameters include the properties of the material, such as thermal conductivity and melting point; the properties of the powder

bed, such as the particle sizes and the layer thickness; as well as the laser parameters, such as the power of the laser, its speed, and the path taken by the laser in each layer. It is very expensive to use experiments to fully explore the high-dimensional design space of AM, and while computer simulations are a viable alternative, they too can be expensive if they incorporate all the physics involved in AM.

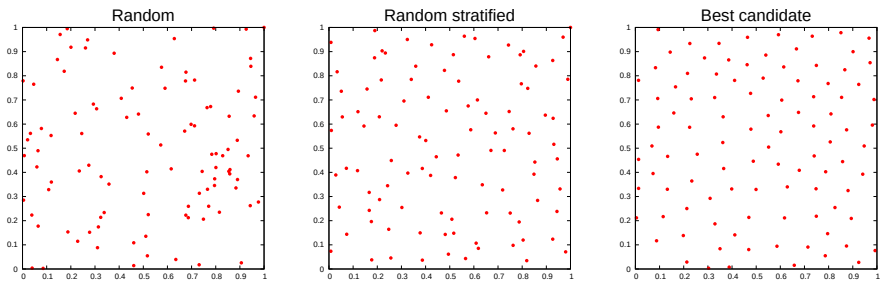
In our previous work [22], we showed that we could combine simple simulations and experiments to identify process parameters that resulted in high-density AM parts. We then extended this work by using the simple simulations to identify the design space of viable sample points, and running a more expensive, but more accurate, simulation at a select subset of these viable points [20]. Our focus in these simulations was the characteristics of the melt pool that is formed when the laser melts the powder. We wanted to select input parameters that would result in melt pools that were deep enough to melt through the powder into the substrate below, but not so deep that we waste energy. In comparing the simulation results with experiments, we found that the simple simulations were not sufficiently accurate in predicting the melt pool characteristics, while the more expensive ones are accurate, but it was time consuming to run too many of them. This naturally prompted us to consider code surrogates for these simulations. We next briefly describe the simulations we used and the process for generating the data used in this paper.

2.1 Computer simulations to determine melt-pool characteristics

We considered two physical models for generating the melt-pool characteristics:

- **Eagar-Tsai model:** This very simple model [13] considers a Gaussian laser beam moving on a flat plate. The resulting temperature distribution on a three-dimensional spatial grid representing the plate is used to compute the melt-pool width, depth, and length 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 $\approx$1 minute to run on a laptop. This means that we can sample the input parameter space of the Eagar-Tsai model rather densely, making it possible to use the model to identify the viable region in the input space.
- **Verhaeghe model:** A more expensive physical model is the Verhaeghe model, which considers various physical phenomena involved in the laser melting of powder on a substrate [38]. This model is more computationally expensive than the Eagar-Tsai model, with the three-dimensional simulations requiring about 1-3 hours of computational time (depending on the laser speed) on a small eight-processor cluster. While this is a moderate requirement in terms of computational resources, the Verhaeghe 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. While the addition of the new physics in the Verhaeghe model makes it more realistic compared with the Eagar-Tsai model, and the output predicts the melt-pool characteristics more accurately, the new parameters lead to a higher-dimensional design space, which requires more sample points for exploration.

In addition to simulations, it is also possible to determine the melt-pool characteristics using single-track experiments [39], where a single layer of powder is spread on a plate, and the laser is used to create single tracks at a specific power and speed values. The plate is then cut perpendicular to the tracks, etched, and polished to reveal the track cross section from which the depth and width are obtained. The cost and time for creating the tracks is small, usually less than a minute, but the pre- and post-processing of the plate after the creation of the track is quite expensive. Therefore the parameters for each track have to be selected carefully, which is a challenge if a material has not been additively manufactured in the past, or the AM machine has characteristics different from current machines. As a more cost-effective alternative, we have used computer simulations to guide the choice of process parameters for the single track experiments [20, 22]. Later in Section 5.3, we will compare the prediction of a surrogate for the Verhaeghe model with single-track experimental data.



(a) (b) (c)

Figure 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.

2.2 Sampling the input space of simulations

The two physical models described in Section 2.1 have a number of input parameters. For each set of values for the inputs, the simulations produce an output consisting of values of variables, such as the temperature, at each grid point in three-dimensional space; from this we can extract the melt-pool characteristics. The input to a simulation run can be considered as a sample point in parameter space, where each parameter takes on a specific value. When we use computer simulations to understand a physical process by changing various parameters in the simulation and observing the effect of these changes on the outputs, we have to decide which parameters to change and what values to use for the parameters. This is especially important for expensive simulations as the number of sample points required to adequately sample a parameter space is exponential in the number of parameters, an issue referred to as the curse of dimensionality [9].

When little is known about a simulation, the initial set of sample points are generated randomly in the design space and additional points are added as required. However, as seen in Figure 1(a) 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 (Figure 1(b)). 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 second sampling scheme considered is Mitchell's best-candidate sampling [25] that was originally proposed in computer graphics. 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 tends to place samples as far apart from each other as possible. It also allows us to specify an arbitrary number of samples, incrementally add new samples to an existing set, and sub-sample from an existing set.

3 Description of the data

To conduct our study into the performance of different surrogate models when the number of sample points is small, we use three data sets generated from the two physical models from Section 2.1 and the two sampling schemes from Section 2.2. These data sets were also used in our previous work in identifying process parameters for additive manufacturing [20].

- **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 [22]. 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 Eager-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, while an improvement on the simpler Eagar-Tsai 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.

4 Description of the code surrogates

We consider several different code surrogates in our study to determine if some surrogates perform better than others on small sample data sets. We next briefly describe the algorithms used to build these surrogates. They were chosen based on their effectiveness in building a good predictive model, their simplicity and the resulting interpretability, as well as their successful use in various application domains. One of these algorithms (the nearest-neighbor method), is a "lazy" method, deferring the processing of the data until the prediction needs to be made at a sample point. The other algorithms involve the more traditional three step process, where the model is first built in a training phase, evaluated in a testing phase with data not used in building the model, and then applied only if the accuracy is found to be sufficient.

4.1 Nearest neighbor methods

The simplest among regression methods, this class of methods predict the value at a point in input space as a function of the values of its nearest neighbors in the input space. The k-nearest-neigbor method predicts the output as the mean of the values of the k nearest neighbors, where k is chosen appropriately. More complex models replace the mean by distance-weighted averages, where near neighbors contribute more than ones that are farther away.

In our work, we use locally-weighted regression [2], where we build a locally-linear model, whose coefficients are obtained using a least squares fit. The distance used is a kernel distance, with a Gaussian kernel. Since our input variables have units such that some inputs (for example, laser power) have values much larger than others (for example, beam size), we scale each input to lie between 0.0 and 1.0 before we apply

the method. This scaling also allows us to select an appropriate bandwidth of the Gaussian, which we set to 0.1. By using a kernel distance, we avoid having to select a value for k, the number of neighbors, as far away neighbors will have a very large kernel distance, resulting in a very small contribution to the result. We refer to this method as locally-weighted kernel regression (LWKR).

We initially used a Euclidean distance metric to calculate the distances between sample points. However, an initial exploratory analysis of the training data using Analysis of Variance (Section 5.1) indicated that we could improve the performance of the LWKR method by using a weighted Euclidean distance.

4.2 Regression trees

A regression tree algorithm [7] uses the data, consisting of simulation inputs and the corresponding outputs, to create a structure that is either a leaf, indicating a continuous value, or a decision node that specifies some test to be carried out on an input, with a branch and sub-tree for each possible outcome of the test. For continuous inputs, there are two branches, depending on whether the condition being tested is satisfied or not. The decision at each node of the tree is made to reveal the structure in the data. To predict the output for a new sample point, its inputs are used to traverse the tree, until a leaf node is reached. The predicted value is the mean of the output values of the training data that end up at that leaf node.

The test at each node of the tree is determined by examining each feature and finding the split that optimizes an impurity measure. We use the mean-squared error, MSE, which for a split A on a certain input is defined as

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

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. In tree-based algorithms, if the data are split too finely, it can lead to over fitting. To avoid this, we stop the splitting 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. As a split can be viewed a decision surface perpendicular to one of the axes (the input variable selected for the split), regression models often have a "block" structure (see, for example, Figure 4.)

A common approach to improving the accuracy of regression algorithms is to use an ensemble, where many models, built from the same training data using randomization, are created [19, 31, 32]. The final prediction is the mean of the prediction from each of the models. In our work, we consider 10 trees in the ensemble, with randomization introduced through sampling. Instead of using all the sample points at a node of the tree to make a split, we use a random subset of the samples [21], thus making each tree in the ensemble different from the others.

4.3 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. The combination involves identifying additive contributions and multivariable interactions based on a divide and conquer strategy.

Let $X = \{X_1, X_2, \ldots, X_p\}$ be a matrix of p input variables and y be the target dependent responses. MARS makes no assumptions about the underlying functional relationships between y and X. In the training

phase, the data are partitioned into separate, piecewise, linear segments of differing slopes. MARS builds piecewise curves, called basis functions, using these linear segments (called linear splines) that are connected smoothly together. The MARS model is a linear combination of the basis functions and their interactions. The points between the curves, called knots, indicate the end of one segment of data and the beginning of another. These knots are incorporated into the basis functions using hinge functions, defined as

$$\max(0, x-t) = \begin{cases} x-t, & \text{if } x \geq t \\ 0, & \text{otherwise.} \end{cases}$$

The knot is represented at value t for a variable $x \in \{X_1, X_2, \ldots, X_p\}$. By connecting the knots, the hinge functions create piecewise linear functions. The MARS model $f(x)$ is in the form of a weighted sum of basis functions $\beta_j(x)$

$$f(x) = \sum_{j=1}^{M} c_j \beta_j(x)$$

with each c_j being a constant coefficient. Each basis function can be a constant, a hinge function, or a product of two or more hinge functions, depending on the interactions of multiple variables. The hinge functions allow the MARS model to capture the nonlinearity in the output variable.

The MARS algorithm requires two user-defined parameters — the maximum number of basis functions and the maximum interaction level — and is composed of two phases. The forward phase locates candidate knots at random positions within the range of each input variable to define a pair of basis functions. The algorithm then searches over all possible univariate candidate knots and across all interactions among the variables, iteratively adding a knot and its corresponding pair of basis functions to reach the maximum reduction in residual sum-of-squares (RSS) error in the prediction. The process of adapting the basis functions continues until the maximum number of functions is reached. At the end of the forward phase, the model is usually very complicated and overfits the data. The backward phase is to delete the redundant basis functions that made the least contributions at each step until the algorithm finds the best sub-model. The best sub-model is determined using the generalized cross validation (GCV) criterion that trades off goodness-of-fit against model complexity:

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

where M is the number of basis functions; $\frac{M-1}{2}$ is the number of hinge function knots; N is the number of training samples; and δ is a penalty that is about 2 or 3. The numerator of GCV is the mean squared error, which 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.

4.4 Support vector regression

The support vector regression (SVR) algorithm starts with a training set of N points, denoted as $\{(x_1, y_1), \ldots, (x_N, y_N)\} \in \Re^p \times \Re$, where $x_i \in \Re^p$ is a p-dimensional feature vector and $y_i \in \Re^1$ is the corresponding output. In the training phase, the algorithm constructs a decision function

$$\begin{aligned} f(x) &= \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k(x_i, x) + b \\ &= \langle w, \phi(x)\rangle + b \end{aligned}$$

where x and $w \in \Re^p$, $b \in \Re$, $k(x_i, x) := \langle \phi(x_i), \phi(x)\rangle$ is a kernel function, representing a dot product of x_i and x in some feature space defined by ϕ, and $w = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)\phi(x_i)$. SVR optimizes α_i and α_i^*, which provides the conditions for computing b. Given a new feature vector x, we can then use $f(x)$ as an estimation of the corresponding output.

Support vectors are a small subset of all training samples that represent a decision boundary that can be used in classification or regression [34]. Given just the support vectors, we can easily make estimations based on the decision boundaries without having to keep all training samples. To extend this sparseness property to the case of support vector regression, the ε-SVR method [37] tries to find a decision function $f(x)$ that has at most ε deviation from all of the actual target output values, y_i, in the training data. Following statistical learning theory, SVR tries to obtain a small risk by minimizing both model complexity ($\|w\|^2$) and training error ($\max\{0, |y - f(x)| - \varepsilon\}$), using a constant C as the trade-off. In our work, we use the ν-Support vector regression (ν-SVR) method [10, 35] that formulates the optimization problem as:

$$\begin{aligned} \min_{w,b,\xi,\xi^*,\varepsilon} \quad & \frac{1}{2} w^\mathsf{T} w + C \cdot \left(\nu\varepsilon + \frac{1}{N}\sum_{i=1}^{N}(\xi_i + \xi_i^*)\right) \\ \text{subject to} \quad & w^\mathsf{T}\phi(x_i) + b - y_i \leq \varepsilon + \xi_i, \\ & y_i - w^\mathsf{T}\phi(x_i) - b \leq \varepsilon + \xi_i^*, \\ & \xi_i, \xi_i^* \geq 0, i = 1, \ldots, N, \quad \varepsilon \geq 0. \end{aligned}$$

where the parameter $\nu \in (0, 1]$ is used to control the number of support vectors so that ε becomes a decision variable that is to be optimized. Since the existence of the decision function f that approximates all pairs (x_i, y_i) with ε precision is not always possible, SVR uses the idea of a soft margin loss function [5] that includes the slack variables ξ_i and ξ_i^*, for $i = 1, \ldots, N$, to cope with otherwise infeasible constraints. The optimization model is solved using its dual problem by introducing two dual variables, α_i and α_i^*, for $i = 1, \ldots, N$ [10].

4.5 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.

We describe GP [14] using a one-dimensional problem with a training set of N samples, $\{x_1, \ldots, x_N\}$ and the associated output values $\mathbf{y} = \{y_1, \ldots, y_N\}$. For ease of exposition, we use the same notation as in the previous sections to describe GP for a one-dimensional problem. Two samples x_i and x_j in the training data are related to each other through the covariance function $k(x_i, x_j)$. We use the squared-exponential function:

$$k(x_i, x_j) = \sigma_f^2 \exp\left(\frac{-(x_i - x_j)^2}{2l^2}\right)$$

where σ_f^2 is the maximum allowable covariance and l is a length parameter that determines the extent of influence of each point. σ_f^2 should be set to a large value for functions which cover a broad range of values. If points x_i and x_j are close to each other, their output values are highly correlated, but if they are far away, then the value at one point does not influence the value at the other point. Therefore, the parameter l controls the smoothness of the interpolation.

Suppose we want to use the training data to predict the output at a new sample point x_*. Since the data can be represented as a sample from a multivariate Gaussian distribution, we have

$$\begin{bmatrix} \mathbf{y} \\ y_* \end{bmatrix} = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix}\right),$$

where $\mathbf{y}$ is the output variable corresponding to the N training data, y_* is the prediction of the output at

point x_*, and the submatrices are defined as follows:

$$K = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_N) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_N) \\ \vdots & \vdots & \vdots & \vdots \\ k(x_N, x_1) & k(x_N, x_2) & \dots & k(x_N, x_N) \end{bmatrix},$$

$$K_* = \begin{bmatrix} k(x_*, x_1) & k(x_*, x_2) & \dots & k(x_*, x_N) \end{bmatrix},$$

and

$$K_{**} = k(x_*, x_*).$$

The probability of y_*, that is, the output at the new sample point, is then given by

$$\overline{y}_* = K_* K^{-1} \mathbf{y}$$

and the uncertainty in the estimate is given by the variance

$$\text{var}(y_*) = K_{**} - K_* K^{-1} K_*^T.$$

The parameters l and σ_f of the Gaussian process can be 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.

5 Experimental results

We conduct several experiments with the three data sets described in Section 3 to evaluate the different surrogates and understand their performance for a small training set. Specifically, we:

- perform an exploratory analysis of the data using ANOVA as a prelude to understanding the data (Section 5.1),
- evaluate the accuracy of the different surrogate models on Eagar-Tsai-462 and Eagar-Tsai-100 data sets using two different metrics (Section 5.2), and
- describe how surrogate models built using a small training set perform in prediction (Section 5.3).

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 in watts, the laser speed in m/s, the beam size, which is the standard deviation of the Gaussian beam in micron, and the absorptivity of the material in the form of a fraction, indicating how much of the laser energy is absorbed by the material. 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.

Before we present our results comparing the different surrogate models, we observe that any such comparison will also reflect the underlying function we are trying to model, as well as the location of the sample points at which we ran the simulations. In our problem in additive manufacturing, the depth of the melt-pool is a relatively well behaved function that varies linearly or mildly-nonlinearly over a large range of the input parameters [20]. So, the comparison of the surrogates will give us insight on how they perform on such functions when the number of sample points is small.

We also observe that the melt-pool depth increases rapidly in a small region of the design input space where the laser power is high and the laser speed is low. This region is avoided in AM for several reasons.

The high energy input leads to melt-pools that are far deeper than necessary, thus wasting energy. The process also becomes unstable, leading to a large variation in depth for a fixed laser power and speed. The high energy also causes vaporization, resulting in porosity and low quality parts [23]. Since this non-linear 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 is used to model a rapidly-varying function.

5.1 Exploratory analysis using ANOVA

Before presenting the results for the different surrogates described in Section 4, we first discuss the results obtained from the application of the Analysis of Variance (ANOVA) method [26] to data derived from the Eagar-Tsai-462 data set. ANOVA can be considered as the use of very simple models based on mean values, and the deviation from mean values, to explain data sets. It is related to linear regression, though it cannot be considered as a surrogate in the strictest sense. It is mainly used in the context of design of physical experiments and therefore, has been applied in additive manufacturing to select process parameters. Our main reason for including ANOVA in this paper is that the insight it provides is very useful 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. We then divided this range into four equal segments and took the mid-point 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 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.

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

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

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:

- 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 [20]. These conclusions suggest that, to build good surrogate models, methods that are based on distances in the input space, such as the nearest-neighbor methods, would benefit by weighting the inputs in the calculation of the distances. To address this, we modified our implementation of the LWKR

Input Parameters	Output Variables		
	depth	length	width
Speed	$\mathbf{4.11 \times 10^2}$	4.50×10^{-2}	$\mathbf{3.88 \times 10^2}$
Power	$\mathbf{1.12 \times 10^2}$	$\mathbf{2.06 \times 10^3}$	$\mathbf{1.15 \times 10^2}$
Beam size	6.96×10^{-4}	1.26×10^{-5}	3.52×10^{-3}
Absorptivity	7.35	$\mathbf{2.59 \times 10^1}$	6.70

Input Parameters	Output Variables		
	depth	length	width
speed & power	$\mathbf{9.73 \times 10^1}$	2.13×10^{-2}	$\mathbf{1.49 \times 10^2}$
speed & beam size	2.20×10^{-2}	2.20×10^{-3}	5.08×10^{-3}
speed & absorptivity	9.73×10^{-1}	2.49×10^{-3}	5.83×10^{-1}
power & beam size	1.15×10^{-2}	9.96×10^{-5}	5.47×10^{-3}
power & absorptivity	1.37×10^{-1}	$\mathbf{5.90 \times 10^3}$	3.38×10^{-2}
beam size & absorptivity	3.26×10^{-3}	1.27×10^{-5}	9.67×10^{-4}

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

method to used a weighted Euclidean distance. The near-optimal feature weights were obtained using a simple approach where we generated random samples in the weight space with values in the range [0.0, 5.0], and selected the point that resulted in the minimum sum of absolute leave-one-out error on the training set. Table 3 lists the weights used for the LWKR model in this paper. Note that these weights, especially for the larger data sets, reflect the observations we made from the ANOVA analysis.

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 3: The near-optimal feature weights for the LWKR surrogate model.

5.2 Evaluating the accuracy of surrogate models

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

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

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 prediction using a leave-one-out (LOO) approach, 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. For a data set with N points, this is essentially N-fold cross validation. 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 five-fold cross validation for the different surrogates, while Figures 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 tends to be less accurate than an ensemble of 10 trees, which is to be expected. However, the cross validation error for an ensemble is still relatively high and the plots of predicted vs. actual values show a good deal of scatter. This is because the condition that a node in the tree is split only if it has a minimum number of instances, results in a very shallow and inaccurate tree, 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 underpredicts 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.
- 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 values of the depth and width in the plots of predicted vs. actual values for 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.

Surrogate	Relative MSE for 5 runs of 5-fold 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%

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

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.

5.3 Comparing surrogate models in practical applications

We have shown in Section 5.2 that several surrogate models give results with prediction accuracy that is satisfactory when evaluated using the two metrics of 5 runs of 5-fold cross-validation and the leave-one-out error. 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 surrogates can be used in practice in additive manufacturing.

5.3.1 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 [20], we found that if we used a powder layer thickness of 30 microns in building a part, then 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 surrogate that provides good predictions with a smaller number of sample points would give 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 using the two data sets as training data for the different surrogate models, 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 predictions of melt-pool depth over this grid of 1600 sample points, along with the viability regions, using the Eagar-Tsai-462 and the Eagar-Tsai-100 data sets. For the GP surrogate, we also include the one standard deviation uncertainty in the prediction. We make the following observations:

- The viable region for the ensemble of regression trees shows the block structure that is observed in tree methods due to the axis-parallel cuts that are made to divide the input space. The results also show that the viable region is very different when a smaller training set is used.
- For SVR, the viable region with the smaller training set completely fills the gap near the top left corner, where the melt-pools are deep and therefore not considered viable for use in AM. This is because SVR underpredicts the depth at larger values.
- 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 2500mm/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 2250mm/s. This availability of the associated uncertainty in the prediction is an added benefit of the GP surrogate.

5.3.2 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. This can be challenging as the local environment around the laser beam is constantly changing. In tracing out a layer, the laser could go over fresh powder, over previously melted-then-solidified powder, or adjacent to a just melted region of powder. The laser speed also changes when it slows down to take a turn. An approach proposed by Beuth and colleagues [6] was to identify curves, called "process maps", in power-speed space where characteristics such as melt-pool depth were constant. So, if the laser had to slow down to turn, it could change the power appropriately to keep the depth constant.

A possible use of code surrogates is to build these process maps for constant depth. Figure 6 shows the curves with melt-pool depths of 60, 75, and 90 microns, as predicted by the four surrogates built 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 would give very similar curves with the smaller training data set as with the larger training data set. We see that 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 as the training set size is reduced. In contrast, the location and completeness of the curves using MARS and SVR is very different between the two training sets.

5.3.3 Predictions for designing experiments

In our earlier work [20], 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 physical model, which is a simple approximation, the Verhaeghe physical model gives results that are much closer to the experiment. However, it is computationally very expensive, and therefore it is not possible to run a large number of Verhaeghe simulations 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 four surrogates for the melt-pool depth for the 14 experimental tracks. 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, 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.

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)

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. 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.

5.4 Discussion of results

Our comparison of regression algorithms for use as code surrogates in additive manufacturing provides some interesting insights into the use of these algorithms with small data sets. We start with some observations on the data sets used in this paper. Our application domain is one where we can evaluate the algorithms not just in terms of their prediction accuracy, but also in terms of how they would be used in practice, making our observations more applicable in the context of a real problem. The use of simulations to generate the data enables us to create data sets of different sizes for comparison. The melt-pool characteristics 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 sample points used as inputs to the simulations. We do observe however, that when we use a small number of sample points, despite our efforts to spread the samples uniformly, but randomly, we can miss some small regions with rapidly changing values. This is an inherent problem in sampling, but as all our algorithms are compared on the same data set, it should not be an issue in interpreting the results in this paper.

Of the five algorithms compared, the regression tree performs the worst with smaller data sets. This is mainly due to two reasons. The axis-parallel cuts of the input space results in structured splits of the space, making the predictions discontinuous at the splits. Further, as the size of the data set decreases, the requirement that each leaf node of the tree have a reasonable number of samples leads to leaf nodes with a large variation in values of the target variable, and therefore less accurate predictions.

The support vector regression method tends to perform poorly when there is a region with large variation, but there are few sample points to capture the variation. We suspect the support vectors do not adequately reflect the target values in such regions. This would make SVR not competetive with smaller data sets.

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 has the added benefit of providing the uncertainty in prediction, which could also 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 use of weights in calculating distances in LWKR can also provide insight into the relative importance of difference input dimensions.

In our comparisons, we did not include 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.

6 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 behavioural sciences, where simple linear regression models on predictor variables were used, the desire to avoid overfitting 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 obervations 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 [11] and for classificaton problems in high-dimensional simulated data sets in bio-informatics [18].

The contribution of our paper is to study how models built using diferent regression methods perform in small data sets, both in terms of prediction, as well as practical use in the domain of additive manufacturing.

7 Conclusions

In this paper, we presented an empirical study on the performance of different machine learning methods for regression in small data sets. We used the problem domain of additive manufacturing. where both computer simulations and experiments are expensive, resulting in small data sets. Our focus was on code surrogates that are models built using data from simulations. Using different metrics such as prediction accuracy and practical application in the problem domain, we compared five different regression algorithms. In the contect of our data sets, we found that ensembles of regression trees and support vector regression did not perform well, but locally weighted kernel regression and multi-variate adaptive regression splines performed well. The best performance of that of Gaussian process models that were not only accurate, but also provided a measure of confidence in the prediction results.

8 Acknowledgment

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 [10], and GP [17]. The Eagar-Tsai data were generated using a code developed by David Macknelly.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

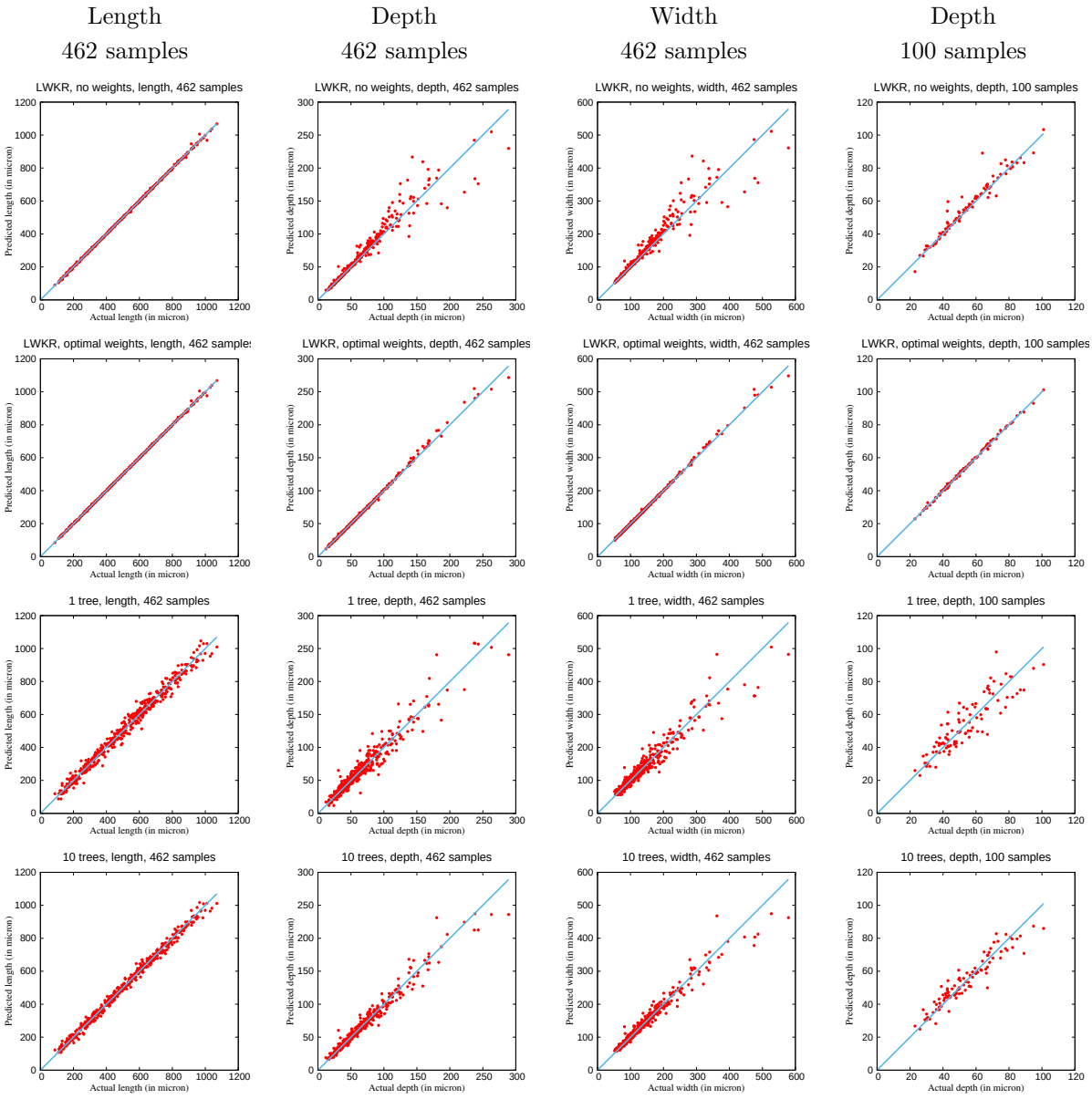


Figure 2: Predicted vs. actual values for the length, depth, and width for the Eagar-Tsai-462 data set and the depth using the Eagar-Tsai-100 data set for the different surrogate models using a leave-one-out approach. From top to bottom: locally weighted kernel regression with no feature weighting, locally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

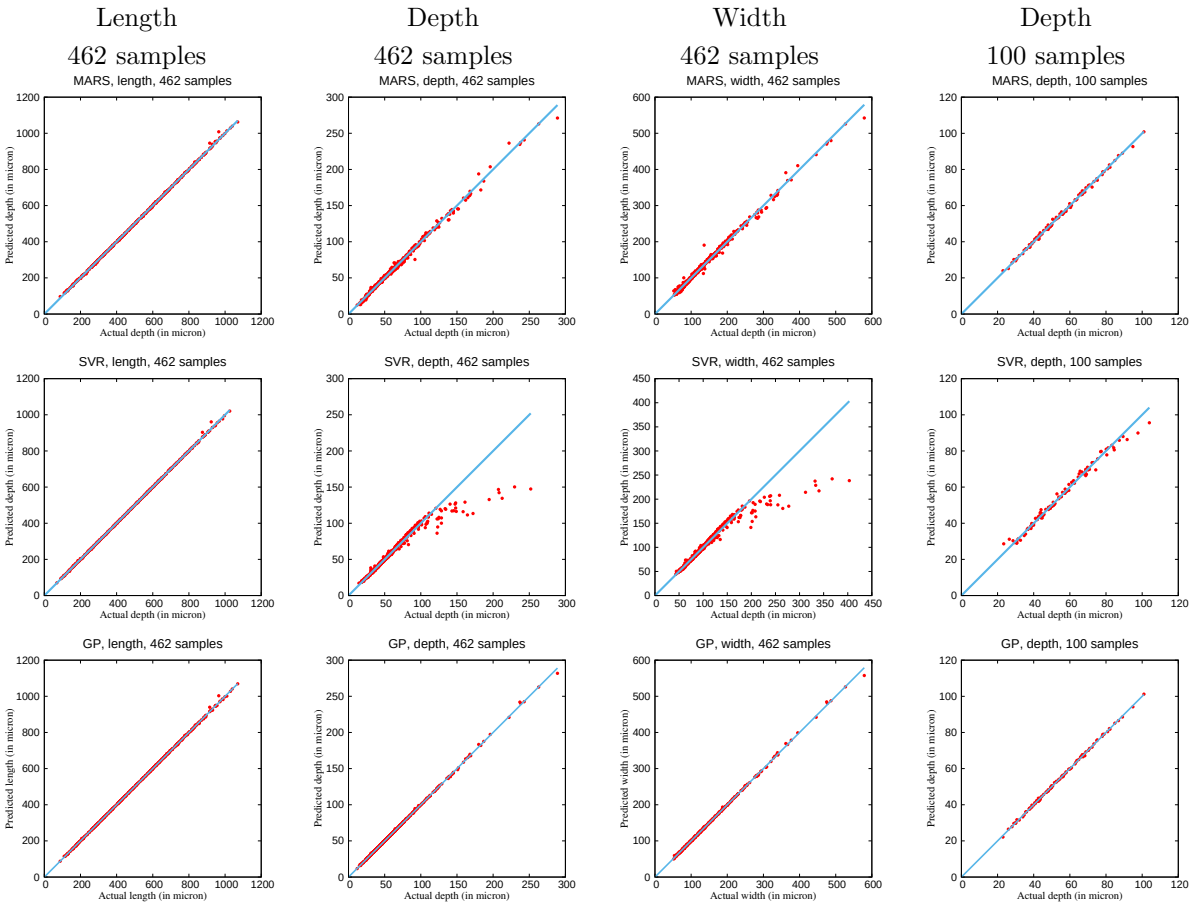


Figure 3: Predicted vs. actual values for the length, depth, and width for the Eagar-Tsai-462 data set and the depth using the Eagar-Tsai-100 data set for the different surrogate models using a leave-one-out approach. From top to bottom: multivariate adaptive regression splines (MARS), support vector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

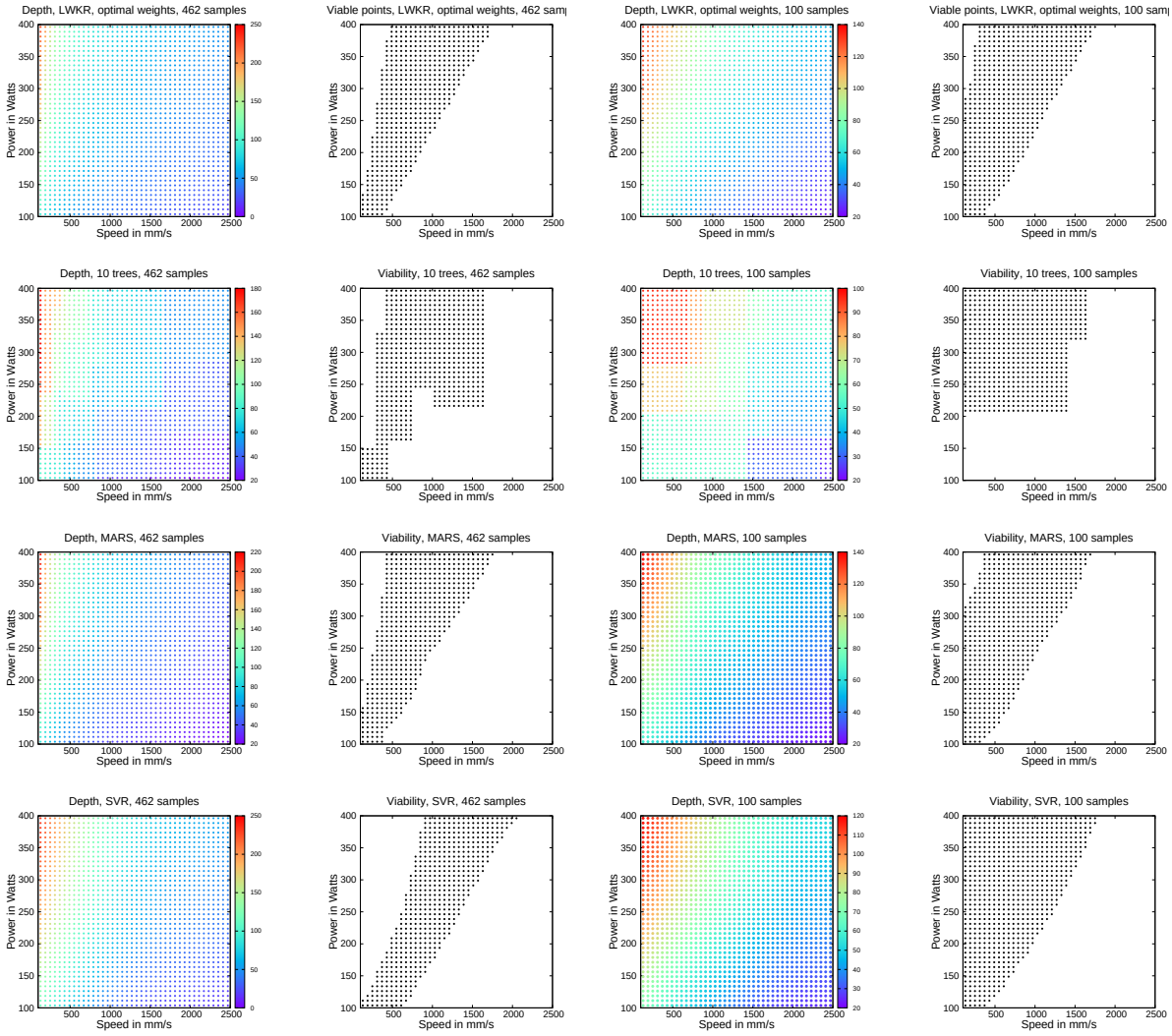


Figure 4: Comparing the viability regions identified by different surrogates for the Eagar-Tsai simulation as the number of samples points is reduced. Columns 1 and 3 are the prediction of the melt-pool depth using Eagar-Tsai-462 and Eagar-Tsai-100, respectively, while columns 1 and 4 are the corresponding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

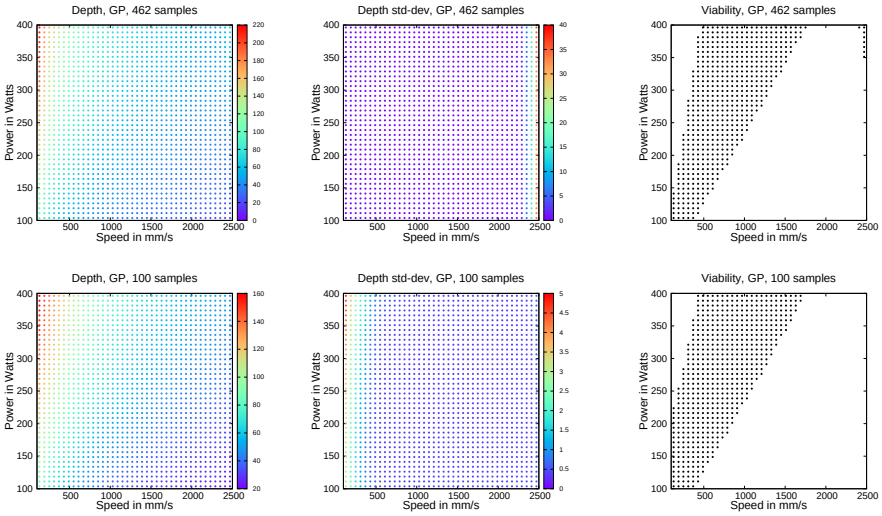


Figure 5: Comparing the viability regions identified by and Gaussian process for the Eagar-Tsai simulation as the number of samples points is reduced. The columns are from left to right: the prediction of the melt-pool depth, the standard deviation in the prediction, and the viability region. Results 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.

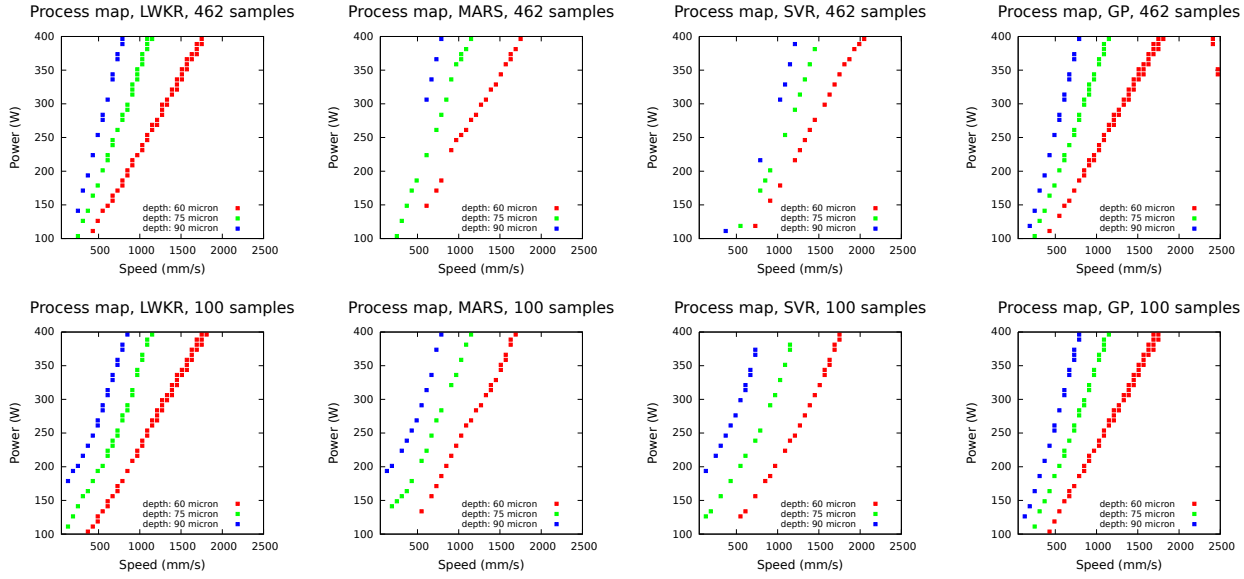


Figure 6: Process maps indicating points at a constant depth in the power-speed space. The columns from left to right are LWKR with optimal weights, MARS, SVR, and GP. Results for the top row are generated using the Eagar-Tsai-462 data set and for the bottom row, the Eagar-Tsai-100 data set.

References

[1] ACME: Accelerated Climate Modeling for Energy Web Page, 2016. `https://climatemodeling.science.energy.gov/projects/accelerated-climate-modeling-energy`.

[2] Atkeson, C., Schaal, S. A., and Moore, A. W. Locally weighted learning. *AI Review 11* (1997), 75–133.

[3] Austin, P. C., and Steyerberg, E. W. The number of subjects per variable required in linear regression analyses. *Journal of Clinical Epidemiology 68* (2015), 627–636.

[4] Babyak, M. A. What you see may not be what you get: a brief, non-technical introduction to overfitting in regression-type models. *Psychosomatic Medicine 66* (2004), 411–421.

[5] Bennett, K. P., and Mangasarian, O. L. Robust linear programming discrimination of two linearly inseparable sets. *Optimization Methods and Software 1*, 1 (1992), 23–34.

[6] Beuth, J., et al. Process mapping for qualification across multiple direct metal additive manufacturing processes. In *International Solid Freeform Fabrication Symposium, An Additive Manufacturing Conference, D. Bourell (Ed.), University of Texas at Austin, Austin, Texas* (2013), pp. 655–665.

[7] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. CRC Press, Boca Raton, FL, 1984.

[8] Burl, M. C., et al. Automated knowledge discovery from simulators. In *Proceedings, Sixth SIAM International Conference on Data Mining* (2006), pp. 82–93.

[9] Carriera-Perpiñán, M. A. A review of dimension reduction techniques. Tech. rep., Technical Report CS-96-09, Department of Computer Science, University of Sheffield, UK, 1996.

[10] Chang, C.-C., and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2* (2011), 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[11] Chapelle, O., Vapnik, V., and Bengio, Y. Model selection for small sample regression. *Machine Learning 48*, 1 (2002), 9–23.

[12] 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. *Assessing the Reliability of Complex Models: Mathematical and Statistical Foundations of Verification, Validation, and Uncertainty Quantification*. The National Academies Press, 2012.

[13] Eagar, T., and Tsai, N. Temperature-fields produced by traveling distributed heat-sources. *Welding Journal 62* (1983), S346–S355.

[14] Ebden, M. Gaussian process for regression and classification: A quick introduction. Available at http://arxiv.org/abs/1505.02965, submitted May 2015, 2008.

[15] Fang, K.-T., Li, R., and Sudjianto, A. *Design and Modeling for Computer Experiments*. Chapman and Hall/CRC Press, Boca Raton, FL, 2005.

[16] Friedman, J. H. Multivariate adaptive regression splines. *Ann. Statist. 19*, 1 (03 1991), 1–67.

[17] GPy. GPy: A Gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

[18] Guo, Y., Graber, A., McBurney, R. N., and Balasubramanian, R. Sample size and statistical power considerations in high-dimensionality data settings: a comparative study of classification algorithms. *BMC Bioinformatics 11* (2010).

[19] Kamath, C. *Scientific Data Mining: A Practical Perspective*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2009.

[20] Kamath, C. Data mining and statistical inference in selective laser melting. *International Journal of Advanced Manufacturing Technology 86* (September 2016), 1659–1677.

[21] Kamath, C., and Cantú-Paz, E. Creating ensembles of decision trees through sampling. In *Proceedings of the 33-rd Symposium on the Interface: Computing Science and Statistics* (2001).

[22] Kamath, C., El-dasher, B., Gallegos, G. F., King, W. E., and Sisto, A. Density of additively-manufactured, 316L SS parts using laser powder-bed fusion at powers up to 400 W. *International Journal of Advanced Manufacturing Technology 74* (2014), 65–78.

[23] King, W. E., Barth, H. D., Castillo, V. M., Gallegos, G. F., Gibbs, J. W., Hahn, D. E., Kamath, C., and Rubenchik, A. M. Observation of keyhole-mode laser melting in laser powder-bed fusion additive manufacturing. *Journal of Materials Processing Technology 214* (2014), 2915–2925.

[24] Kleijnen, J. P. C. *Design and Analysis of Simulation Experiments*. Springer, New York, NY, 2008.

[25] Mitchell, D. P. Spectrally optimal sampling for distribution ray tracing. *Computer Graphics 25*, 4 (1991), 157–164.

[26] Oehlert, G. W. *A First Course in Design and Analysis of Experiments*. W. H. Freeman, 2000. Available from `http://users.stat.umn.edu/~gary/Book.html`.

[27] Owen, A. B. Quasi-monte carlo sampling. Course notes from Siggraph 2003 course. Available from http://www-stat.stanford.edu/ owen/reports/.

[28] Owen, A. B. Latin supercube sampling for very high-dimensional simulations. *ACM Transactions on Modeling and Computer Simulations 8*, 1 (1998), 71–102.

[29] Qian, Y., et al. Uncertainty quantification in climate modeling and projection. *Bulletin of the American Meteorological Society 97*, 5 (May 2016), 821–824.

[30] Rasmussen, C. E., and Williams, C. K. I. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[31] Rokach, L. *Pattern Classification Using Ensemble Methods*. World Scientific Publishing, Singapore, 2010.

[32] Rokach, L., and Maimon, O. *Data Mining with Decision Trees: Theory and Applications*. World Scientific Publishing, Singapore, 2014.

[33] Rudy, J. Py-earth. `https://http://contrib.scikit-learn.org/py-earth/`, 2013.

[34] Schölkopf, B., Burges, C., and Vapnik, V. Extracting support data for a given task. In *Proceedings, First International Conference on Knowledge Discovery & Data Mining, Menlo Park* (Menlo Park, CA, 1995), AAAI Press, pp. 252–257.

[35] Schölkopf, B., Smola, A. J., Williamson, R. C., and Bartlett, P. L. New support vector algorithms. *Neural Comput. 12*, 5 (May 2000), 1207–1245.

[36] Shiflet, A. B., and Shiflet, G. W. *Introduction to Computational Science: Modeling and Simulation for the Sciences*. Princeton University Press, Princeton, NJ, 2006.

[37] Vapnik, V. N. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[38] Verhaeghe, F., Craeghs, T., Heulens, J., and Pandalaers, L. A pragmatic model for selective laser melting with evaporation. *Acta Materialia 57* (2009), 6006–6012.

[39] Yadroitsev, I., Gusarov, A., Yadroitsava, I., and Smurov, I. Single track formation in selective laser melting of metal powders. *Journal of Materials Processing Technology 210* (2010), 1624–1631.