Novel Meshes for Multivariate Interpolation and Approximation

Thomas C. H. Lux
Dept. of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia
tchlux@vt.edu

Layne T. Watson
Depts. of Computer Science, Mathematics, and Aerospace & Ocean Engineering
Virginia Polytechnic Institute and State University

Tyler H. Chang
Jon Bernard
Bo Li
Xiaodong Yu
Dept. of Computer Science
Virginia Polytechnic Institute and State University

Li Xu
Dept. of Statistics
Virginia Polytechnic Institute and State University

Godmar Back
Ali R. Butt
Kirk W. Cameron
Danfeng Yao
Dept. of Computer Science
Virginia Polytechnic Institute and State University

Yili Hong
Dept. of Statistics
Virginia Polytechnic Institute and State University

ABSTRACT

A rapid increase in the quantity of data available is allowing all fields of science to generate more accurate models of multivariate phenomena. Regression and interpolation become challenging when the dimension of data is large, especially while maintaining tractable computational complexity. This paper proposes three novel techniques for multivariate interpolation and regression that each have polynomial complexity with respect to number of instances (points) and number of attributes (dimension). Initial results suggest that these techniques are capable of effectively modeling multivariate phenomena while maintaining flexibility in different application domains.

KEYWORDS

Interpolation, Approximation, Splines, Multivariate, Regression

ACM Reference Format:
Thomas C. H. Lux, Layne T. Watson, Tyler H. Chang, Jon Bernard, Bo Li, Xiaodong Yu, Li Xu, Godmar Back, Ali R. Butt, Kirk W. Cameron, Danfeng Yao, and Yili Hong. 2018. Novel Meshes for Multivariate Interpolation and Approximation. In *ACM SE '18: ACM SE '18: Southeast Conference, March 29–31, 2018, Richmond, KY, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3190645.3190687

1 INTRODUCTION

Regression and interpolation are problems of considerable importance that find applications across many fields of science. Pollution and air quality analysis [8], energy consumption management [11], and student performance prediction [4] are a few examples of interdisciplinary applications of multivariate regression for predictive analysis. As discussed later, these techniques can also be applied to prediction problems related to high performance computing (HPC) file input/output (I/O), Parkinson's patient clinical evaluations [13], and forest fire risk assessment [3].

Multivariate interpolation is formally defined when there exists some function $f : \mathbb{R}^d \to \mathbb{R}$ and a set X of n points in $\mathbb{R}^d$ along with associated response values $f(x)$ for all $x \in X$. The problem is to construct an approximation $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ such that $\hat{f}(x) = f(x)$ for all $x \in X$. It is often the case that the form of the true underlying function f is unknown, however it is still desirable to construct an approximation $\hat{f}$ with minimum approximation error at $y \notin X$.

Multivariate regression is often used when the underlying function is presumed to be stochastic, or stochastic error is introduced in the evaluation of f. Hence, multivariate regression relaxes the conditions of interpolation by minimizing the error in $\hat{f}$ at $x \in X$ while maintaining some parametric form with parameters P. This can be written as $\min_P ||\hat{f}(X) - f(X)||$, where $f(X)$ is a vector of $f(x)$ for all $x \in X$ and $|| \cdot ||$ is an appropriate measure. The difficult question in the case of regression is often what parametric form to adopt for any given application. This paper proposes basis functions with overlapping regions of support as the general parametric form.

As the dimension of data increases, the number of possible interactions between dimensions grows exponentially. Quantifying all possible interactions becomes intractable and hence beyond three-dimensional data, mostly linear models are used. That is not to say nonlinear models are absent, but nonlinearities are often either preconceived or model pairwise interactions between dimensions at most. Similarly, higher order models of single variables can be generated from the techniques proposed in this paper, however only first-order interactions between variables are considered.

Regression and interpolation have a considerable theoretical base in one dimension [2]. Splines in particular are well understood

ACM SE '18, March 29–31, 2018, Richmond, KY, USA

ACM ISBN 978-1-4503-5696-1/18/03…$15.00
https://doi.org/10.1145/3190645.3190687

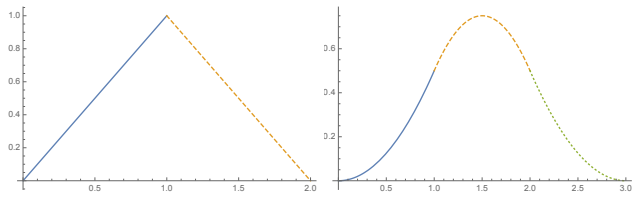

Figure 1: 1D linear (order 2) and quadratic (order 3) box splines with direction vector sets $\begin{pmatrix}1 & 1\end{pmatrix}$ and $\begin{pmatrix}1 & 1 & 1\end{pmatrix}$ respectively. Notice that these direction vector sets form the B-Spline analogues, order 2 composed of two linear components and order 3 composed of 3 quadratic components (colored and styled in plot).

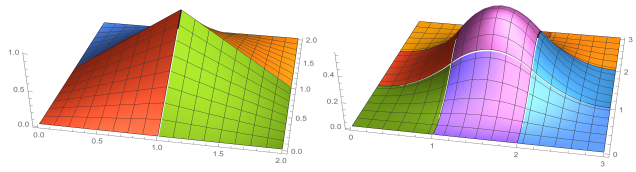

Figure 2: 2D linear (order 2) and quadratic (order 3) box splines with direction vector sets $\begin{pmatrix}I & I\end{pmatrix}$ and $\begin{pmatrix}I & I & I\end{pmatrix}$ respectively, where I is the identity matrix in two dimensions. Notice that these direction vector sets also produce boxes with order^2 subregions (colored in plot).

as an interpolation technique in one dimension [6], particularly B-splines. Fewer techniques have been discussed and evaluated for two dimensional problems. Though in two and three dimensions tensor product splines remain computable [14], tensor products have an unfortunate exponential scaling in parameterization with increasing dimension. Exponential scaling prohibits tensor products from being reasonably applied beyond three-dimensional data. In order to address this dimensional scaling challenge, C. de Boor and others have recently proposed box splines [7]. Two of the three meshes introduced in this work involve box-shaped regions of support and use box splines as the underlying basis functions.

1.1 Box Splines

A box spline in $\mathbb{R}^d$ is defined by its *direction vector set* A, composed of s d-vectors where $s \geq d$. Further, A will be written as a $d \times s$ matrix. The first m column vectors of A are denoted by A_m, $m \leq s$. A_d is required to be nonsingular. Consider the unit cube in s dimensions $Q_s = [0,1)^s$. $A_s\left(Q_s\right)$ is now the image (in d dimensions) of Q_s under the linear map A. This image is the region of support for the box spline defined by A_s in d dimensions. The box spline function in d dimensions for A_d is defined as

$$B(x \mid A_d) = \begin{cases} (\det(A_d))^{-1}, & x \in A_d(Q_d), \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

For A_s when $s > d$ the box spline is computed as

$$B(x \mid A_s) = \int_0^1 B(x - t v_s \mid A_{s-1}) dt, \tag{2}$$

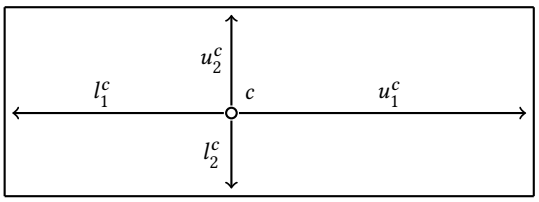

Figure 3: An example box in two dimensions with anchor c, upper widths u_1^c, u_2^c, and lower widths l_1^c, l_2^c. Notice that c is not required to be equidistant from opposing sides of the box, that is $u_i^c \neq l_i^c$ is allowed.

where v_s is the sth direction vector of A.

The application of box splines presented in this paper always utilizes the d-dimensional identity matrix as A_d. This simplifies the computation in Equation 1 to be the characteristic function for the unit cube. Composing A strictly out of k repetitions of the identity matrix forms the kth order B-spline with knots located at $0, 1, \ldots, k-1, k$ along each axis (see Figure 1). Furthermore, while the number of subregions for the kth order d-dimensional box spline grows as k^d (see Figure 2), the symmetry provided by direction vector sets composed of repeated identity matrices allows the computation of box splines to be simplified. The value of a box spline at any location is then the product of all axis-aligned 1-dimensional kth order box splines.

The box splines as presented are viable basis functions. Each box spline can be shifted and scaled without modifying the underlying computation (similar to wavelets), yet the underlying computation is simple and scales linearly with dimension. For a more thorough introduction and exploration of box splines in their more general form, readers are referred to [7].

2 INTERPOLATION AND REGRESSION

Throughout this section, the notation will be reused from Section 1. $X \subset \mathbb{R}^d$ is a finite set of points with known response values $f(x)$ for all $x \in X$. Also let $L, U \in \mathbb{R}^d$ define a bounding box for X such that $L < x < U$ for all $x \in X$.

Define a box $b^c = (c, l^c, u^c)$ in d dimensions with anchor $c \in \mathbb{R}^d$, lower width vector $l^c \in \mathbb{R}^d_+$, and upper width vector $u^c \in \mathbb{R}^d_+$ (where u_i^c refers to the ith component of u^c). A visual example of a box in two dimensions can be seen in Figure 3. Now, define a componentwise rescaling function $g : \mathbb{R}^d \to \mathbb{R}^d$ at point $x \in \mathbb{R}^d$ to be

$$\left(g^c(x)\right)_r = \frac{k}{2}\left(1 - \frac{(x_r - c_r)_-}{l_r^c} + \frac{(x_r - c_r)_+}{u_r^c}\right), \tag{3}$$

where $y_+ = \max\{y, 0\}$, $y_- = (-y)_+$, k is the order of the box spline as described in Section 1.1. Finally, each box spline in a box mesh can be evaluated as $B^c(x) = B(g^c(x) \mid A)$ presuming the order of approximation implies A. Both box meshes described in the following subsections use box spline basis functions of this form.

A notable property of boxes defined with the linear rescaling function g^c, is that C^0 and C^1 continuity of the underlying box spline are maintained. C^0 continuity is maintained through scaling. C^1 continuity is maintained for all box splines with C^1 continuity (order ≥ 3) because the scaling discontinuity is located at c, where

all box splines (of the presented form) have first derivative zero. All continuity beyond the first derivative is lost through the rescaling function g^c.

2.1 Max Box Mesh

The first of the three meshes produces a set of boxes around chosen control points and each box has maximal distance between the control point and the nearest side of the box. This centrality property is one mechanism for creating the largest reasonable regions of support for the underlying basis functions. The individual boxes are constructed via the following procedure given a set of control points $C \subseteq X$, $c^{(i)} \in C$,

(1) Initialize a box $b^{c^{(1)}} = \left(c^{(1)}, (c^{(1)} - L), (U - c^{(1)})\right)$.
(2) Identify $c^{(i)}$ over $\left\{j \;\middle|\; j \neq 1, B^{c^{(1)}}\left(c^{(j)}\right) \neq 0\right\}$ that minimizes $\left\|c^{(j)} - c^{(1)}\right\|_\infty$.
(3) Change the the box $b^{c^{(1)}}$ along the first dimension r such that $\left\|c^{(1)} - c^{(i)}\right\|_\infty = \left|c^{(1)} - c^{(i)}\right|_r$, to exclude $c^{(i)}$ from the support of $B^{c^{(1)}}$.
(4) Repeat steps 2 and 3 until no point in C is in the support of $B^{c^{(1)}}$ (at most $2d$ times, once for each boundary of a box).

The same process is used to construct boxes around all control points in C. In order to improve the generality of the approximation, a set of control points is initially chosen to be well-spaced using a statistical method from [1]:

(1) Generate a sequence of all pairs of points sorted by ascending pairwise Euclidean distance between points $\left(x^{(i_1)}, x^{(j_1)}\right)$, $\left(x^{(i_2)}, x^{(j_2)}\right), \ldots$, so that $\left\|x^{(i_k)} - x^{(j_k)}\right\|_2 \leq \left\|x^{(i_{k+1})} - x^{(j_{k+1})}\right\|_2$.
(2) Sequentially remove points from candidacy until only $|C|$ remain by randomly selecting a single point from each pair $\left(x^{(i_m)}, x^{(j_m)}\right)$ for $m = 1, \ldots$ if both $x^{(i_m)}$ and $x^{(j_m)}$ are still candidates for removal.

Once the boxes for a max box mesh have been constructed, the parameters can be identified via a least squares fit. The max box mesh (denoted *MBM*) is used to generate a $|X| \times |C|$ matrix M of box spline basis function evaluations at all points in X. The solution to the least squares problem $\min_P \left\|M\,P - f(X)\right\|_2$ is the parameterization of *MBM*. When $C = X$, M is the $|X| \times |X|$ identity matrix, making the max box mesh approximation $\hat{f}$ an interpolant.

While setting the number of boxes equal to the number of points causes the max box mesh to be an interpolant, the generality of the max box mesh approximation can often be improved by bootstrapping the selection of control points. Given a user-selected batch size $s \leq |X|$, start with s well-spaced control points. Next, measure the approximation error at $x \notin C$ and if the error is too large (determined by user), pick s points at which the magnitude of approximation error is largest, add those points to C, and recompute the max box mesh. The user is left to decide the batch size s and the error tolerance based on validation performance and computability. This work uses a batch size of one.

Definition 2.1. The hyperplane $x_r = c_r + u_r^c$ is the upper boundary of box b^c along dimension r, and similarly $x_r = c_r - l_r^c$ is the lower boundary of box b^c. When the anchor point y, for some box b^y, lies in the hyperplane (and facet of box b^c) defining either boundary along dimension r of b^c it is said that b^y *bounds* b^c in dimension r and is denoted $\left(b^c \;\middle|_r\; b^y\right)$.

Throughout all experiments and all repeated trials conducted for this study, all tested interpolation points were covered by at least one box in the *MBM*. However, it is possible for the *MBM* to not form a covering of $[L, U]$ when there are cyclic boundaries. Consider the following example in three dimensions:

$$\begin{aligned} C &= \left\{(0,0,0), (1,0,2/3), (1,1,4/3)\right\}, \\ b^{c^{(1)}} &= \left((0,0,0),\ (*,*,*),\ (1,*,4/3)\right), \\ b^{c^{(2)}} &= \left((1,0,2/3),\ (1,*,*),\ (*,1,*)\right), \\ b^{c^{(3)}} &= \left((1,1,4/3),\ (*,1,4/3),\ (*,*,*)\right). \end{aligned}$$

Asterisks are used to represent boxes that are not bounded by other boxes along some dimensions. The point $(2, 2, -3)$ is not in any of the max boxes defined above. In this case, there is a cycle in box boundaries that looks like $\left(b^{c^{(1)}} \;\middle|_1\; b^{c^{(2)}} \;\middle|_2\; b^{c^{(3)}} \;\middle|_3\; b^{c^{(1)}}\right)$. This example demonstrates that it is geometrically possible for the max box mesh to fail to cover a space, however experiments demonstrate that it is empirically unlikely.

The max box mesh remains a viable strategy for computed approximations. Given a maximum of c control points in d dimensions with n points, the computational complexities are: $O(c^2 d)$ for computing boxes, $O(cd^2 + d^3)$ for a least squares fit, and $O(n/s)$ for bootstrapping (which is multiplicative over the fitting complexities). Evaluating the max box mesh requires $O(cd)$ computations.

2.2 Iterative Box Mesh

The iterative box mesh (*IBM*) comprises box-shaped regions that each contain exactly one control point in their interior just as in the *MBM*. However, the mesh is a covering for $[L, U]$ by construction and places boxes in a way that reduces apparent error. The boxes are constructed via the following procedure given a finite set of points $X \subset \mathbb{R}^d$, where $C \subseteq X$ is the (initially empty) set of control points.

(1) Add the box that covers $[L, U]$ anchored at the most central point $x^{(k)} \in X$, add $x^{(k)}$ to C, and least squares fit the *IBM* model to all $x \in X$.
(2) Add a new box $[L, U]$ anchored at $x^{(i)} \notin C$ such that $\left|IBM(x^{(i)}) - f(x^{(i)})\right| = \max_{x \in X \setminus C} \left|IBM(x) - f(x)\right|$, reshaping all boxes $b^{x^{(j)}}$ that contain $x^{(i)}$ by bounding the first dimension r such that $\left|x_r^{(j)} - x_r^{(i)}\right| = \left\|x^{(j)} - x^{(i)}\right\|_\infty$ (also reshaping the box $b^{x^{(i)}}$ symmetrically), add $x^{(i)}$ to C, and then least squares fit the *IBM* model to all $x \in X$.
(3) Repeat Step 2 until model approximation error is below tolerance t.

Just as for the *MBM*, the parameters can be identified via a least squares fit. The iterative box mesh is used to generate a $|X| \times |C|$ matrix M of box spline function evaluations at all points in X. Now the box spline coefficients are the solution to the least squares problem $\min_P \left\|M\,P - f(X)\right\|_2$. Also as for the *MBM*, $C = X$ causes M to equal the $|X| \times |X|$ identity, making the iterative box mesh approximation $\hat{f}$ an interpolant.

As opposed to the max box mesh, the bootstrapping procedure is built into the iterative box mesh. The user is left to decide the most appropriate error tolerance, however a decision mechanism and analysis is presented in Section 3.2. As mentioned earlier, the iterative box mesh is a covering for $[L, U]$ by construction and this can be proved by an inductive argument.

An *IBM* least squares fit $\hat{f}(z) = \sum_j P_j B^{c^{(ij)}}(z)$ can generate approximations at new points $z \in Z \subset \mathbb{R}^d$ by evaluating $\hat{f}(z)$. The computational complexity for generating the mesh is $O(c^2nd)$ where c is the number of control points determined by the minimum error threshold and $n = |X|$. The computational complexity of evaluating the mesh at a single point is $O(cd)$.

2.3 Voronoi Mesh

The final of the three meshes utilizes 2-norm distances to define boundaries rather than max norm distances. A well-studied technique for classification and approximation is the nearest neighbor algorithm [5]. Nearest neighbor inherently utilizes the convex region $v^{x^{(i)}}$ (Voronoi cell [9]) consisting of all points closer to $x^{(i)}$ than any other point $x^{(j)}$. The Voronoi mesh smooths the nearest neighbor approximation by utilizing the Voronoi cells to define support via a generic basis function $V : \mathbb{R}^d \to \mathbb{R}_+$ given by

$$V^{x^{(i)}}(y) = \left(1 - \frac{\left\|y - x^{(i)}\right\|_2}{2\, d(y \mid x^{(i)})}\right)_+,$$

where $x^{(i)}$ is the center of the Voronoi cell, $y \in \mathbb{R}^d$ is an interpolation point, and $d(y \mid x^{(i)})$ is the distance between $x^{(i)}$ and the boundary of the Voronoi cell $v^{x^{(i)}}$ in the direction $y - x^{(i)}$. $V^{x^{(i)}}\left(x^{(j)}\right) = \delta_{ij}$ and $V^{x^{(i)}}$ has local support. While $V^{x^{(i)}}(x^{(i)}) = 1$, the 2 in the denominator causes all basis functions to go linearly to 0 at the boundary of the twice-expanded Voronoi cell. Note that this basis function is C^0 because the boundaries of the Voronoi cell are C^0. In the case that there is no boundary along the vector w, the basis function value is always 1.

While the cost of computing the exact Voronoi cells for any given set of points grows exponentially [10], the calculation of d is linear with respect to the number of control points and dimensions. Given any center $x^{(i)} \in \mathbb{R}^d$, set of control points $C \subseteq X$, and interpolation point $y \in \mathbb{R}^d$, $d\left(y \mid x^{(i)}\right)$ is the solution to

$$\max_{c \in C \setminus \{x^{(i)}\}} \frac{\left\|y - x^{(i)}\right\|_2}{2} \frac{y \cdot \left(c - x^{(i)}\right) - x^{(i)} \cdot \left(c - x^{(i)}\right)}{c \cdot \left(c - x^{(i)}\right) - x^{(i)} \cdot \left(c - x^{(i)}\right)}. \quad (4)$$

The parameters of the VM can now be computed exactly as for the *MBM* and *IBM*. The Voronoi mesh is used to generate a $|X| \times |C|$ matrix M of basis function evaluations at all points in X. Now the VM coefficients are the solution to the least squares problem $\min_P \left\|M\ P - f(X)\right\|_2$. When $X = C$, M is the identity making the mesh an interpolant. Bootstrapping can be performed with an identical procedure to that for the *IBM*.

(1) Pick the most central point $x^{(k)} \in X$ to be the first control point in C and fit the VM model to all $x \in X$.

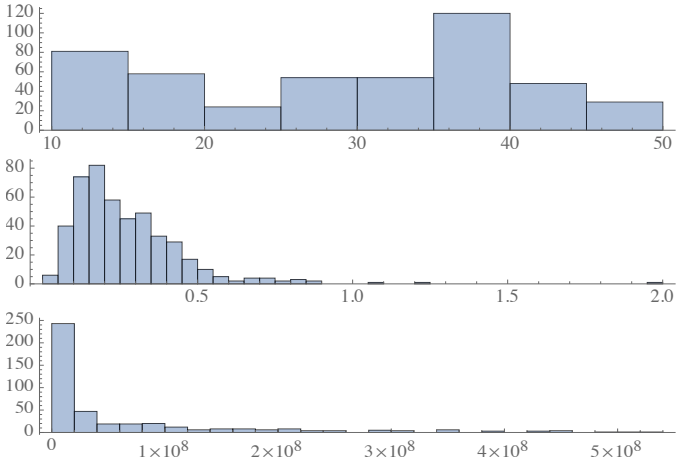

Figure 4: Histograms of Parkinsons (total UPDRS), forest fire (area), and HPC I/O (mean throughput) response values respectively. Notice that both the forest fire and HPC I/O data sets are heavily skewed.

(2) Identify a control point $x^{(i)} \notin C$ such that $\left|VM(x^{(i)}) - f(x^{(i)})\right| = \max_{x \in X \setminus C}\left|VM(x) - f(x)\right|$, add $x^{(i)}$ to C, and then fit the VM model to all $x \in X$.
(3) Repeat Step 2 until approximation error is below tolerance t.

Any VM is naïvely a covering for $[L, U]$, since any possible interpolation point will have a nearest neighbor control point. The computational complexity of evaluating a parameterized Voronoi mesh with c control points is $O(c^2d)$. Bootstrapping the generation of a Voronoi mesh requires $O(c^2nd)$ computations for a maximum number of basis functions c determined by the error threshold.

3 DATA AND ANALYSIS

In order to evaluate the proposed interpolation and approximation techniques, this paper utilizes three data sets of varying dimension and application. In the following subsections the sources and targets of each data set are described, as well as challenges and limitations related to interpolating and approximating these data sets. The distributions of response values being modeled can be seen in Figure 4. The preprocessing and approximation processes are described in Section 3.2.

3.1 Data Summary

3.1.1 High Performance Computing I/O ($n = 532, d = 4$). The first of three data sets is a four-dimensional data set produced by executing the IOzone benchmark from [12] on a homogeneous cluster of computers. The system performance data was collected by executing IOzone 40 times for each of a select set of system configurations. A single IOzone execution reports the max I/O file-read throughput seen. The 40 executions for each system configuration are converted to their mean, which is capable of being modeled by each of the multivariate approximation techniques presented in Section 2. The four dimensions being modeled to predict throughput mean are file size, record size, thread count, and CPU frequency.

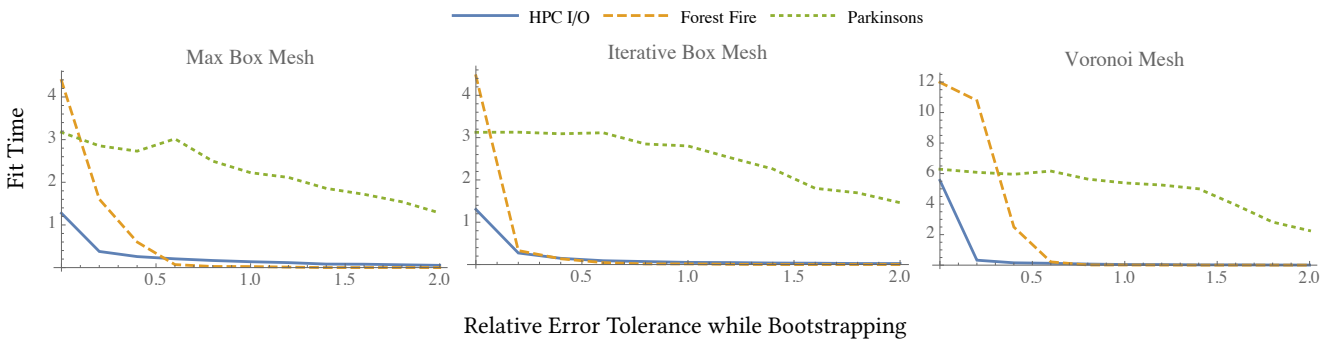

Figure 5: Time required to generate model fits for each technique with varying relative error tolerance during bootstrapping.

3.1.2 Forest Fire ($n = 517, d = 12$). The forest fire data set [3] describes the area of Montesinho park burned on specific days and months of the year in terms of the environmental conditions. The twelve dimensions being used to model burn area are the x and y spatial coordinates of burns in the park, month and day of year, the FFMC, DMC, DC, and ISI indices (see source for details), the temperature in Celsius, relative humidity, wind speed, and outdoor rain. The original analysis of this data set demonstrated it to be difficult to model, likely due to the skew in response values.

3.1.3 Parkinson's Telemonitoring ($n = 468, d = 16$). The final data set for evaluation [13] is derived from a speech monitoring study with the intent to automatically estimate Parkinson's disease symptom development in Parkinson's patients. The function to be predicted is a time-consuming clinical evaluation measure referred to as the UPDRS score. The total UPDRS score given by a clinical evaluation is estimated through 16 real numbers generated from biomedical voice measures of in-home sound recordings.

3.2 Performance Analysis

The performance of the approximation techniques varies considerably across the three evaluation data sets. Relative errors for the most naïve approximators such as nearest neighbor can range from zero to $\left(\max_x f(x) - \min_x f(x)\right) / \min_x f(x)$ when modeling a positive function $f(x)$ from data. Each of the approximation techniques presented remain within these bounds and all errors are presented in signed relative form $(\hat{f}(x) - f(x))/f(x)$. Before the models are constructed all data values (components $x_r^{(i)}$ of $x^{(i)} \in X$) are shifted and scaled to be in the unit cube $[0, 1]^d$, while the response values are taken in their original form. All models are evaluated with 10 random 80/20 splits of the data.

Each of the approximation techniques presented incorporates bootstrapping based on an allowable error tolerance t. An analysis of the effects of bootstrapping error tolerances on validation accuracy can be seen in Figure 6. The approximation meshes perform best on the forest fire and Parkinson's data sets when the error tolerance used for fitting is large (smoothing rather than interpolating), while near-interpolation generally produces the most accurate models for HPC I/O. Another performance result of note is that the *MBM* and *IBM* have very similar basis functions with largely different outputs.

Data Set	Technique	Tolerance	Average Error
HPC I/O	MBM	1.2	0.597
Forest Fire	MBM	1.8	3.517
Parkinson's	MBM	0.6	0.114
HPC I/O	IBM	0.4	0.419
Forest Fire	IBM	1.8	3.615
Parkinson's	IBM	1.8	0.121
HPC I/O	VM	0.2	0.382
Forest Fire	VM	1.0	4.783
Parkinson's	VM	2.0	1.824

Table 1: The optimal error tolerance bootstrapping parameters for each technique and each data set as well as the average absolute relative errors achieved by that tolerance. Notice that large relative error tolerances occasionally yield even lower evaluation errors, demonstrating the benefits of approximation over interpolation for noisy data sets.

The selection of bootstrapping error tolerance also effects the computation time required to fit each of the models to data. Figure 5 presents the time required to construct approximations for each model and each data set with varying t. The rapid reduction in computation time required for the forest fire and HPC I/O data sets suggests that large reductions in error can be achieved with relatively few basis functions. The Parkinson's data set however presents a more noisy response, with increasing number of basis functions reducing error much less quickly.

The distributions of errors experienced by each approximation technique when the optimal bootstrapping relative error tolerance is selected can be seen in Figure 7. HPC I/O exhibits the most normal approximation errors, which suggests that the models are converging on the random noise of the response for the data set. The worst relative approximation errors are produced by the Voronoi mesh on the forest fire data set. The small magnitude true response values contribute to the larger relative errors. Regardless, the *VM* errors are unacceptably large.

4 DISCUSSION

The bootstrapping procedure presented for each approximation technique still has much room for improvement. Initial analysis suggests that the appropriate relative error tolerance needs to be discovered empirically for each application of a modeling technique.

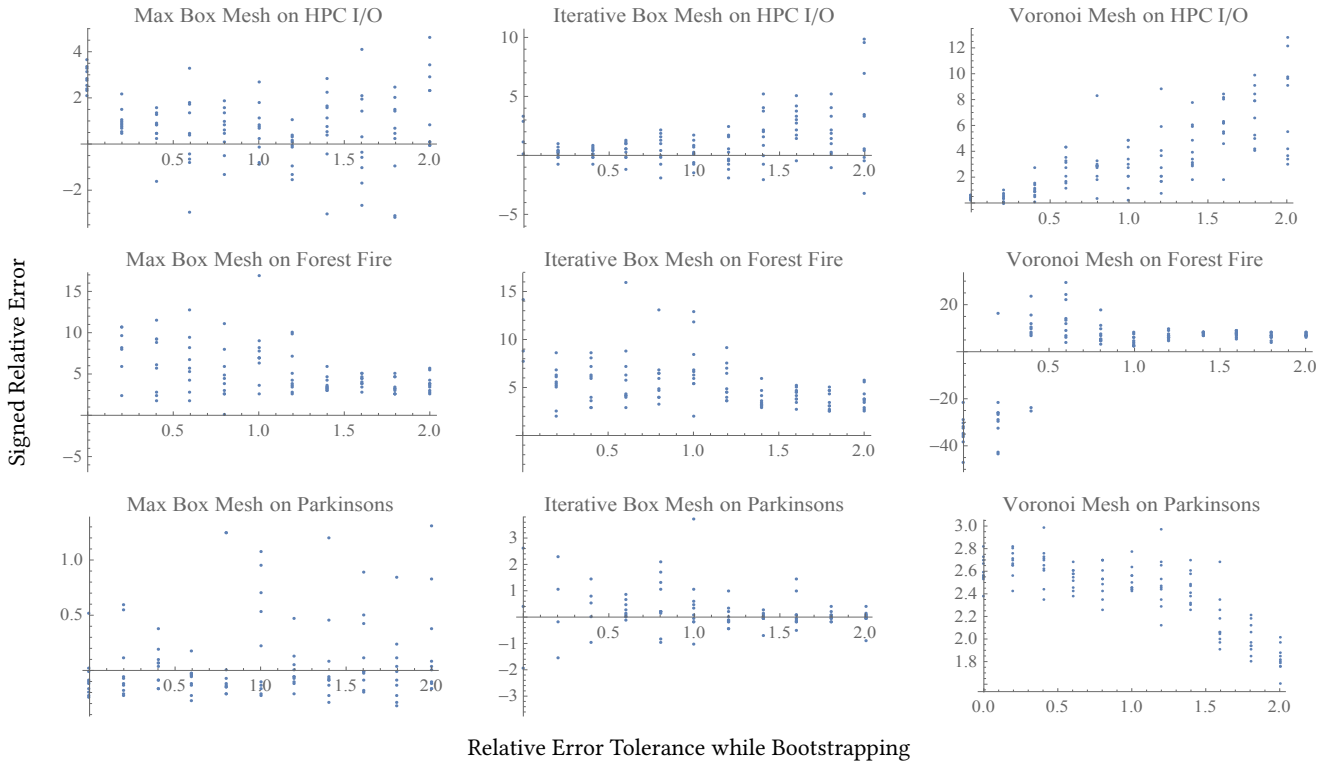

Figure 6: The performance of all three techniques with varied relative error tolerance for the bootstrapping parameter. The columns are for Max Box Mesh, Iterative Box Mesh, and Voronoi Mesh, respectively. The rows are for HPC I/O, Forest Fire, and Parkinson's respectively. Notice the techniques' behavior on the Parkinson's and Forest Fire data sets, performance increases with larger error tolerance.

Further analytic studies could arrive at methods for determining optimal error tolerances at runtime, however increases in runtime complexity may not be afforded in many applications.

The box-shaped basis functions and the construction algorithms used for the *MBM* and *IBM* could become a source of error when d (the dimension of the data X) is comparable to n (the number of known points). The blending regions in which multiple basis functions overlap are always axis aligned and in applications such as image analysis, any single dimension may be unsuitable for approximating the true underlying function. The Voronoi mesh attempts to address this problem by utilizing boundaries between points in multiple dimensions simultaneously. However, it is empirically unclear whether the true benefits of the *VM* are seen in applications where $d \ll n$.

Each of the case studies presented have fewer than 1000 points. The complexities of the presented approximation techniques are suitable for large dimension, but the increased complexity associated with brute-force bootstrapping currently prohibits their use on larger data sets. The Voronoi mesh in particular has a large complexity with respect to n which could be significantly improved via more greedy bootstrapping. While each technique requires less than ten seconds on average to produce a fit in the presented case studies, the fit time required quickly grows into minutes around 1000 points. While these initial results appear somewhat limiting, they demonstrate the viability of each mesh and leave room for further theoretical exploration of techniques to reduce the runtime complexity while maintaining the approximation power and flexibility.

5 CONCLUSION

The Max Box Mesh, Iterative Box Mesh, and Voronoi Mesh each provide novel strategies for effectively approximating multivariate phenomonon. The underlying constructions are theoretically straightforward, yet powerful and flexible. The computational complexities of each make them particularly suitable for applications in many dimensions, while the bootstrapping error tolerance parameter allows a balance between smoothing and interpolation to be explored empirically with each application.

5.1 Future Work

A thorough comparison with constituent multivariate approximation techniques including but not limited to, linear Shepard interpolation, multivariate adaptive regression splines, multilayer perceptron regression, and Delaunay triangulation constitutes future work. A more detailed study of alternative bootstrapping techniques may also provide valuable insight.

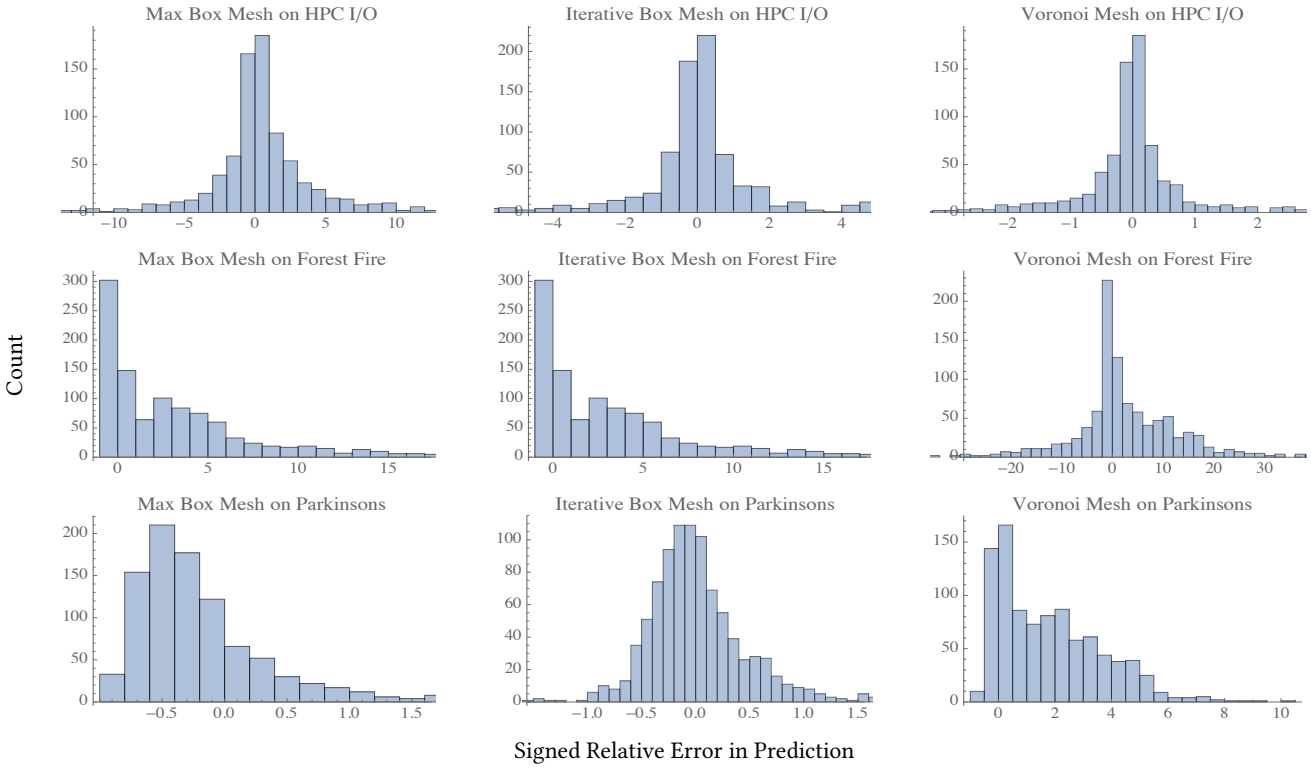

Figure 7: A sample of relative errors for all three techniques with optimal selections of error tolerance. The columns are for Max Box Mesh, Iterative Box Mesh, and Voronoi Mesh, respectively. The rows are for HPC I/O, Forest Fire, and Parkinson's respectively.

REFERENCES

[1] Brandon D. Amos, David R. Easterling, Layne T. Watson, William I. Thacker, Brent S. Castle, and Michael W. Trosset. 2014. Algorithm XXX: QNSTOP quasi-Newton algorithm for stochastic optimization. *Technical Report 14-02, Dept. of Computer Science, VPI&SU, Blacksburg, VA* (2014).

[2] Elliott Ward Cheney and William Allan Light. 2009. *A Course in Approximation Theory*. Vol. 101. American Mathematical Soc.

[3] Paulo Cortez and Aníbal de Jesus Raimundo Morais. 2007. A data mining approach to predict forest fires using meteorological data. *13th Portuguese Conference on Artificial Intelligence* (2007).

[4] Paulo Cortez and Alice Maria Gonçalves Silva. 2008. Using data mining to predict secondary school student performance. *Proceedings of 5th Annual Future Business Technology Conference, Porto* (2008).

[5] Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13, 1 (1967), 21–27.

[6] Carl De Boor, Carl De Boor, Etats-Unis Mathématicien, Carl De Boor, and Carl De Boor. 1978. *A practical guide to splines*. Vol. 27. Springer-Verlag New York.

[7] Carl De Boor, Klaus Höllig, and Sherman Riemenschneider. 2013. *Box splines*. Vol. 98. Springer Science & Business Media.

[8] S De Vito, E Massera, M Piga, L Martinotto, and G Di Francia. 2008. On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sensors and Actuators B: Chemical* 129, 2 (2008), 750–757.

[9] G Lejeune Dirichlet. 1850. Über die Reduction der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen. *Journal für die Reine und Angewandte Mathematik* 40 (1850), 209–227.

[10] Mathieu Dutour Sikirić, Achill Schürmann, and Frank Vallentin. 2009. Complexity and algorithms for computing Voronoi cells of lattices. *Math. Comp.* 78, 267 (2009), 1713–1731.

[11] Dimitris Lazos, Alistair B Sproul, and Merlinde Kay. 2014. Optimisation of energy management in commercial buildings with weather forecasting inputs: A review. *Renewable and Sustainable Energy Reviews* 39 (2014), 587–603.

[12] W. D. Norcott. 2017. IOzone Filesystem Benchmark. (2017). http://www.iozone.org [Online; accessed 2017-11-12].

[13] Athanasios Tsanas, Max A Little, Patrick E McSharry, and Lorraine O Ramig. 2010. Accurate telemonitoring of Parkinson's disease progression by noninvasive speech tests. *IEEE Transactions on Biomedical Engineering* 57, 4 (2010), 884–893.

[14] G unther Greiner and Kai Hormann. 1996. Interpolating and approximating scattered 3D-data with hierarchical tensor product B-splines. In *Proceedings of Chamonix*. 1.