Precision Aggregated Local Models

Adam Michael Edwards

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Statistics

Robert B. Gramacy, Chair
David M. Higdon
Leanna L. House
Shyam Ranganathan

December 2, 2020
Blacksburg, Virginia

Keywords: approximate kriging neighborhoods, Gaussian process surrogate, nonparametric regression, nearest neighbor, boosting, sequential design, active learning

Precision Aggregated Local Models

Adam Michael Edwards

(ABSTRACT)

Large scale Gaussian process (GP) regression is infeasible for larger data sets due to cubic scaling of flops and quadratic storage involved in working with covariance matrices. Remedies in recent literature focus on divide-and-conquer, e.g., partitioning into sub-problems and inducing functional (and thus computational) independence. Such approximations can speedy, accurate, and sometimes even more flexible than an ordinary GPs. However, a big downside is loss of continuity at partition boundaries. Modern methods like local approximate GPs (LAGPs) imply effectively infinite partitioning and are thus pathologically good and bad in this regard. Model averaging, an alternative to divide-and-conquer, can maintain absolute continuity but often over-smooth, diminishing accuracy. Here I propose putting LAGP-like methods into a local experts-like framework, blending partition-based speed with model-averaging continuity, as a flagship example of what I call precision aggregated local models (PALM). Using N_C LAGPs, each selecting n from N data pairs, I illustrate a scheme that is at most cubic in n, quadratic in N_C, and linear in N, drastically reducing computational and storage demands. Extensive empirical illustration shows how PALM is at least as accurate as LAGP, can be much faster in terms of speed, and furnishes continuous predictive surfaces. Finally, I propose sequential updating scheme which greedily refines a PALM predictor up to a computational budget, and several variations on the basic PALM that may provide predictive improvements.

Precision Aggregated Local Models

Adam Michael Edwards

(GENERAL AUDIENCE ABSTRACT)

Occasionally, when describing the relationship between two variables, it may be helpful to use a so-called "non-parametric" regression that is agnostic to the function that connects them. Gaussian Processes (GPs) are a popular method of non-parametric regression used for their relative flexibility and interpretability, but they have the unfortunate drawback of being computationally infeasible for large data sets. Past work into solving the scaling issues for GPs has focused on "divide and conquer" style schemes that spread the data out across multiple smaller GP models. While these model make GP methods much more accessible to large data sets they do so either at the expense of local predictive accuracy of global surface continuity. Precision Aggregated Local Models (PALM) is a novel divide and conquer method for GP models that is scalable for large data while maintaining local accuracy and a smooth global model. I demonstrate that PALM can be built quickly, and performs well predictively compared to other state of the art methods. This document also provides a sequential algorithm for selecting the location of each local model, and variations on the basic PALM methodology.

Contents

List of Figures

Chapter 1

Introduction

Perhaps the most common problem in statistics is to describe the relationship between variables. The most common method of estimating this relationship assumes that the form is known, i.e. $y = mx + b$, a line. In this case, the dependency between the variables x and y can be described by estimating a slope and an intercept. In a broader context, as long as the functional form between variables is known, the specific relationship can be learned by estimating the parameters of that function. This falls into a class of regression problems called *parametric regression*, and it requires some prior knowledge about how the variables are related. In many real world contexts, for example exponential decay of radioactive elements over time, the form of the relationship is known, and can be well described by parametric regression.

It is more difficult to obtain an accurate measure of the dependency between variables when the form of the function is unknown. This class of models, known as *non-parametric regression* is comprised of flexible models that learn the form of the relationship from the data itself.

One of the simplest forms of non-parametric regression is a decision tree. The basic concept is to split up the input space, and assign a different prediction to each portion. This concept is illustrated in the left panel of Figure 1.1. A set of 200 points are generated from a sine wave. Next the input is recursively split into 16 separate sections, and the average of each section is used as the prediction for that area of the input space. This creates a step

function with discontinuous jumps between adjacent predictions. By using an ensemble of many different trees with higher fidelity splits across the input space, we can obtain a much closer approximation to the true function. The right panel shows the Gradient Boosting Machine [13], which successively builds step functions to correct the error from previous trees. By using an ensemble, we are able to achieve a relatively close approximation to the true function.

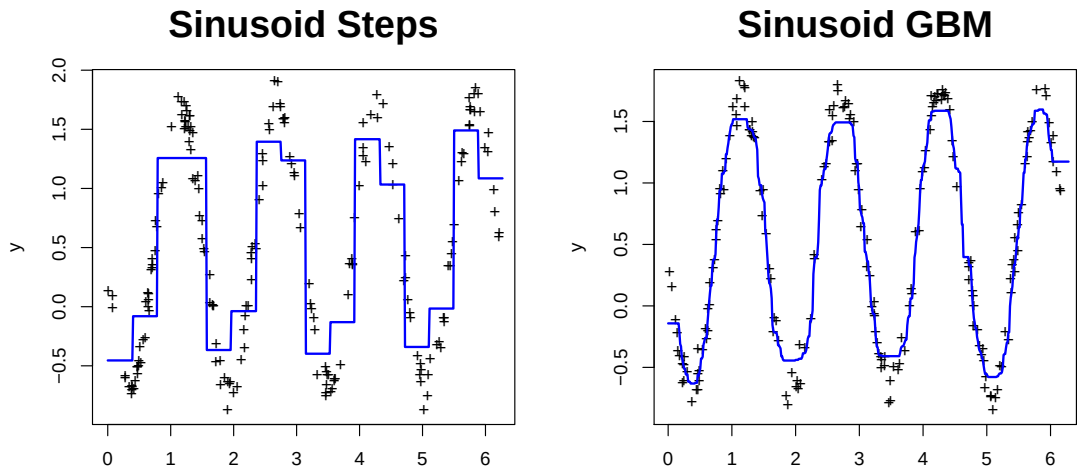

Figure 1.1: Left: Estimation of a sine wave by a single decision tree. Right: a Decision tree ensemble, the gradient boosting model, applied to the same sine wave.

The obvious concern with tree based models is that despite the ability to increase the fidelity, they are discontinuous by nature. This can be observed by the relatively flat areas and the sharp drops even in the right panel of Figure 1.1.

Regression splines provide a model that allows for continuity while learning fairly complicated functions. Spline models estimate a function that can closely match the true function for a small area of the input space and choose a number of *knots* throughout that function as adjustments to this function. The left panel of Figure 1.2 shows how powerful this method can be. The basic function estimated here is $y = \beta_0 + \beta_1 x + \beta_2 x^2$. At each of the know points, indicated by the vertical blue lines, an adjustment to the linear and quadratic component of

the model is made. In this example I chose the knot points to closely match the oscillation of the actual sine wave, and the result is a function estimation that is nearly smooth, and fairly close to the truth.

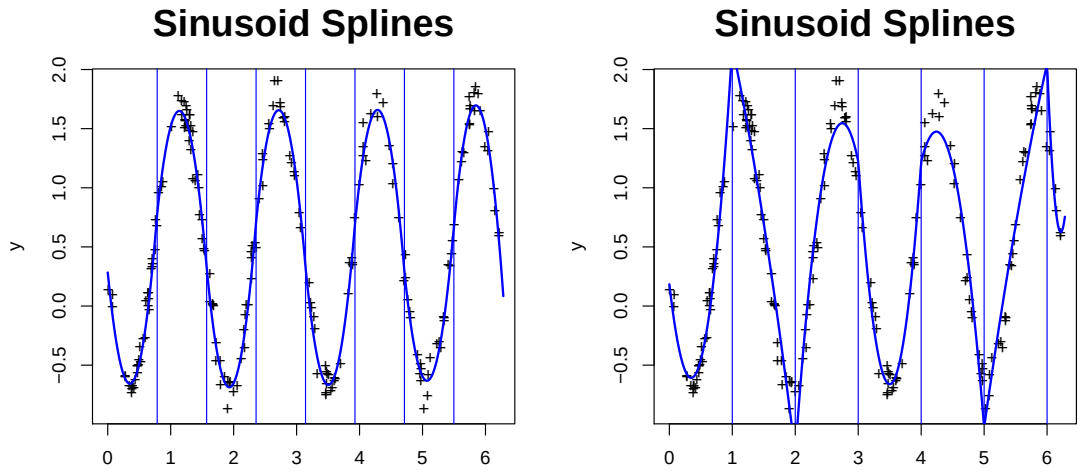

Figure 1.2: Left: Splines with appropriately chosen knot points applied to a sine wave. Right: A spline model with too few splines.

The main issue for models of this type is that the number and placement of the areas where the function is allowed to change has a large impact on the predictive performance of the model. In the right panel of Figure 1.2, I fit a model following the same procedure, except that there are fewer knot points, and their location is less ideal. The result is a considerably worse fit.

We desire a model that is flexible enough to capture the relationship between variables without a lot of tuning, or knowledge of the function, but maintains smoothness. Gaussian Processes (GPs) fill that gap. The most basic assumption of a GP is that if two points are close to each other in the input space, they should have a similar response value. We learn about the entire function by a distance based correlation with all of the points we have observed.

Gaussian processes provide a good fit of many functions with minimal tuning. Figure

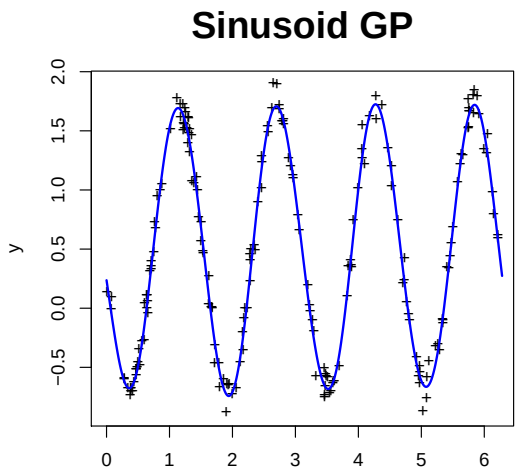

Figure 1.3: A basic Gaussian Process fit to a sine wave.

1.3 shows how powerful they can be. They are widely used for their flexibility and accurate uncertainty quantification. In Geospatial statistics GPs provide smoothly transitioning predictions that are appropriate for many real life Geospatial problems. In Machine Learning they provide much of the flexibility of more complex models, like neural networks, while maintaining a considerable advantage in interpretability.

The other methods briefly outlined here give some form of approximation for the actual underlying function relating the variables. As such, once the estimation is completed, the data are no longer needed. This is the major area where GPs fall short. Estimation and prediction is only performed by the strength of the relationship to observed data, necessitating some representation of each data point in the final model.

The reliance on all of the data points causes GPs to scale very poorly with the amount of data collected. In fact the above illustrations are kept intentionally small to accommodate GP estimation. The vast majority of research on GP methods focuses, at least in some part, on fixing the issue of rapidly increasing computational complexity and storage requirements for GP implementation. The method I outline here follows in this tradition while allowing for increased flexibility over the basic GP model.

The rest of this document lays out the research tradition and ideological innovation of Precision Aggregated Local Models (PALM). Chapter 2 covers Gaussian Process research that attacks similar problems to my method. This provides a general background on the research tradition into which PALM fits. Specific framework for my PALM method is developed in Chapter 3 with a brief reprise of several methods from which it borrows. I will provide a general framework for a wide swath of GP methods before establishing PALM's place within it. The concept of PALM is established outside of the GP framework before specific details as they apply to GPs are given. Chapter 4 provides an algorithmic way to allocate computational resources appropriately within the PALM framework. Finally, in Chapter 5, I lay out future work to improve the PALM method and make it available to the wider body of researchers.

Chapter 2

Review

In this section I review Gaussian Process models and advancements that lead to my innovation. Starting with Section 2.1 I go over basic GP concepts including formulation, distribution, and estimation. Section 2.2 covers approximations to a full GP. The specific research tradition into which my method falls, divide and conquer schemes, is covered in Section 2.3. Finally an in depth of review of Local Approximate Gaussian Processes (LAGP), the basic component of my model, is provided in Section 2.4.

2.1 GP Basics

Gaussian Process models are a popular method for nonparametric modeling of non-linear functions as are common in surrogate modeling of computer experiments [e.g., 38, 39], modeling of spatially referenced data [9], and machine learning [37]. A Gaussian process is a type of linear model (LM), but unlike standard LMs, GPs have the flexibility to model a variety of functional forms without a basis function representation.

At its simplest, responses from a GP can be thought of as a joint draw from a normal distribution with some mean function, $\mu(f(x))$, and positive definite covariance function, $\Sigma(f(x))$, both defined over the input space. The mean function can be arbitrarily treated to be 0. The covariance of the responses, defined by a function on the input space, defines how the input variables are related to the output. With a smoothly covariance function, a

realization of a GP can be equivalent to a linear regression model with an infinite number of basis functions [37].

While conceptually, a GP is a function evaluation over an entire space of inputs, in practice we estimate the model over an observed set of input points, X.

$$f(X) \backsim \mathcal{N}\left(\mu(X), K(X,X)\right)$$

where $\mu(X)$ is some mean process, and $K(X,X)$ is the observed covariance function on X. The covariance structure defines the error over the input space, so we can arbitrarily treat the function as a mean zero normal distribution with covariance defined on the observed input space. We can equivalently formulate the response as the sum of a mean and a structured covariance.

$$f(X) = \mu(X) + \eta(X)$$

In this equation $\eta(X)$ is a zero mean Gaussian random process defined by the covariance $K(X,X)$.

Treating the entire function as a correlated draw from a normal distribution, we can derive predictive equations for the function value at unobserved points by using the joint distribution between those points, and the observed data through the correlation, $K(X^\star, X)$. We can construct the joint distribution of the observed data, and the predictive points by partitioning a covariance matrix of all points as follows:

$$\begin{bmatrix} f(X^\star) \\ f(X) \end{bmatrix} \backsim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} K(X^\star, X^\star) & K(X^\star, X) \\ K(X, X^\star) & K(X, X) \end{bmatrix} \right)$$

Since we have observed $y = f(X)$, we can solve for the expected value and variance of

$f(X^\star)$. This gives the following estimation for the mean and variance of any unobserved data:

$$\mathrm{E}\left(f(X^\star)|X, y, X^\star\right) = K(X^\star, X)K(X, X)^{-1}y$$

$$\mathrm{Cov}\left(f(X^\star)\right) = K(X^\star, X^\star) - K(X^\star, X)K(X, X)^{-1}K(X, X^\star)$$

It is trivial to show this formula leads to a deterministic view of the observed data, since plugging in the training data to the prediction equations, i.e. $X^\star = X$, gives the observed responses, y, as the expected value, and a covariance of 0. Also note that as the correlation with the existing data approaches zero, the GP predictions become 0 with variance $K(X^\star, X^\star)$.

This is a frequentist view of estimation and prediction using a GP. A Bayesian formulation fo the same model can made by setting the prior knowledge to the set of all functions that can be generated using the function above, and allowing the data to refine our model to only those that match the observed data.

Conceptually this makes a GP easy to grasp, but the above specification applies only to deterministic models. To adjust our model for cases containing noise, we can add a new white noise process $\epsilon(X)$ to the function equation.

$$f(X) = \mu(X) + \eta(X) + \epsilon(X)$$

In practice we combine this with the covariance function from our Gaussian distribution by including a diagonal term. Note that with the addition of a diagonal term, the matrix is no longer constrained on $[-1, 1]$, and so is not a true covariance, although the common nomenclature for the result is still "covariance matrix." This leads to the following process

distribution.

$$f(X) \backsim \mathcal{N}\left(\mu(X), K(X,X) + \sigma^2 I\right)$$

Here I represents the $N \times N$ identity matrix. The end result of this is that two observations from the same point in the input space do not need to have equal response values. This can be shown by plugging in the inverse of the new covariance matrix in place of the deterministic one to the predictive equations above.

$$\mathrm{E}\left(f(X^\star)|X, y, X^\star, \sigma^2\right) = K(X^\star, X)\left(K(X,X) + \sigma^2 I\right)^{-1} y$$

$$\mathrm{Cov}\left(f(X^\star)\right) = K(X^\star, X^\star) - K(X^\star, X)\left(K(X,X)^{-1} + \sigma^2 I\right) K(X, X^\star)$$

With this adjustment, the predictions at the original data points are no longer the observed responses with no variance. An added benefit to including the diagonal term is increased computational stability. In practice, we include a small diagonal term, even in deterministic models, to make model estimation easier. Note that for both deterministic and noisy models, function value predictions are formed by a linear combination of the observed response values. A GP model with a correctly specified covariance structure is the BLUP for new function values. It is also important that the covariance for a GP is formed by subtracting a quadratic form from the prior covariance $K(X^\star, X^\star)$. This means that using a GP uniformly decreases the variance of function estimations at any new points.

It remains to choose an function for the covariance, $k(x, x^\star)$, from which we will form the covariance matrices. While there are many functions that satisfy the necessary features, creating a symmetric positive definite matrix, such as the Matérn covariance, I will use the

separable inverse exponential function, broadly defined as:

$$k(x, x^\star) = \tau^2 \exp\left\{ -\sum_{i=1}^{d} \frac{(x_i - x_i^\star)^2}{\theta_i} \right\}$$

where θ_i is a lengthscale parameter that determines the effective distance of relationships in dimension i. τ is a parameter that determines the amplitude of the resulting function. Once we have defined the covariance function for any pair of points, we can define the covariance matrix $K(X, X^\star) = [[k(x_j, x_l^\star)]] \ \forall \ x_j \in X, x_l^\star \in X^\star$. Another feature of this specific covariance function is that it yields a GP that is infinitely differentiable. This means GPs with the inverse exponential covariance function define a class of infinitely smooth functions. While it has been argued that infinitely smooth functions may not be versatile enough to model real data (which is frequently less smooth) [44], for many applications, such as the specific purview of my method, surrogate modeling, an infinitely differentiable model is desirable.

With the function defined, the hyperparameters that define the covariance, lengthscale, amplitude, and nugget, must be estimated. There are many methods used to obtain these values, including Maximum Likelihood Estimation (MLE), integration, cross validation, or fully Bayesian methods such as Markov Chain Monte Carlo. I will use MLE to estimate all hyperparameters $\Theta = (\theta, \tau, \sigma^2 = \tau * \eta)$. The likelihood to be maximized, as defined by the normal distribution, is

$$P(y|K_\Theta) = (2 * \pi)^{\frac{N}{2}} \, |K(X, X)|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2} y^T K(X, X)^{-1} y \right\}$$

Because the data have been observed, the only thing that can increase the likelihood is the correlation function defined by the parameters Θ. Various iterative algorithms, like Newton methods, can be used to accomplish this.

The result is a versatile model that is widely used in many fields: computer emulation for its ability to interpolate, geospatial statistics for its smoothness, and machine learning for its general flexibility. For an example of general GP performance, consider the following function, which is governed by a general sinusoidal pattern with an increasingly rapid offsetting smaller cosine oscillation, and a small amount of independent white noise.

$$f(X) = \sin(X) + \frac{2}{3}\cos(\log(X^X)) + \epsilon(X), \quad X \in [0, 3\pi]$$

$$\epsilon(X) \backsim N(0, \sigma^2 = .01)$$

The result is a function that would be difficult to recognize, or represent with a series of basis functions. To test GP flexibility, I sampled 200 X locations from $X \backsim \text{unif}(0, 3\pi)$, and generated $f(X)$ from this distribution. I then fit a boilerplate GP with MLE estimates for hyperparameters using the `newGP` and `jmleGP` functions from the `laGP` package in `R`. The resulting GP model is shown in figure 2.1 using a solid black line for the mean and dotted lines to represent the 95% prediction intervals. The true $f(X)$ is plotted as a blue line. The resulting predictions follow very closely with the true mean, and the predictive intervals have good coverage.

GPs are popular because the MVN structure can be extended to new predictive locations x, where closed form conditioning yields that $Y(x) \mid (Y_N, X_N)$ is also Gaussian. Given settings of hyperparameters, moments of that distribution are given as

$$\hat{\mu}(x) = k_N^\top(x) K_N^{-1} Y_N \qquad \text{and} \qquad \hat{\sigma}^2(x) = \tau^2(1 + \eta - k_N^\top(x) K_N^{-1} k_N(x)), \tag{2.1}$$

where K_N, $k_N(x)$ are the covariance matrix formed from the full input data, and the vector of correlations with the predictive location respectively. η is a so-called 'nugget'

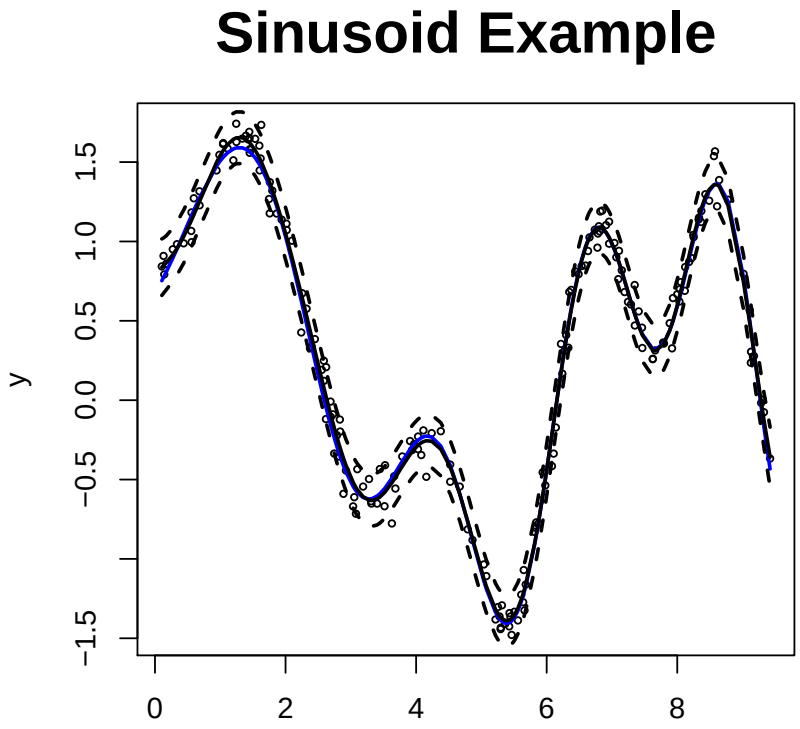

Figure 2.1: GP model fit to a sinusoid example. The true function is plotted in blue, while the GP fit is in black. The error bounds from the GP model are the dashed black lines. The model is able to capture both the mean and the variance of the true function well.

parameter. This formulation, which I will use for the rest of this thesis, is equivalent to the earlier model for predictions at a single point, where the hyperparameter $\sigma^2 = \tau^2\eta$.

2.2 Approximations

The obvious issue with GP models lies in the predictive equations. In order to obtain $(K(X,X) + \sigma^2 I)^{-1}$, you must invert an $N \times N$ matrix, with is $\mathcal{O}(N^3)$, and creates a matrix that is $\mathcal{O}(N^2)$ in terms of storage requirements. Additionally, obtaining actual predictions requires multiplying an $N \times 1$ vector by the $N \times N$ inverted covariance matrix, which is $\mathcal{O}(N^2)$. Obtaining the hyperparameters by maximum likelihood also requires the determinant of the

covariance matrix, which is $\mathcal{O}(N^3)$ as well. The computational burden is eased slightly by using the Cholesky decomposition, which returns the determinant and the inverse in one $\mathcal{O}(N^3)$ operation. Despite this, the computational requirements for GPs are too large to be handled for even moderately large data sets. The result is that although GPs are a flexible and provide good predictive qualities, the full GP model is frequently infeasible.

Because of ever increasing access to data leading to massive datasets that can be well modeled by GP methods, much research has been done to mitigate the computational cost. Many early methods focused on selecting a subset of the full training data that was computationally feasible to model. By selecting the best subset of size n from a full data set of size N, you invert a much smaller matrix, and reduce all computational requirements to acceptable limits. In an early attempt at this, [41], defines a quadratic algorithm to iteratively select the best new point to join a model that approximates the full GP. Additionally Smola and Bartlett define a stopping criteria based on the quality of the approximation, but note that even with the reduction to quadratic time, picking a preset n is more feasible in practice than evaluating the quality of the GP approximation after each point is added. [40] define an information loss criteria to sequentially select training points in $\mathcal{O}(1)$ time, for approximating the full GP.

The obvious detriment to any model built on a subset of the full dataset is scaling. You cannot guarantee a good approximation of the true function will be made from a small enough subset, and the method itself provides no reduction in computational time on the selected points. For sufficiently 'wiggly' functions, it is unlikely obtain a close approximation with a small enough subset to be computationally feasible. Learning the hyperparameters presents another issue, as these methods iteratively select locations to approximate the mean, and assume the covariance structure is already known. In approximating the mean, lowering σ^2 is paramount, and leads to relatively space filling designs. In practice, we learn the

covariance structure from data, and the lack of short distances eliminates the opportunity to learn about short lengthscales. The MLE for θ_i is unlikely to select a value shorter than the smallest pairwise distance between points in the selected subset. Space filling designs, like those produced by sparse GP's have a tendency to overestimate the lengthscale parameters, leading to a smoothing of the global function [50]. In practice this can lead to a poor estimation of the approximation quality, as evaluating how close a subset model matches the global model relies on the estimated covariance function.

To remedy issues with subset modeling, specifically to correct hyperparameter misspecification and to allow for use of all the input data, research moved into sparsity within the covariance structure itself rather than the points used for fitting the model. Compactly supported variance functions [16] can be used to introduce a large portion of zeroes into the covariance matrix, allowing for much faster inversion. Effectively this lets the model contain information from all observed data points of a much larger dataset before computational infeasibility is reached. [15] apply this directly to GPs to observe the effect on the MSE of the resulting model. While they are able to show the tapered MSE is asymptotic to the MSE from the full, correctly specified GP with either increasing sample size, or increasing taper length, increases in either cut into the time advantage. Obviously an increase in sample size means a higher N, and increasing the taper length increases the number of non-zero entries in the resulting covariance matrix. Additionally this formulation produces worse results for irregularly spaced points than for grids which makes it less desirable for real data scenarios.

The dip in mean surface accuracy represents a loss of some of the extrapolatory qualities that make GPs attractive to many researchers. Despite this, Kaufman et al. [27] were able to show that even small tapers can accurately estimate the parameters of an isotropic covariance function. Accurate hyperparameter estimation represents a significant improvement over global subset models. In Kaufman et al. [26], they extend efficient emulation to include

anisotropic covariance functions. To do this, they define a taper length in each direction, τ_j, and require that $\sum_j \tau_j < C$, where C is a constant chosen by experimentation. The result is, in effect, a shrinkage prior over the set of τ_j. To remedy the poor extrapolation qualities of tapers they employ a general estimate of the global mean before using the tapered GP to estimate the variance structure. Tapered GPs still revert to the mean estimate much faster than the full GP by design, but this global mean structure helps the prediction surface revert back to something more sensible.

2.3 Divide and Conquer

Another attempt to use all the data is what I am calling "divide and conquer" schemes. These are attempts to use all the data to build multiple models that will be recombined into a global prediction surface. By splitting the data into disparate models total computational burden of model estimation and prediction is limited by the number of data points in each model, N_i, instead of N, the size of the data. Because $N_i \ll N$, the overall computation and storage are greatly reduced.

A good example of model averaging can be found in the Bayesian Committee machine (BCM) [47]. By splitting the data into M different subsets of approximately equal size, $\mathbf{X} = \{X_1, \ldots, X_M\}$, each with a unique model, the computational cost of matrix inversion from a global model is reduced from $\mathcal{O}(N^3)$, to inverting M matrices of size N/M: $\mathcal{O}\left(N^3/M^2\right)$. It is obvious that picking the number of models M so that it increases as a function of N produces a computational complexity and storage both increase linearly with N.

The important notes for me, from BCM, are the way they give weight the component models, and the assumptions they make as part of the framework. To start with the latter, the BCM assumes that each subset of the data can form a model that is approximately

independent of the data used by the rest of the models. Tresp notes that this is only a reasonable assumption conditioned on the full target function, f. With locally focused model averaging schemes, broadly what I will be covering, it is unreasonable to condition on the full target function. For that reason I will not assume independence.

Weights for the BCM are chosen proportional to $K_i\left(X_i, X^\star\right)^{-1}$, the inverse predictive covariance matrix for model i. If $X^\star$ is chosen to be a single point, this reduces to inverse precision weights for all models. I use this as a good starting point for choosing model weights when the assumption of model independence clearly does not hold. Some other attempts to form weights from recombined models seem particularly unsuited to my application. Rasmussen and Ghahramani [36] sample from a dirichlet gating network to assign weights to potentially infinite GPs. The reliance on MCMC is more time intensive than I want for very large data sets. Chen and Ren [4] use linear regression weights to determine the quality of the individual models as predictors, but this relies on each model being relatively trustworthy for all predictive locations.

Another popular computation reduction method is partitioning the input space, $\mathbf{D} = \{D_i, \ldots, D_M\}$, where $\mathcal{D}_k \cap \mathcal{D}_j = \emptyset$ for all $k \neq j$, and $\cup_{k=1}^{K} \mathcal{D}_k = \mathcal{D}$. After partitioning, separate GP models are built for each piece of the input space. By building smaller models in each partition space, the computational cost of estimating the model is reduced to $(O)(\sum_{i=1}^{M} n_i^3)$ where n_i is the number of points used to build the model in the i^{th} partition. This can be equivalently thought of as a GP with a block diagonal covariance matrix [10]. The prediction cost is further reduced by first checking into which partition query points fall, to $\mathcal{O}(n_i^2)$ for points that fall into the i^{th} partition. Another added advantage is that even while using a stationary GP within each partitioned space, a partitioned GP can account for large degrees on non-stationarity in the true surface, including complete discontinuity.

The need for some level of non-stationarity can be observed in Figure 2.2. In it we see

the motorcycle data, which tracks helmet acceleration through time. The actual data points, represented by the black dots, appear to show at least three distinct variance paradigms. Fitting a stationary GP to the entire data set (left in Figure 2.2) results in badly overestimated variance on [0,14), and possibly overestimated variance on [40, 58], although the mean surface seems close to correct in most places. The right panel shows a model with partitions [0,14), [14,40) and [40,58]. The variance matches much better in this plot, and even though the partitions have introduced discontinuity at the boundary, the mean still matches fairly well.

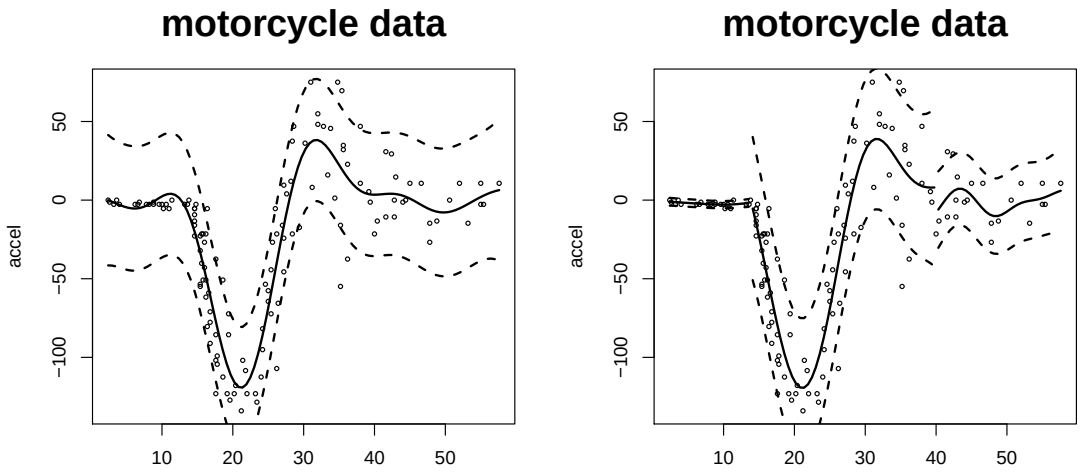

Figure 2.2: Left: A stationary GP fit to the motorcycle data. Even though the mean prediction is close, the variance clearly shifts paradigms, and is not captured well by the model. Right: a partitioned model (splits at 14 and 40) introduced discontinuity, but overall fits the data better.

Kim et al. [28] point out that the big questions for a partitioned GP are how and where to partition the data. Their answer to the first question is to use Voronoi tessellations [22]. They put a random distribution on the number and location of 'center' points C, and split the input domain, $\mathcal{D}$ into partitions, such that $\mathcal{D}_i$ contains all the points in $\mathcal{D}$ that are closest to C_i. They propose an MCMC algorithm that uniformly explores space of possible centers by selecting from the set of observed locations. Interestingly, the number and location of

partitions is decided by the algorithm, although this means that a computationally efficient solution may not be achieved if the number of partitions based on the likelihood is low. Another note, when using Voronoi tessellations, is that rather than an explicit function defining the boundaries of each partition, predictive locations fall into the tessellation with the closest center. A slight move of one 'center' will change the boundaries of all neighboring regions somewhat. In practice this means that determining which model should be used for each predictive calculation requires the distance to the 'center' of each partition.

There are many methods that employ GP partitions to achieve computational efficiency while allowing for some degree of non-stationarity. Each of these methods shares to some degree an issue with discontinuity at the boundaries of the partitions, owing to the independent treatment of data within the partition [43]. Essentially, for a set of points lying on the boundary between two partitions, $\Gamma_{ii'}$, there is no explicit requirement that the mean and variance surfaces are the same for models built on partition D_i and $D_{i'}$ [35].

$$k_i(x^\star, X_i)\left(K_i(X_i, X_i) + \sigma_i^2 I\right)^{-1} y_i \neq k_{i'}(x^\star, X_{i'})\left(K_{i'}(X_{i'}, X_{i'}) + \sigma_{i'}^2 I\right)^{-1} y_{i'},\ x^\star \in \Gamma_{ii'}$$

Because the covariance function, K_i, the variance estimate, σ_i^2, and the data used, X_i, can be different in each partition, neither the mean surface (shown above), nor the variance surface will match up perfectly along Γ. In some cases, this discontinuity is desired, but leads to worse estimation along the boundaries of partitions in cases where the degree of non-stationarity is relatively low. For an example, refer back to the left panel of figure 2.2. The first partition at 14 effectively models an actual sharp drop in the true helmet acceleration, and the predictive surface does not appear to suffer. The second split, at 40, is necessary to accurately capture the variance of the two regions, but results in much worse mean estimation near the boundary.

In some cases this may not be much of an issue, because predictive accuracy can be mostly maintained while reducing computational complexity. Partitioning purely for computational efficiency can be a problem in cases where either M, the number of partitions, or d, the dimensionality of the input space, are high owing increased amount of the interior space that will be taken up by the boundaries between partitions. This can represent a non-trivial degradation in prediction quality, especially close to the boundaries [34].

Various methods have been adopted to maintain computational efficiency of partitions while alleviating the degree of discontinuity in cases where the true surface is smooth. Gramacy et al. [21] employ regression trees [3] to recursively make splits. Gramacy uses the same Bayesian tree exploration as Bayesian CART and BART [5, 6], but builds GP models in the splits. Because this scheme employs a full MCMC, the posterior tree draws can be averaged over to lessen the degree of discontinuity between splits in regions where the non-stationarity is smoothly transitioning [19]. Another advantage over other GP partition models is that by controlling the priors on tree depth, you can force the model to make more splits than necessary to capture the nonstationarity, which will alleviate some of the computational burden of fitting the full GP even for relatively stationary responses.

The use of MCMC to integrate out some of the discontinuity associated with boundaries does increase the computational burden, leading to infeasibility for very large data sets [35]. Additionally, while averaging can decrease the degree of discontinuity, it does not eliminate it entirely. In their Domain Decomposition Method, Park et. al reframe GP as an optimization problem. This allows them to use a constrained optimization at a number of reference points along the boundaries between zones to enforce continuity in the mean surface. In one dimension, this equates to continuity along the whole surface, but in higher dimensions, there is only continuity at the boundary locations used for optimization. In Park and Huang [34], this idea is extended to the mean function along the entire set of pairwise boundaries by

estimating a boundary function using a finite mesh, and enforcing the continuity along the entire estimated function. In a further elaboration of the same concept of forced boundary continuity, Patchwork Kriging [33] defines the difference between GP models as an auxiliary process. They give this process a mean and variance of zero to ensure the fit must have no difference at the boundary points. In practice a small number stitching of points are used, since there are infinitely many points in each boundary. By representing the partition structure as a block diagonal matrix, and augmenting with the sparse correlation structure of the auxiliary process, continuity can be forced at the these points in both the mean and the variance surface.

It has been noted that for prediction, the true distribution at a point can be well approximated with fewer than the full data [49]. From this we can derive a predictive process: methods that seek to reduce computational burden while maintaining accuracy by building small GP models for every prediction. In addition to reducing the computational burden to $\mathcal{O}(n_m^3)$, and the prediction requirement to $\mathcal{O}(n_m^2)$, where n_m is some predetermined predictive model size, these methods require no storage beyond the original data. Predictive processes can also be viewed as a logical extension of the active subset models; seeking an active subset to build a local model rather than a global predictor. Finally predictive processes fit into the framework as the most extreme version of a partition model. Instead of partitioning the space into a finite number of subsets, they instead partition the input domain into a collection of points, each with its own model. As a partition model, these processes will suffer from a high degree of discontinuity although the high fidelity of a model produced for every point will mitigate most drastic jumps.

The simplest version of this is a Nearest Neighbor Gaussian Process (NNGP) [11]. For each desired predictive location, a small set of the nearest points in X in Euclidean space are used to form an approximate density. A specific application of this concept is the Dynamic

Nearest Neightbor Gaussian Process (DNNGP) [12]. Instead of the broad definition found in the later paper, Datta et al. define a nearest neighbor process for spatial functions with a time domain. For each of the query locations, a small set of points, from the immediate past in the time domain, that are nearest in the correlation space are used to make predictions. Implementation of a true DNNGP depends on prior knowledge of the correlation structure, which is not always available. In practice, Datta et al. suggest finding a 'eligible sets' using lags of different sizes in each input direction, and a maximum number of nearest neighbor (NN) points.

Approximate restricted likelihood [45] gives proofs for many of the core tenants of predictive processes. First, in a series of proofs they show that for accurate estimates of the mean, choosing only the points that are closest to the desired predictive locations can produce worse results. Then they prove that nearby points can be much better for accurately determining he hyperparameters of a GP than a spaced out set of points. The correct selection of points, therefore, should include both a set of nearest neighbors, and a set of distant points. This is an idea echoed by Zhang et al. [50], which notes that space filling designs (i.e. maximum coverage) do a poor job of estimating the length scale parameters. Stein et al. use this to make predictive designs as a combination of n_m nearest neighbors and $n_{m'}$ points selected by ranked distances. In this way they combine accurate length scale estimates with good estimation properties.

There are several side effects of the specific choices made by Stein et al. The first is that this predictive process can easily be made online. After each prediction and the observation of the associated response, that point can be added to the data set, and used for future predictions at minimal computational cost. This feature will be shared by all predictive processes. The second effect, which is unique to this formulation, is that the farthest point from the predictive location is always selected as part of the design, regardless of changes to

any of the other tuning parameters. For very large data sets, it is detrimental to, by design, always include the point that is farthest from the $x^\star$.

Fuentes [14] lays out a neat framework for combining both averaging schemes and partition schemes that I will be adopting in this paper. Assuming we want to model some global process, $f(x)$, and we have built a number of models, k, the combined predictor should be as follows:

$$f(x) = \sum_{i=1}^{k} f_i(x) w_i(x)$$

While they define $w_i(x)$ to be an arbitrary positive kernel function, I will extend this to apply to all weighting functions, i.e. $w_i(x)$ s.t. $\sum_i w_i(x) = 1$. This new formulation encompasses both model averaging schemes, which generally give some positive weight to all models, and partition models which choose exactly one $w_i(x) = 1$. I will also take a closer look at the associated variance function. Both model averaging methods and partition schemes avoid the issue of covariance between models. The former bypasses it by assuming independence, and the latter by giving only one model weight, thus nullifying the effect of covariance. My model will not be able to ignore the pairwise covariance.

These methods are not the only ways to attack the large data problem using GPs, but encompass the general logical progression of methodology leading to the novel components of my proposed method. There are several 'bake off' papers that give a general overview of GP methodologies not covered here [2, 23]. Specifically for this paper, Heaton et al. [23] provides an overview of many of the methods to which I compare mine, as well as the real data example I use. What follows is a brief overview of my direct competitors that are not already covered above.

2.4 LaGP

I also use a predictive process as the core of my method. In the paper that will be used as the basic building block for my model [20], the choice of local design it made by a more results-oriented approach. After choosing a small set of NN points, Local Approximate Gaussian Processes (LAGP) greedily adds points either to minimize Mean Squared Prediction Error (MSPE), or maximize Active Learning Cohn (ALC) [8]. Obviously minimizing MSPE will seek to lower both the predictive bias and the variance simultaneously. Maximizing ALC is equivalent to minimizing the predictive variance alone. Gramacy and Apley note that both methods will have significant overlap, although the final design choice will be different. In some examples, the two different metrics may not provide substantially different results, but the ALC criteria is 2-3 times quicker to evaluate, so it is the method I use.

Figure 2.3 shows a comparison of the subset selection between NNGP and LaGP using ALC as the criteria. The NNGP points, shown in red, are exclusively clumped around the predictive location, while the LaGP points, in black, have a number of NN points, but also include several rays of points further away. Despite being further from the prediction, these points lower the predictive variance more than the closer observations.

Require: input set X, $x^\star$ predictive location, observed responses y, and s nearest neighbor points, X_s, final model size f

1: Calculate GP(X_s) {Calculate initial Nearest Neighbors GP}
2: Calculate $V_s(X_s)$, θ_j {Calculate initial NN variance and hyperparameters}
3: **for** j in s to f **do**
4: Find $\underset{x_{j+1}}{\text{argmax}} \Delta V_j(x_{j+1}|\theta_j)$ {Find the point that maximizes the variance change}
5: Calculate next $V_{j+1}(X_{j+1})$, θ_{j+1} {Calculate new GP variance and hyperparameters}
6: **end for**
7: Output LaGP(X_f)

Algorithm 1: Basic steps for finding an laGP design using ALC

The basic idea of the ALC criteria is to maximize the change in the predictive variance

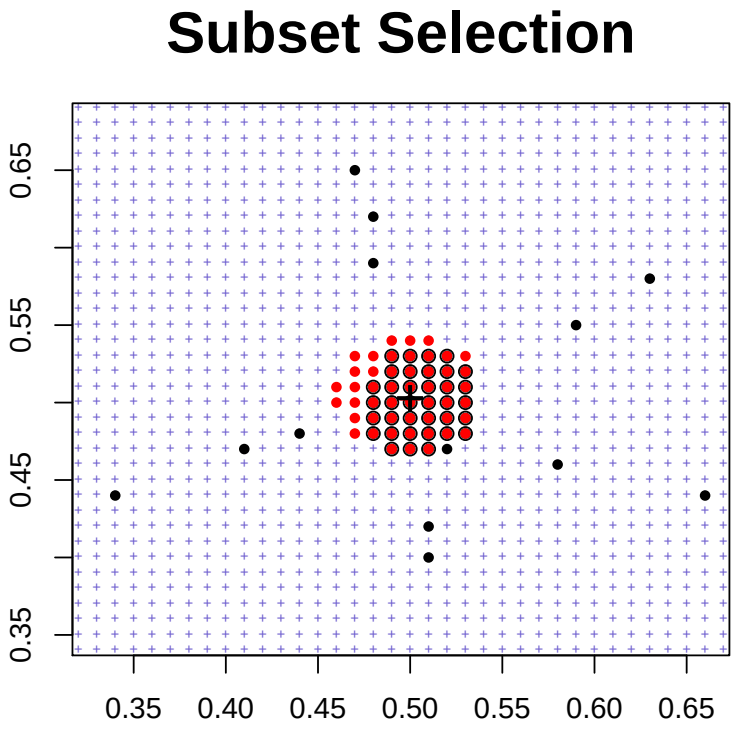

Figure 2.3: A Comparison of the subset selection for NNGP and LaGP using ALC. The observed data locations are marked by the blue crosses, while the query location is shown as a black cross. The NN points are marked in red. The LaGP subset is shown in black.

when adding a new point $\Delta V_j(x_{j+1}|\theta)$, where V_j is the variance of the current design at the reference set (typically chosen to be one point), x_{j+1} is the new point to be added, and θ are the currently estimated hyperparameters. It has been shown that this approximates maximum information designs [8]. Because the predictive variance cannot go up when adding a new point to a design, maximizing the change in variance is equivalent to minimizing the variance for each point added to the design. The sequential selection of points is summed up in Algorithm 1.

Despite the search algorithm being greedy, the resulting model is a good approximation of the minimum variance prediction at the reference point, $x^\star$. This can provide a reasonably accurate model for every point in the input space with only a small matrix inversion, however, because it is a partition model, laGP is pathologically discontinuous. Figure 2.5 illustrates

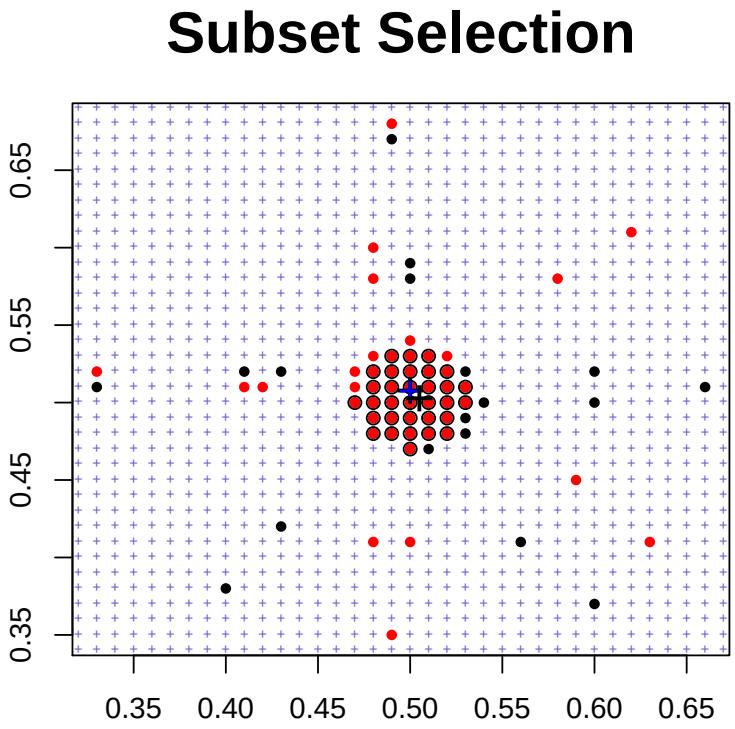

Figure 2.4: Subset selection for LaGP using ALC from two very close locations. Small crosses represent observed data locations. Red points represent the subset of points chosen for the GP centered at the blue cross. Black points are the subset for the model centered at the black cross. Red points with a black outline are chosen for both subsets.

the issue at the core of the laGP predictive surface. I have plotted the 2D design locations of two laGP predictors on the same set of data. The observed data is denoted by the small crosses. The predictive locations for the model centered at the black cross are shown in black, whereas for visibility's sake the red dots represent the design for the model centered at the blue cross. Red dots with a black outline are present in both models. Even though the two locations where predictions are desired are closer than any two points in the observed data, there are noticeable differences in the design. The Nearest neighbor set clump around the point is only slightly different, but the two models share no radial points in common. The result of two models using different subsets of points is a discontinuity, the severity of which is determined by the response values of the included points.

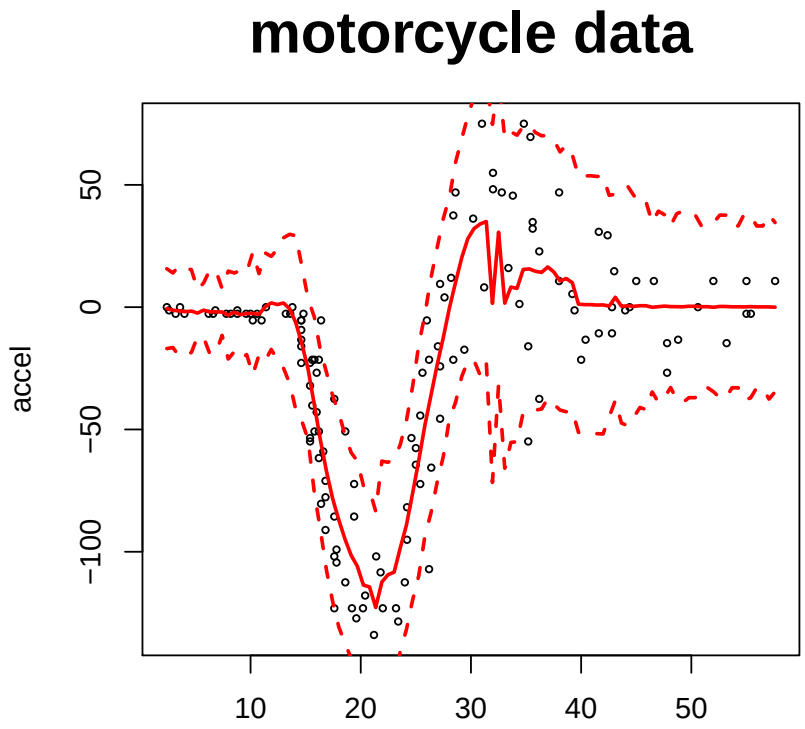

Figure 2.5: LaGP predictions for the motorcycle data. Ability to capture non-stationarity results in both improved predictive accuracy, especially the variance, as well as obvious discontinuity.

Figure 2.5 shows the effect this discontinuity can have on an actual predictive surface. The red line represents the predictive surface at 100 equally spaced points across the input domain, while the dashed line shows the 95% prediction intervals. Notably the variance represents the true process much better than the full GP, but there are noticeable discontinuities especially in the areas where the variance is large. While, on the whole, the discontinuity is less extreme than the partition model, I would like a predictive surface that is actually smooth. Precision Aggregated Local Models (PALM) addresses two major issues with the laGP model. First, PALM provides a surface that is absolutely continuous in both mean and variance over the entire input space in any number of input dimensions. Secondly, PALM achieves a significant computational reduction by creating a global model that can be reused instead of searching for a new subset at each desired query location.

Chapter 3

PALM Methods

In this section, I cover the basic innovations of the Precision Aggregated Local Models framework. Section 3.1 provides a formal unification of partition and aggregation models into which PALM itself falls. A general introduction to weighting predictions from multiple models is provided in Section 3.2. In Section 3.3 I extend the concepts of LAGP to create a global model for deterministic data. Sections 3.4 and 3.5 cover weight selection for model recombination and pairwise model covariance respectively, both agnostic to the choice of a GP as a local model. GP specific implementation details for the PALM framework are provided in Section 3.6. Comparisons of PALM performance versus other state of the art models are provided in the following sections, on a simulated example in Section 3.7, and on a real dataset used for a competition in 3.8. Finally general notes on the methodology, including possible extensions, are given in Section 3.9.

3.1 Unifying Partition and Aggregation

To develop the formal framework into which my model fits, consider partition models. Another way to see this – one which suits my proposed methodology and developments coming shortly – is to represent the partitioned predictor as a weighted average. Suppose we wish to predict at testing site x. Let $w_k(x) = \delta(x \in \mathcal{D}_k)$ indicate whether or not x is in partition element $\mathcal{D}_k$. Then the partitioned predictor, using GP predictive notation (2.1) applied

separately for each index k via training data $(X_k, Y_k) \in \mathcal{D}_k$, may be represented as

$$\hat{\mu}(x) = \sum_{k=1}^{K} \hat{\mu}_k(x) w_k(x) \qquad \text{and} \qquad \hat{\sigma}^2(x) = \sum_{k=1}^{K} \sum_{j=1}^{K} \hat{\sigma}_{kj}(x) w_k(x) w_j(x) \tag{3.1}$$

Above, $\hat{\sigma}_{kj}(x)$ represents the covariance between the predictors in partitions k and j, and $\hat{\sigma}_{kk}(x) \equiv \hat{\sigma}_k^2(x)$. Since the $\mathcal{D}_k$ are mutually exclusive only one component of the sums (3.1), receive any weight. In this way partition scheme avoid the need to estimate the covariance between component models. Each set of predictive equations is itself Gaussian, arising from a GP for $k = 1, \ldots, K$, so the weighted sum is also Gaussian and thus a GP. By this formulation, partition models fit neatly into already existing model averaging schemes because $\sum_{k=1}^{K} w_k(x) = 1$. My method, PALM, also fits neatly into this framework.

To illustrate partitioning on a concrete example I will be returning to throughout this thesis, consider *Herbie's tooth* [29]. For scalar inputs z, define

$$g(z) = \exp\left(-(z-1)^2\right) + \exp\left(-0.8(z+1)^2\right) - 0.05\sin\left(8(z+0.1)\right). \tag{3.2}$$

For inputs x with m coordinates $x_1, \ldots, x_d$, the response is $f(x) = -\prod_{j=1}^{d} g(x_j)$. Here the default $d = 2$ case is chosen for ease of visualization. Consider a training set composed of a 100×100 uniform grid in $[-2, 2]^2$ and commensurately sized (101×101), but slightly shifted out-of-sample testing set. In Figure 3.1, a regular $K = 100$-element partition is created, and GPs fit independently fit to the training data residing in each element of the partition. Full GPs are all but intractable on data of this size, requiring hours to fit on an ordinary workstation[1]. Through divide-and-conquer, fitting each 100-run element in serial takes around two minutes total, and potentially much faster in parallel.

From the figure, a downside to the partition model is readily evident: lack of continuity at

[1]... using ordinary linear algebra packages

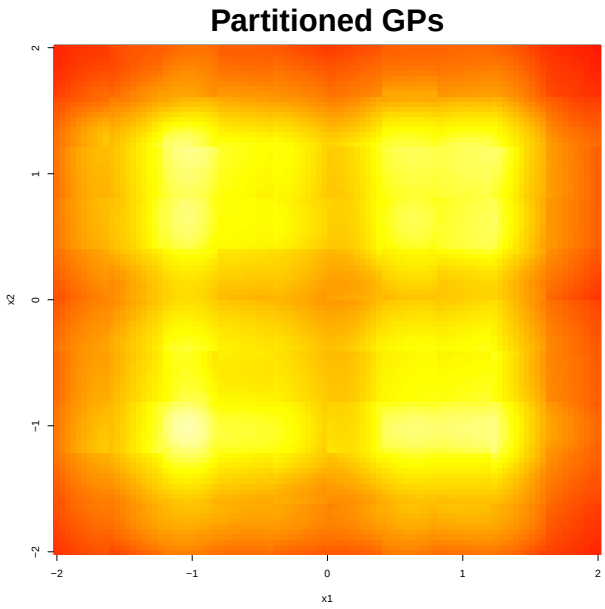

Figure 3.1: Predictive mean obtained from uniform-partition GP fit to Herbie's tooth data.

boundaries, which at times can be extreme. The discussed attempts to remedy these are not without their limitations. Some seem limited to, or have only been successfully illustrated in, two-dimensional input spaces, and can only guarantee continuity in the mean surface, while variances remains discontinuous [34, 35]. *Patchwork kriging* [33] can guarantee point-wise continuity at reference locations in both surfaces, but is discontinuous along the full manifold of partition boundaries. As the number of continuity–inducing reference points increases, it will converge to a fully continuous model. Perhaps the biggest downside of approaches attempting to "stitch" boundaries together is that it is not immediately obvious whether resulting predictor is also a GP.

The main component of my PALM model, LaGP, shares these same features. A form of "infinite partitioning" is happening implicitly in the input space $\mathcal{D}$ space from the point of view of the testing set, which makes LAGP an example of *transductive learning* [48], as opposed to the usual *inductive* sort. It fits into the form of Eq. (3.1) when enumerating all K subsets N which are of size n and giving only positive weight to those which are "near"

to x.

Despite many advantages, LAGP does have drawbacks – many of which may be evident from the narrative above. First, it does not create any permanent model. Each time a new prediction is desired, a new fit must be calculated. This leaves room for similar methods with a permanent model to drastically undercut, trading off space (to store the permanent model) for time. It does not avoid the continuity concerns experienced by other partitioning based schemes. Instead, they are grossly exacerbated: discontinuities are everywhere, although their precise nature may sometimes be difficult to detect with the naked eye.

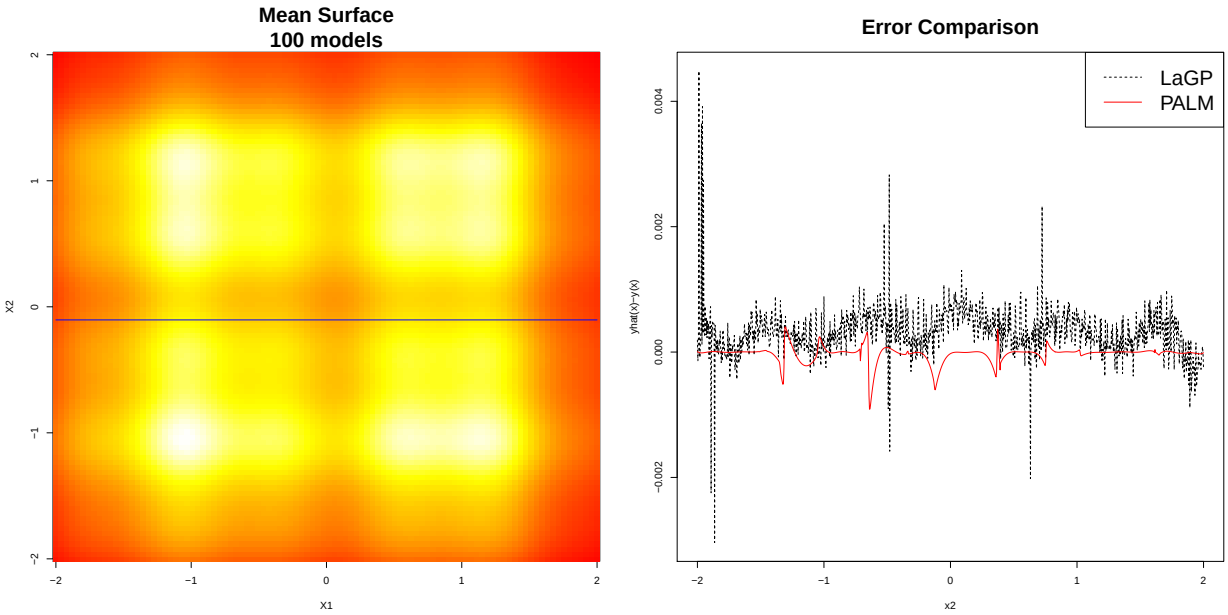

Figure 3.2: Herbie's tooth fits: LAGP on left; bias (right) in a slice through LAGP measured against the truth and a PALM competitor.

Figure 3.2 shows an LAGP fit on Herbie's tooth data. The left panel mirrors Figure 3.1, showing a predictive mean surface from LAGP using the defaults in the `laGP` package which uses a neighborhood size of $n = 50$. The right panel in the figure provides a higher-resolution view via a slice defined by fixing x_2 at -0.104, as indicated by the horizontal line on the left panel. Rather than showing predictive means on the right, bias $\hat{y}(x) - y(x)$ is plotted instead as a means of "zooming in". Focus for now on the dashed gray line, corresponding

to LAGP. Notice that LAGP's relationship to the true response, which is itself smooth, is pathalogically nonsmooth. Although the overall magnitude of the bias is quite small, at worst 0.006 in absolute value, there are isolated exceedingly poor predictions. Although many biases are much smaller than that, being less than 0.001 in absolute value, there is a systematic (almost sinusoidal) patten in these out-of-sample residuals. The red curve, which corresponds to my proposed PALM method introduced shortly, suggests that both issues – smoothness and predictibility left on the table – could be resolved with a modest amount of aggregating, inducing dependence between otherwise functionally independent local fits.

3.2 Averaging

So far I have only considered boolean weights in Eq. (3.1). What about more general $w(\cdot)$ satisfying $\sum_{k=1}^{K} w_k(x) = 1$? The result is a (true) model averaging predictor, representing a potentially smoother alternative to partition models. Such setups have been combined with GPs in order to combat the computational burden of fitting a single large GP [47]. Rather than partitioning, components being averaged can involve overlapping domains or random subsetting of the data, both with and without replacement. A common weighting scheme for these models is $w_k(x) \propto \phi_k(x)$, where $\phi_k(x) \equiv 1/\hat{\sigma}_k^2(x)$ is the predicted precision at x from model k. Such a choice is optimal if the predictors indexed by k are independent and unbiased [7]. The unbiased assumption is often violated, e.g., GPs which are biased toward the prior mean, but precision weighting is nontheless a default.

A bigger issue is independence. Because model averaging schemes abandon the use of indicator weights, and moreover may use overlapping data subsets, they must deal with multiple model covariance calculations $\sigma_{jk}(x)$ or risk inducing further bias into the meta-predictor (3.1). Model averaging measures rely on the fact that predictions from component

models are independent as long as they have learned the same function. Functional independence of global models can be expressed probabilistically as $P(X_i \mid X_{-i}, f) = P(X_i \mid f)$ where f is the global function. With this equality, we can use $\sigma_{jk}(x) = 0$ for all $i \neq j$, since, if they are learning the same function, the separate models are independent. Similar to the partition model, this reduces the variance calculation to $\hat{\sigma}^2(x) = \sum_{k=1}^{K} \hat{\sigma}_k^2(x)\, w_k^2(x)$.

Recombining component models into a unified prediction can work even if the data are not partitioned, as long as independence of the model, given a global function, can be maintained [[4]]. The computational complexity for fitting, and storage of the model are reduced like in a typical partition model, while prediction is K-fold more work because every predictor is involved for each predictive location x. A drawback, however, is that a common method for justifying functional independence is to have the training sets in each component being quite sparse relative to the global training data. This limits the weighted average's ability to accurately capture complicated trends, and results in over-smoothing. To illustrate, consider the left panel of Figure 3.3 which is based on precision averaged global models. The ten thousand point training set is randomly partitioned into 100 models containing 100 points each. Each model's prediction is weighted by a softmax on precision. Compared to either the left panel of Figure 3.2 via LAGP, or Figure 3.1 via partitioning, fidelity is clearly lost. Smoothly varying (e.g., precision) weights guarantee continuity of the surface, but emphasis on local "authority of influence" is starkly absent.

Therein lies the setup for our methodological contribution: PALM (precision aggregated local model). The idea behind PALM is to build locally focused models, as in a partition scheme or LAGP, but which can be smoothly recombined into a global model. The framework introduced momentarily could apply in principle with any (accurate) local model furnishing locally smooth predictive means and variances, but my presentation and empirical work favor LAGP. My goal is to retain the accuracy of an LAGP predictor, but lean on the

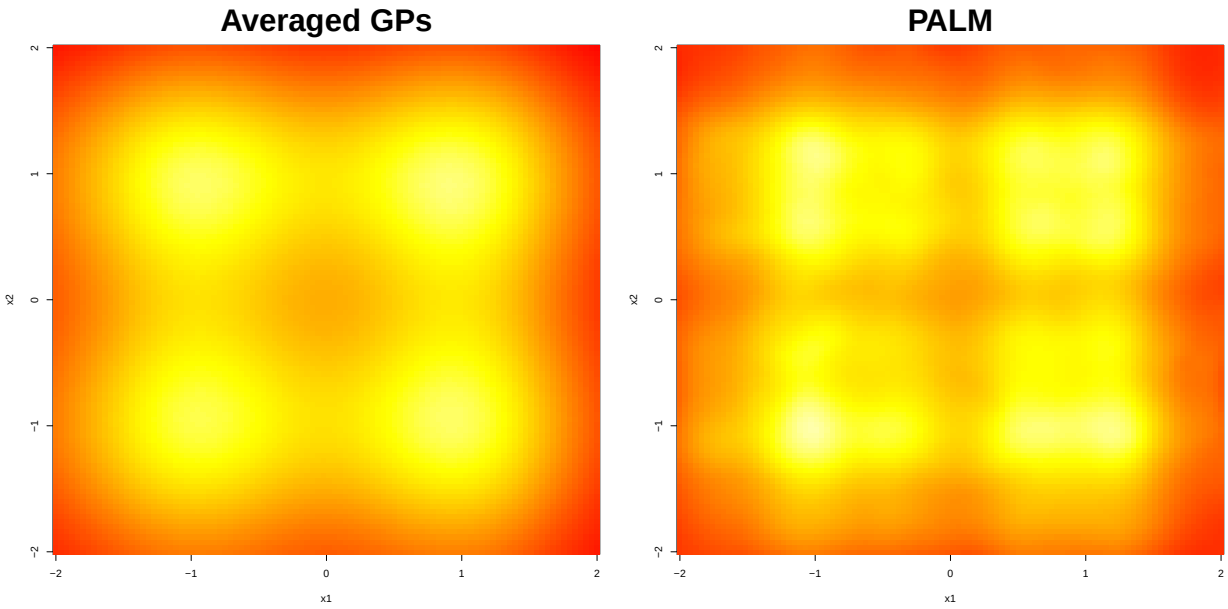

Figure 3.3: Model averaging (left) and PALM (right) predictions on Herbie's tooth.

PALM framework to obtain smoothness and computational gains. Appropriate weighting and accounting for covariance is highly component-model specific and I provide details for LAGP, however similar analogues will be readily apparent. The right panel of Figure 3.3 serves to whet the appetite. PALM offers a hybrid between predictors shown in earlier figures. Focusing on the slice in the right panel of Figure 3.2, observe that bias in these out-of-sample residuals is lower, smoother, and has less recognizable pattern than under LAGP.

3.3 Deterministic Evaluations

Local GP approximation, via LAGP, is designed to predict at a specific point x, but in fact that fit offers high quality predictions over a much broader area of the input space. To illustrate, consider the left panel Figure 3.4 which shows how an LAGP model can offer effective surrogate modeling (prediction and uncertainty quantification) of a sine wave

$x \in (3, 7)$ despite being trained only at the "center" value of $x_k = 5$. The objective function is comprised of 1000 equally spaced points from a sine wave on $[0, 20]$. The training subdesign is derived from default settings in the `laGP` package: starting with six nearest neighbor (NN) points, the ALC criteria is used to greedily select subsequent points before reaching a final size of 50. In the middle panel, the same idea is applied at four other centers spaced evenly throughout the input space. Finally, in the panel on the right, these $\mu_k(x)$ and $\sigma_k^2(x)$, for $k = 1, \ldots, K = 5$, are combined with inverse variance weights (3.1), details coming momentarily, to yield a global predictor which is quite accurate throughout the domain of interest despite a non-uniform density of sampling. The overlapping regions of these five local experts is enough to cover prediction for the entire region.

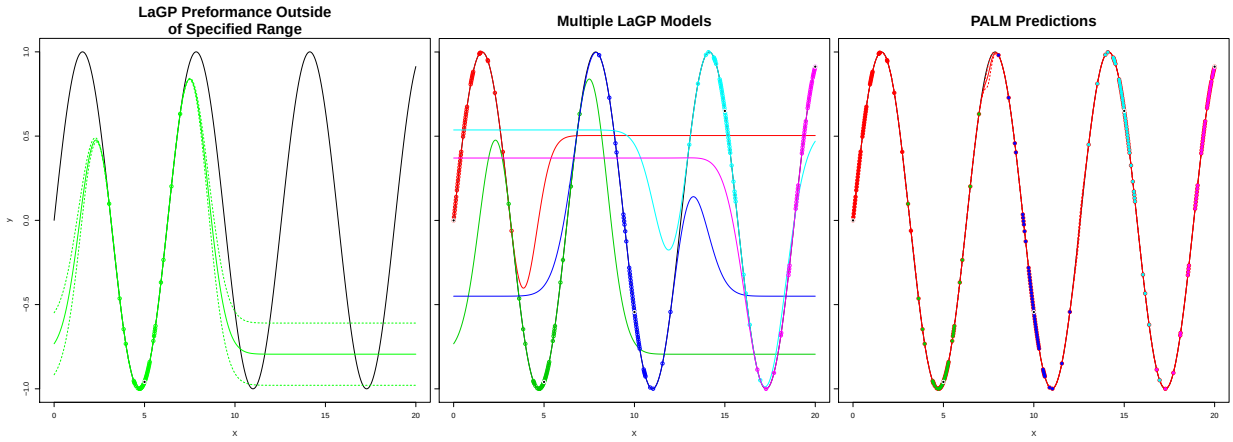

Figure 3.4: A sine wave modeled by LAGP predictors: alone at one location (left); at five locations (middle); after PALM weighted averaging (right).

A one dimensional example offers convenient visuals – overlapping predictive areas of multiple LAGP and their PALM combination together with an underlying truth – but the same principles hold true for higher dimensions, although visualization becomes somewhat more challenging. The right panel of Figure 3.3 shows a PALM fit to Herbie's tooth (3.2). The training set is based on a 5000 element regular grid, and the "centers" for 100 LAGP local experts (again via `laGP` defaults) were chosen with maximin design[2] [25]. Those local designs

[2]The maximin criteria was extended to include distance to the boundary in order to build a buffer to

are not shown in the figure, because there would have been too much clutter. However, some are shown later in Section 3.4 when discussing details of the PALM weighting scheme. The testing set was a regular grid of size ten thousand. Full LAGP prediction on those locations takes twelve minutes; our PALM analog takes six seconds.

The right panel of Figure 3.2 shows a slice through this surface, comparing error from LAGPs trained to every element of that slice of the predictive grid. Notice how the PALM predictor is both more accurate, and also exhibits a smoother bias/higher accuracy compared to the truth. I will present more examples throughout this thesis, but first it makes sense to deliver more detail on how PALM patches together a limited number of local experts to obtain smooth, more accurate prediction, than local experts trained exhaustively in the predictive grid.

3.4 Choosing Weights

Choosing a smooth weighting function that simultaneously selects and hybridizes between the right local experts, while maintaining continuity in both mean and variance surfaces, is key to the PALM framework. Inverse variance weighting $w_k(x) = \phi_k(x)/\sum_{\ell=1}^{K} \phi_\ell(x)$, introduced in Section 3.2, offers a good starting point, combining smoothness and localization according to the local experts' own judgment. Locally trained GP predictors, e.g., from LAGP, offer an organic and smooth reduction in variance nearby to the local design they are trained upon, and consequently will receive higher weight there. Conversely, higher variance away from their area of expertise will result in lower weight if another expert is commensurately better there. A downside, however, is that GP models are not unbiased – they revert to the prior process, and in particular the `laGP` implementation uses a prior mean of zero. Such

prevent centers of `laGP` models on the edges of the space, which would effectively clip their area of expertise.

bias, then, would become more severe as local GP experts extrapolate.

This means that theory of optimality of inverse-variance weights [7] does not apply, but coupled with that is perhaps a more practical issue: an inevitable compression of weight in regions of the input space where disparate local experts are equally good/bad. Intuitively, a weight $\phi_k(x)$ should increase nearby a simultaneously shrinking domain of expertise around x, i.e., have weight decreasing elsewhere. However, this cannot happen with a GP expert because predictive variance is also prior reverting. Consequently, adding local experts into PALM, no matter where their local domain of expertise lies, has a flattening or "cooling" effect on individual weights. To combat both issues (bias and cooling) we propose raising precisions to a power p before applying the softmax, i.e., $w_k(x) = \phi_k(x)^p / \sum_{\ell=1}^K \phi_\ell(x)^p$, where p is a function of the number of local experts, K, and the size of the input space, i.e., it's dimension d assuming inputs coded to $[0,1]^d$. In particular, we suggest $p = \log_d K$ as a sensible automatic choice, however another option is to treat p as a tuning parameter which could be optimized on a hold out set.

Figure 3.5 illustrates the effect of powering up weights. What we see in all panels are the predictive weights given to one LAGP expert's "center" (open circle) of the PALM used in Figure 3.3. The figure has been zoomed into the local expert, and the centers of other local experts used in the original model are represented by commensurately sized filled circles. Smaller dots nearby the open circle indicate the local design selected by LAGP for the open-circle expert. Red colors in the image plot(s) indicate low weight, and lighter/whiter colors indicate high weight. Now consider each panel individually. The top left panel corresponds to softmax precision weights, i.e., $p = 1$, nowpowering up. For contrast, the bottom left panel uses our recommended power instead, $p = \log_d K$. Observe how powering up has a "heating" effect, making weights more extreme on both ends, leading to fewer yellow/orange and more white/red. In particular, weight is greatly increased in the region nearby the central expert.

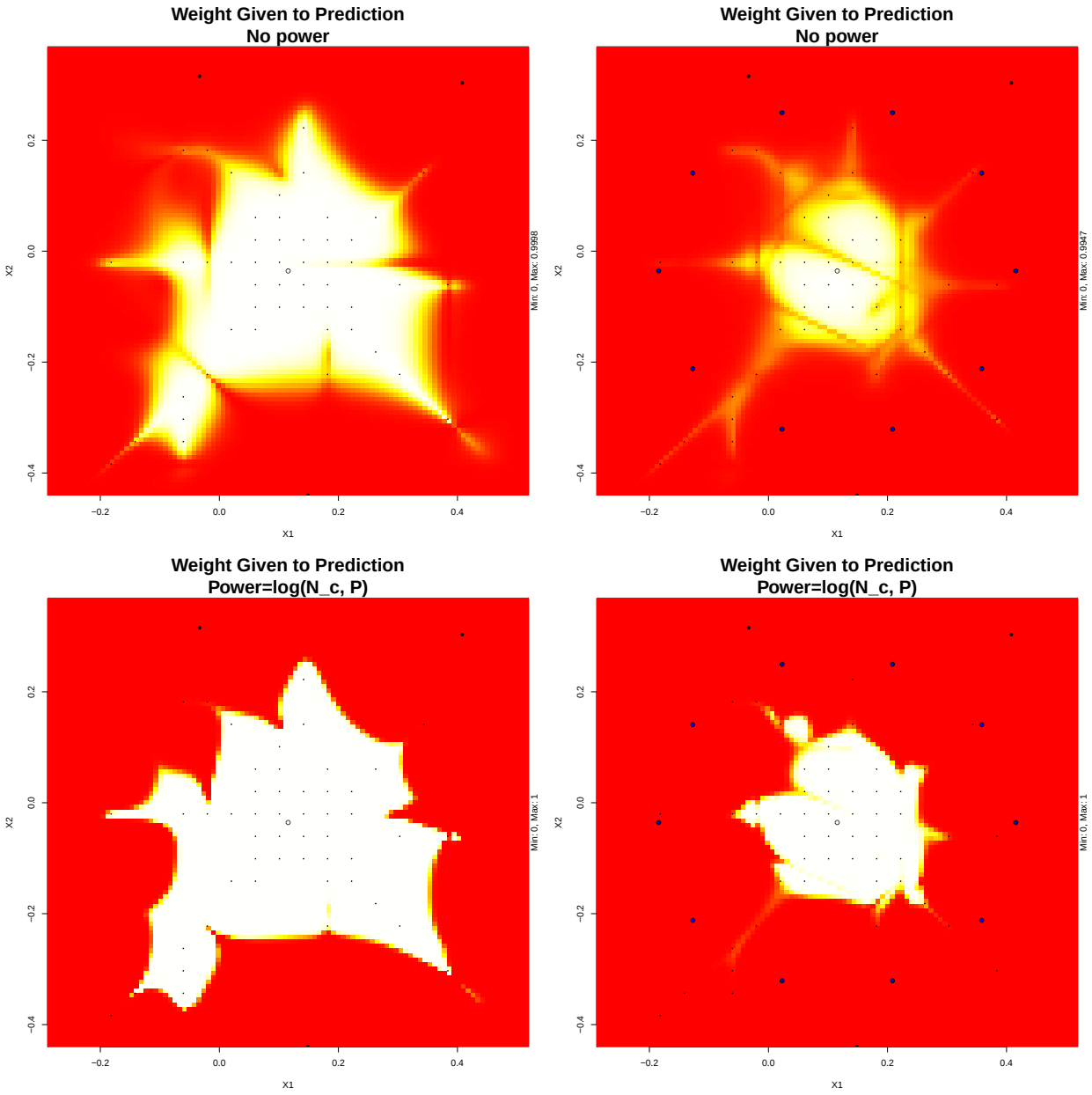

Figure 3.5: Top left: Weight given to a local expert with $p = 1$, i.e., un-powered. Top Right: Decrease in weight caused by adding extra models outside of the area of expertise. Bottom left: Local expert weights when powers are used. Bottom right: Added centers outside of the area of expertise do not affect the weight. Large black/blue dots represent other local experts in the PALM, while small dots represent the training inputs X_N.

Otherwise the same of the area with non-trivial (red) weight is largely unchanged.

Now consider the panels on the right in the figure which consider the effect of adding new

local experts into the PALM fit. The "centers" of those experts are indicated by larger blue-filled dots. The top right panel corresponds to no powering up ($p = 1$), and the bottom right one uses $p = \log_d K$. Otherwise everything is the same as on the left; weights correspond to the central expert indicated by the open circle, etc. In the top right, notice that even though the new experts are added largely outside of the original "center's" area of expertise, the weight for the central expert has gone down, in relative terms (i.e., after re-normalization) everywhere. The non-red area is smaller, and the proportion of yellow is much larger than in any other panel in the figure. Not only has weight gone down within that expert's domain of expertise but, intriguingly, the presence of the new experts have created streaks of low weight in regions that are very close to those new experts. (The new experts are somehow not experts nearby themselves, albeit in a small volume.) Likewise we can still see the central expert receiving sizable non-zero weights in the area of the new experts where it has not effectively learned the surface. This indicates a lack of ability for pure precision weights to accurately determine which of our experts should be trusted. In the bottom right panel, after appropriately powering up precisions, we see that the same ring of new experts has lowered the weight only inside their own areas of expertise, whereas weight given to the central model remains high in the densely packed region of design points. More experts mean more reactivity in domain of expertise across the input space.

Figure 3.6 shows the effect powered weights have on the resulting predictive surface is drastic. The left panel shows observed data generated on a 100×100 grid from Herbie's tooth with $\sigma = 0.05$. I fit a PALM with 100 space filling centers. In the center panel, I made predictions on a testing grid generated on a 101×101 grid using the same procedure, using $p = 1$ in the above weighting equations, i.e. unpowered weights. The result is what we should expect from a model that has difficulty identifying the best local expert for the problem. The surface is generally much flatter than the observed data. Predictions in the right panel are made from the exact same PALM using $p = \log(100, 2)$, our recommended

setting, and have actually captured the trend of the observed data. The result is a significant improvement in score, from around 3.72 to 4.75, which is caused by a nearly 50% drop in RMSE (0.98 to 0.56), and an equally drop in average standard deviation.

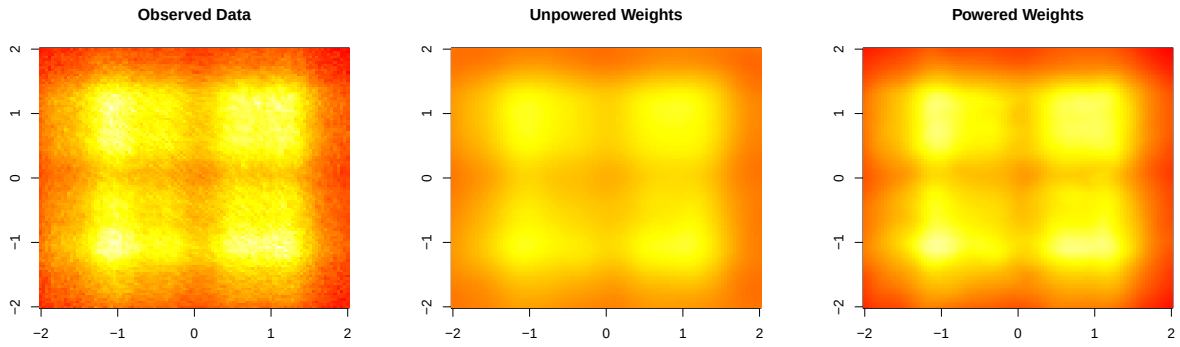

Figure 3.6: Left: Observed data from Herbie's Tooth. Center: Predictions from a palm using $p = 1$, i.e. pure precision weights. Right: Predictions from the same PALM using $p = \log(N_C, 2)$. Local model predictions are unchanged. Only the weight is different.

3.5 Estimating Covariance

A major difference between PALM and other model averaging schemes is that we cannot assume functional independence. Other schemes get away with approximate functional independence because each expert k is learning something broad, $\hat{f}_k$, which is a good approximation to the global function, f. Consequently, globally focused averaging schemes have a relatively constant covariance between models across the input space that authors claim to be low, and tacitly approximate as zero without noticeably ill effect. Locally focused models, on the other hand, can have an extremely high covariance in areas where their domains of expertise overlap, and low elsewhere. An illustration in the LAGP case is coming in Figure 3.8, a little later in Section 3.6. Because of this, we need a way to estimate the covariance between local experts. Assuming zero covariance will result in underestimating the uncertainty in our predictions following Eq. (3.1). A challenge in estimating those covariances

is that our local experts assume independence when performing local design and inference. Therefore observing co-variability must transpire as a post-processing step.

For an empirical illustration of the necessity of including covariance in the model, observe Figure 3.7. In the both plots, standard deviation is plotted for a 100 center PALM fit to Herbie's Tooth with $\sigma = 0.05$. The left plot considers covariance and the right plot assumes that model predictions can be combined under an assumption of independence. In both plots, the black crossed represent the centers of the local models used to build the PALM. Because the plots are constricted to the same color scheme, we can see that the standard deviation is, on average, much higher across the input space when co-variability is introduced into the model, and this is a good thing!

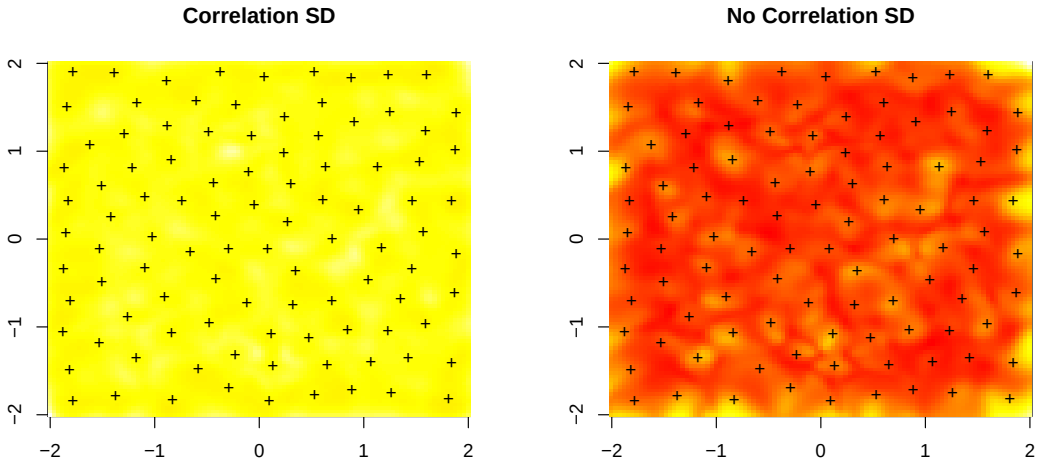

Figure 3.7: Left: Standard deviation surface from a model that considers pairwise correlation between local expert models. Right: Standard deviation from a PALM that treats local experts as independent. Black crosses denote the centers for local experts. Colors on both plots are restricted to be the same

The result of the independence assumption may be lower standard deviation, but it is important to note the location where the lowest values occur. The standard deviation achieves its minimum, not at the centers of the local models, but in the spaces between those experts. In effect this model is suggesting it has the most surety about predictions

in the spaces where the least data is used. This is because we are missing the piece of the variance equation 3.1 for every $k \neq j$, and the resulting sum is much lower than it should be. The minimum standard deviation for model wise predictions in the left panel is 0.02 which is much lower than the standard deviation for the actual generating process. In the left panel the variance is higher in the area between local experts, as we would expect, and all of the values are in $(0.48, 0.57)$. The result is a model that scores drastically better (4.75 compared to 2.63) despite the fact that the mean is unaltered between the two.

Toward estimating those covariances, we begin by decomposing covariance into the product of standard deviations and correlation: $\sigma_{kj}(x) = \sigma_k(x)\sigma_j(x)\rho_{kj}(x)$. The two standard deviations, $\sigma_k(x)$ and $\sigma_j(x)$ are furnished by the predictor from each local expert, so we only need a way to additionally estimate the correlation, ρ_{kj}. Pointwise, for each x and experts j and k, we desire an estimate of the correlation between every pair of models:

$$\rho_{kj}(x) = \mathbb{C}\text{orr}(\hat{\mu}_k(x), \hat{\mu}_j(x)) = \frac{\mathbb{E}\{(\hat{\mu}_k(x) - Y(x))(\hat{\mu}_j(x) - Y(x))\}}{\hat{\sigma}_k(x)\hat{\sigma}_j(x)}.$$

For many models, $\rho_{kj}(x)$ may be difficult to estimate, especially when "lazy evaluation" at x is required, i.e., when we don't yet know where we wish to predict at fitting time. One of the main aims in PALM front-load calculations at training and avoid such expensive calculation at predict time. One way is by estimating a single $\hat{\rho}_{kj}$ for all x, and for all $j, k = 1, \ldots, K$, imposing a form of stationarity in correlation over the testing set. Because correlation between models is not constant over the input space, and is only high in the areas where a pair of models have overlapping influence, the non-overlapping space on the set $X_{kj} = X_k \cup X_j$ will artificially drag correlation down. Combined with powered up weights, which helps to ensure that two models will only receive high weight in the area where their influence overlaps, this empirical estimation compounds into a poor estimation of the correlation in exactly the area where the value has the most impact on the model.

Again I delay a specific example until Section 3.6.

I find that a suitable stationary value to plug in for $\hat{\rho}_{kj}$ is the maximum value over the input space: $\hat{\rho}_{kj} = \max_x \rho_{kj}(x)$. This formulation, however, becomes increasingly hard to estimate in higher dimension. Moreover $\rho_{kj}(x)$ depends on knowledge of $Y(x)$, a chicken-or-egg-problem at the time of fitting. Thus the final layer of approximation is to replace continuous maximization $\max_x$ with discrete search over data X_{kj} used to train local models j, k. That is, given a class of models for which we can obtain a point-wise estimate of model correlation, $\hat{\rho}_{kj} = \mathbb{C}\text{orr}(\hat{\mu}_k(x), \hat{\mu}_j(x))$, we use:

$$\hat{\rho}_{kj} = \max_{x \in X_{kj}} \{\rho(\hat{\mu}_k(x), \hat{\mu}_j(x))\}.$$

Using this formulation, we obtain a high estimate of correlation if either model contains a point in the overlapping area, and a low estimate otherwise. Specific forms for $\rho(\hat{\mu}_k(x), \hat{\mu}_j(x))$ depend the local expert's predictive equations; details are given for LAGP in Section 3.6.

We recognize this is an approximation, but our weighting scheme ensures than any inaccuracies would have a minor impact on the uncertainty captured by the PALM predictor. In Eq. (3.1), observe that the impact on $\mathbb{V}\text{ar}(x)$, for a single pairing of $k \neq j$, follows

$$\begin{aligned} w_k(x) w_j(x) \hat{\sigma}_k(x) \hat{\sigma}_j(x) \hat{\rho}_{kj} &\propto \left(\hat{\sigma}_k^{-2}(x)\right)^p \left(\hat{\sigma}_j^{-2}(x)\right)^p \hat{\sigma}_k(x) \hat{\sigma}_j(x) \hat{\rho}_{kj} \\ &= \hat{\sigma}_k^{-2p+1}(x) \hat{\sigma}_j^{-2p+1}(x) \hat{\rho}_{kj}. \end{aligned}$$

If we pick a form for $\hat{\rho}_{kj}$ which is not too small, i.e., $\hat{\rho}_{kj} \geq \hat{\rho}_{kj}(x)$, we get sensible results from the perspective of conservative estimates of uncertainty. Examine the case where two local models have an overlapping area of extremely high correlation, i.e., $\max_x(\rho_{kj}(x)) \approx 1$, which is the worst case scenario for the disparity between our stationary estimate and reality. When we predict close to the area where $\rho_{kj}(x) = \max(\rho_{kj}(x))$, the true correlation will be

close to the maximum and the value we have chosen will benefit the model. As we move away from this area, $(\sigma_k \sigma_j)^{-2p+1}$ decreases faster than the disparity between that max and the true correlation. This squashes the piece of the model variance that is due to this covariance.

3.6 Implementation Details

The PALM introduction was intentionally generic, or agnostic to the choice of expert although LAGP featured as an exemplar. Fixating specifically on that choice, several details must be in place in order to fully operationalize the methodology. Fitting GP model, local or otherwise, requires selecting hyperparameter settings, such as lengthscales $\theta^k = (\theta_1^{(k)}, \ldots, \theta_d^{(k)})$ and scale τ_k^2. The `laGP` library returns fitted values for these, independently for each predictive location or "center" for PALM. These are designed for predicting at one site only, without knowledge of how PALM intends to utilize those predictors more widely. It may be sensible to revise those estimates in light of the global scope of the wider PALM predictor. Local variation in lengthscales θ_k is sensible, and I have found no need to make adjustments for the PALM setting, except to tailor `laGP`'s prior to discourage extremely large lengthscales. Assuming we have no knowledge of the global lengthscale, we can find a reasonable estimate by building several global GP models using small random subsets of the data, and selecting the largest maximum likelihood estimate from among them to be the maximum value that each `laGP` may choose. Local experts may then adjust downward as necessary.

Scale τ_k^2 requires a more careful treatment, however, since it determines the amplitude of variability in the predictor (2.1). When trained on a local design, $\hat{\tau}_k^2$ could only reflect the scale of the local training design. When applied globally through predictive equations, this could be a mismatch to a much broader global scale, and thus result in extreme inaccuracy

especially in terms of predictive variance. This is particularly important since predictive variance is used to determine the weight of local models upon recombination through the softmax. Therefore, instead of estimating a separate amplitude for every model, we prefer to leverage information from all local experts to enforce a single hyperparameter value. In other words, we are building in an assumption of global stationarity in scale. One option here which is attractive theoretically is to perform inference for a global τ^2 by maximum likelihood, where the likelihood is comprised of a product of K MVN densities whose means and covariances follow K applications of Eq. (2.1), one for each local expert. In practice, however, this is intractable for even modestly sized training sets N, owing to the requisite cubic decomposition of $N \times N$ matrices. Instead, we prefer the following idea.

Hyperparameter τ^2 doubles as amplitude and asymptotic variance $\tau^2(1+\eta)$ of a GP model, measured as the correlation to training data X_n decays to zero. For now, take nugget as jitter for interpolation of simulations observed without noise, $\eta = \varepsilon$, however details for the noisy case will be provided momentarily. Reverse engineering a bit then, consider measuring that asymptotic variance through the PALM predictor instead. Below s^2 represents the empirical variance of the response vector, y, via simple residual sum of squares around the average $\bar{y}$. The plan is to use that estimate to define what the asymptotic variance of the final PALM fit should be. Let quantity τ^2_{PALM} represent the effective amplitude parameter of the full PALM, while τ^2 is the amplitude I use to fit the local experts in order to achieve the correct extrapolation properties.

What I want is for the asymptotic amplitude of our PALM fit to be equal to the measured amplitude of the training data. In other words, I want $\tau^2_{\text{PALM}}(1+\eta) = s^2(1+\eta)$. In order to achieve this, begin with the formula for variance from the PALM fit (3.1)

$$\hat{\sigma}^2(x) = \sum_{k=1}^{N}\sum_{j=1}^{N} w_k(x) w_j(x) \hat{\rho}_{kj} \hat{\sigma}_k(x) \hat{\sigma}_j(x),$$

with covariance estimates (correlation and standard deviations) plugged in. It is important to remark that the weight function, $w_k(x)$ depends on the GP correlation function between the data used to train model k, and the predictive location x as $k_k(X_k, x)$ through the GP variance function, $\hat{\sigma}_k^2(x) = \tau^2(1 + \eta - k_k^\top(x)K_k^{-1}k_k(x))$. As predictive location x moves away from the corpus of data used to train PALM experts (a subset of the full training data), each of the GP correlation functions approach 0. This causes the weight each PALM "center" receives as they extrapolate to be approximately equal, i.e., uniform. Consequently, we may replace those weight functions in the double sum above with $1/N$ where N is the total number of experts in the full PALM.

$$\text{As} \quad \lim_{k(X_k,x)\to 0} w_k(X) = \frac{1}{N}. \quad \text{we have that} \quad \hat{\sigma}^2(x) \to \frac{1}{N^2}\sum_{k=1}^{N}\sum_{j=1}^{N}\rho_{kj}\hat{\sigma}_k(x)\hat{\sigma}_j(x).$$

We may further use the limit of the GP correlation function to remove $\hat{\sigma}_k(x)$ from the double sum. As each $k_k^\top(x)K_k^{-1}k_k(x)$ approaches 0, $\hat{\sigma}_k^2(x) \to \tau^2(1+\eta)$, simplifying to

$$\hat{\sigma}^2(x) \to \tau^2\,(1+\eta)\,\frac{1}{N^2}\sum_{k=1}^{N}\sum_{j=1}^{N}\rho_{kj}.$$

Finally, remembering that we want to pin this value to a reasonable amplitude based on the variance of the observed response vector, we can solve for τ^2, the amplitude that should be plugged into each individual local expert. Specifically,

$$s^2(1+\eta) = \tau^2\,(1+\eta)\,\frac{1}{N^2}\sum_{k=1}^{N}\sum_{j=1}^{N}\rho_{kj}, \qquad \text{giving} \quad \tau^2 = s^2\frac{N^2}{\sum\sum\rho_{kj}}.$$

The final ingredient above, circling back the generic discussion in Section 3.5, is estimating ρ_{kj}, the correlation between LAGP experts. Define the *predictive kernel* of each expert j to

be the matrix wise correlation between each point in the model, and predictive location x:

$$k_j(X_j, x)^\top K_j k_j\left(X_k, x\right). \tag{3.3}$$

For each pair of models kj, we may find the kernel when using one model to predict the other, and visa versa. A sensible estimate for between-expert empirical correlation is the maximum value found in the combined vector of predictive kernels. Let $\rho_{kj}^{\ell} = k_k\left(x_\ell\right) K_k^{-1} k_k\left(x_\ell\right)$, where ℓ indexes each $x_\ell \in X_j$. Then, take

$$\hat{\rho}_{kj} = \max\left(\rho_{kj}^1, \rho_{kj}^2, \ldots, \rho_{ij}^{|X_j|}, \rho_{jk}^1, \rho_{jk}^2, \ldots, \rho_{jk}^{|X_j|}\right). \tag{3.4}$$

This kernel based correlation estimate (3.3) is bounded on $[0, 1]$, and is highly positive for models that are close to each other, while approaching zero for those that are separated. To illustrate this, Figure 3.8 displays two local experts used to model the sine wave from Figure 3.4. Solid lines represent the GP correlation (3.3) between each model and every point along the wave. Dashed lines show the weight that each model receives in predicting along the function. I have put the points used to build local experts along the correlation line for the opposite model, and colored the maximum point (which is used as the value for $\hat{\rho}_{kj}$) in light blue. Notice the value we choose for $\hat{\rho}_{jk}$ following Eq. 3.4, is very close to 1, which is accurate for the area of weight overlap. This value may not be accurate for the entire input space, but one of the models receives nearly 0 weight for all egregious areas. Using estimates $\hat{\rho}_{kj}$ we calculate predictive variance at any new point using

$$\hat{\sigma}^2(x) = (\vec{w}(x) * \vec{\sigma}(x))^\top \hat{\rho} (\vec{w}(x) * \vec{\sigma}(x))$$

where $\vec{w}(x) = (w_1(x), w_2(x), \ldots, w_K(x))$, $\vec{\sigma}(x) = \{\hat{\sigma}_1(x), \hat{\sigma}_2(x), \ldots, \hat{\sigma}_K(x)\}$, "$*$" is a component wise, Hadamard product, and $\hat{\rho}$ is the matrix containing empirical correlation values.

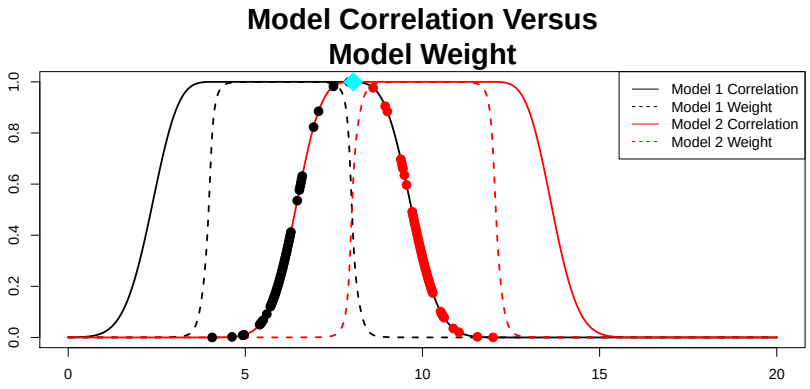

Figure 3.8: The correlation chosen between two local experts overlaid with the weight they receive in the model. The estimated correlation is shown as a blue diamond for visibility.

Regression for noisy data

The major difference between a deterministic GP model, and the stochastic version is the incorporation of a nontrivial nugget value, η, which breaks interpolation for smoothing, allowing for variance at training data points. The `laGP` function can return an estimated nugget for each local expert, obtained by maximum likelihood. This ignores that we intend to use the this local expert as part of a unified model. It's also overkill because separate nuggets mean heteroskeasticity [i.e., input-dependent noise 1, 18], and thus unnecessary estimation risk. I instead desire a pooling of estimated variance, toward a single nugget for the entire global surface, although return to the framework has the potential for greater flexibility.

One option is maximum likelihood, though the sum of normal log densities with mean and variance defined by (3.1). Unfortunately, we have not found this to be computationally tractable. Instead, we prefer a moment-based approach. Recognizing that the minimum possible variance is $\tau^2\eta$, I select η such that this minimum meets my estimate of the variance at a specific point. For each local expert, we find $\widehat{\text{mse}}_k = \frac{1}{|X_k|}\sum_{i=1}^{|X_k|}(\hat{y}_i^{(k)} - y_i^{(k)})^2$, where $\hat{y}_i^{(k)}$ and $y_i^{(k)}$ are fitted values from model k and observed training values, respectively. I then use the empirical $\widehat{\text{mse}} = \sum_k \widehat{\text{mse}}_k / N_k$ as my estimate of the minimum variance for a model,

$\hat{s}^2$. MSE is typically used as an estimate of variance plus bias squared, however, we expect each local expert to be unbiased on the data used to build it, despite the fact that we cannot assume they are globally unbiased. This leads to the selection of $\hat{\eta} = \hat{s}^2/\hat{\tau}^2$ as the unified nugget for all experts.

3.7 Illustration

To see the how powerful the PALM framework can be applied to common, difficult to predict functions, I benchmark it against LAGP on Herbie's tooth 3.2. First, to compare quality of predictions, I measure the out-of-sample root mean squared prediction error (RMSE). For N' testing data examples,

$$\text{RMSE} = \sqrt{\frac{1}{N'}\sum_{i=1}^{N'}(y_i - \mu(x_i))^2}.$$

Second, I consider the proper scoring rule introduced by Eq. (27) in [17], which is a reduced form of the scoring rule utilizing two moments – a special case of Eq. (25) from that same paper.

$$\text{score}\left(\mu(X), y\right) = -\frac{1}{N'}\sum_{i=1}^{N'}\frac{\left(y_i - \mu(x_i)\right)^2}{\sigma^2(x_i)} - \log(\sigma^2(x_i))$$

Notice how score offers good balance of predictive accuracy through $(y_i - \mu(x_i))^2$, and uncertainty quantification through $\sigma^2(x_i)$. I expect that whatever we might lose in predictive accuracy from LAGP can be made up by leveraging global information for proper variance estimates. Lastly, I note predictive time. This metric is the main catalyst for developing the PALM methodology, and I would expect a massive advantage over other methods with a similar predictive accuracy.

Table 3.1 summarizes the results of a simulation study comparing PALM and LAGP on

these important metrics, including time per prediction in seconds. I fit a PALM with 100 space-filling centers, each with the default `laGP` settings, to Herbie's Tooth (3.2) generated on a 100×100 grid in 2d. Training responses are sampled under a standard zero-mean Gaussian with sd $= 0.05$. The testing set is generated form the same function on a 101×101 grid, deliberately spaced to miss the training data locations. Observe that RMSE and score are practically identical between the two predictors. Even for a small PALM (only 100 local models, utilizing fewer than 5000 of the 10000 training sites), we have achieved comparable results to LAGP, which utilizes every training data location, while predicting in a thirtieth of the time.

	RMSE	Score	Time
LAGP	0.0515	4.9062	0.0628
PALM	0.0525	4.8887	0.0021

Table 3.1: Mean RMSE, score, and predictive time per point on Herbie's tooth. PALM had an average model build time of 93 seconds, for a 114 seconds to predict 9801 points.

If we increase the size of the PALM, we can match the performance of LAGP while maintaining predictive advantage. Figure 3.9 shows the predictive capability of PALMs of various sizes versus that of LAGP. To accommodate predictors trained on larger data sets, I increase the size of the training data set to 40 thousand on a 200×200 grid, and similarly expanded the testing set is on a 199×199 grid (size 39601), both otherwise generated as described above. The black line in the right panel of the figure represents the score of PALMs with the number of space-filling centers indicated on the x-axis. On the right, total predictive time is plotted for the same fits. LAGP is in red on both plots. Note that the line is horizontal because LAGP builds a separate model for each prediction, and thus does not have an increasing size component. Of particular interest is PALM's predictive time, shown in the right panel, which indicates a very slow increase. The predictive complexity of PALM increases in $\mathcal{O}(N_C^2)$, although even for a relatively large number of centers the trend still appears linear. Despite diminishing returns in score as model complexity increases, a larger

PALM represents generally better performance in terms of cost–benefit trade-off relative to raw LAGP.

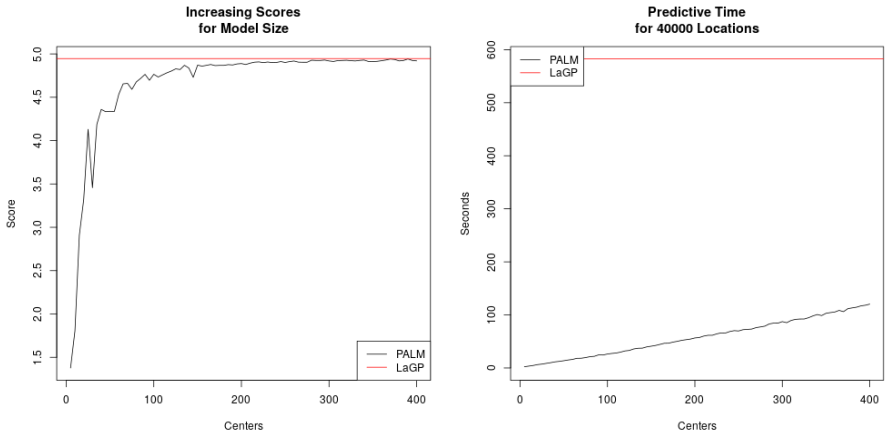

Figure 3.9: Score (left) and predictive time (right) for increasingly large PALM v. LAGP.

For deterministic data, accurate variance representation allows for PALM to perform much better than LAGP with a vastly improved predictive time. Figure 3.10 shows the performance of both models on a Herbie's tooth (3.2), generated here with a standard deviation of 0. The training and test sizes, and design locations remain the same as above. The plot should be read the same way as Figure 3.9. Notice that PALM matches and surpasses the performance of laGP while maintaining a large advantage in time.

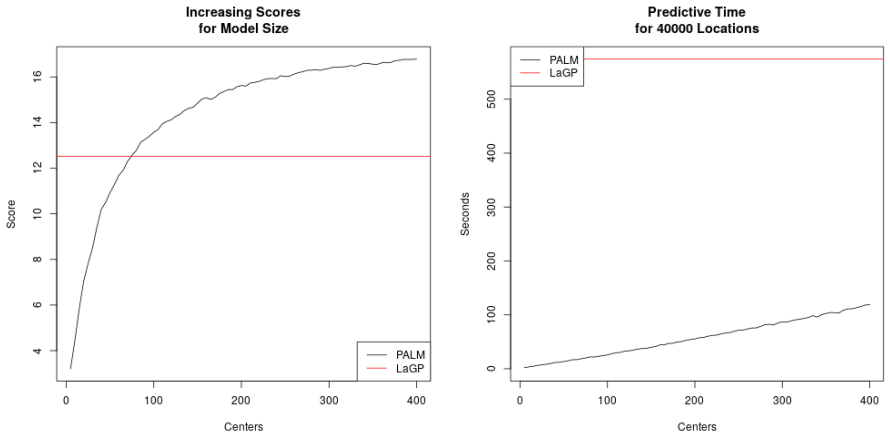

Figure 3.10: Analog of Figure 3.9 on deterministic Herbie's tooth.

3.8 Satellite Data

To apply PALM in a real data setting while simultaneously comparing it to other state of the art methods, I introduce the Sattelite Temperature dataset. In [24], thirteen state-of-the-art spatial smoothing methods, many based on GPs, were compared out-of-sample on satellite temperature data under a variety of performance metrics. This provides a real data example over which I can benchmark PALM to modern competitors. Several were reviewed in Section 1–3.1, including a variation on LAGP. The big advantage of this example over other similar "bake-off" style comparisons is that each method is applied to the data by its progenitor. Thus, no method gains an advantage due to familiarity of the scientist with its implication. I view a comparison of PALM to the models built here as a true comparison of the method to its biggest competitors. All methods and their implementation can be found in the repository linked below.

https://github.com/finnlindgren/heatoncomparison

The dataset consists of land surface temperatures measured remotely over 150 thousand locations selected from a grid, however 1691 record "missing" (or `NA`) values, leaving 148309. See the left panel of Figure 3.11. Heaton et al. partitioned these data into training and testing sets; see the right panel. The training data represents a large portion of the full data, at 105569 sites, but has substantial swaths of sparse or entirely missing coverage (right panel of Figure 3.11). In the exercise reported in that paper, the testing set was completely hidden from competitors.

I built two variations on PALM to add into the mix. The first uses $N_C = 2000$ local models built in the usual fashion: to directly predict the response. Centers for those models are selected as a space-filling subset of the desired prediction locations – the testing set – which focuses the majority of our computational power on sites where the predictor will be

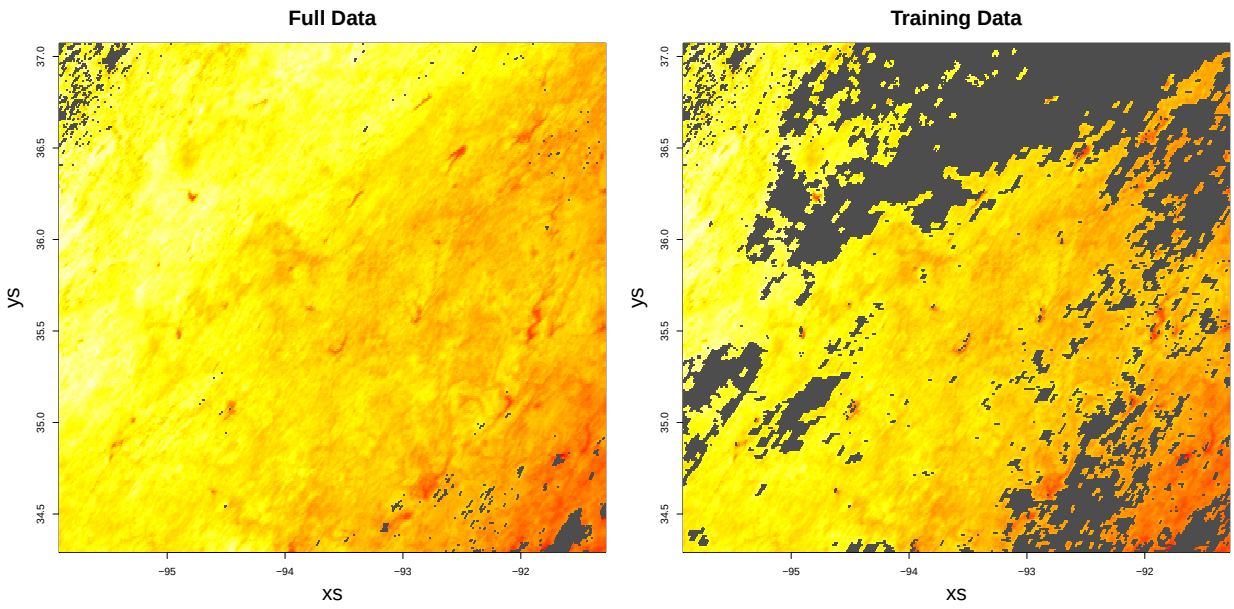

Figure 3.11: Left: full satellite temperature data set. The few missing values are displayed in gray. Right: dataset subset used for training. Much more data is left intentionally missing.

scored. In this way, the application of PALM here is similar to the transductive spirit of LAGP [48], where learning is focused on locations where predictions will be tested. It should be noted that the resulting PALM is not appropriate to describe the entire surface, but a prediction-focused model is better suited to the uniquely extrapolative nature task at hand. The second PALM variation is fit in two stages: first a global GP model is trained to a 1000-sized sub-sample of the data points; then a PALM is fit to residuals from that global subset GP using the same 2000 central locations from the first PALM variation.

Table 3.2 shows a comparison to the methods from the Heaton et al. bakeoff on the same metrics given in the original paper: mean absolute error (MAE), root mean squared error (RMSE), continuous ranked probability score (CRPS; see Section 4.2 of [17]), interval score (INT; see Section 6.2 of [17]), prediction interval coverage (CVG), and run time (minutes). Additionally I added a proper scoring rule [i.e., Eq. (27) from 17] metric to enable a more direct comparison to LAGP connecting to other criteria and comparison made throughout

this article. Notice that PALM was run on just one core, whereas several of the other methods leverage symmetric multi-core parallelization. As discussed previously, PALM is easily parallelizable. Speculatively, my method could be 40 times faster if given access to the same number of cores as, say, LAGP.

	MAE	RMSE	CRPS	INT	CVG	Time	Cores	Score
FRK	1.96	2.44	1.44	14.08	0.79	2.32	1	
Gapfill	1.33	1.86	1.17	34.78	0.36	1.39	40	
Lattice Krig	1.22	1.68	0.87	7.55	0.96	27.92	1	
Metakriging	2.08	2.50	1.44	10.77	0.89	2888.52	30	
MRA	1.33	1.85	0.94	8.00	0.92	15.61	1	
NNGP Conjugate	1.21	1.64	0.85	7.57	0.95	2.06	10	
NNGP Response	1.24	1.68	0.87	7.50	0.94	42.85	10	
Partition	1.41	1.80	1.02	10.49	0.86	79.98	55	
Pred. Proc.	2.05	2.52	1.85	26.24	0.75	640.48	1	
SPDE	1.10	1.53	0.83	8.85	0.97	120.33	2	
Tapering	1.87	2.45	1.32	10.31	0.93	133.26	1	
Periodic Embedding	1.29	1.79	0.91	7.44	0.93	9.81	1	
LAGP	1.65	2.08	1.17	10.81	0.83	2.27	40	-2.55
PALM	1.59	1.93	1.15	11.78	0.78	4.64	1	-2.85
Global + PALM	1.44	1.76	1.03	9.28	0.84	4.64	1	-2.39

Table 3.2: The original table from Heaton's bake-off paper with added rows for PALM comparison, and an added column for the proper scoring rule

Observe in the table that PALM is performing favorably compared to other state-of-the-art methods. Using only one core, I am able to make predictions faster than most other models, and ones which are competitive with the predictive accuracy (i.e., via MAE and RMSE) of the best of the others. In this example, the "Global+PALM" method does slightly better on all predictive measurements. This could be due to the extrapolatory nature of the problem at hand. With the majority of the computational power focused on an area with relatively little observed data, some estimate of a global trend improves the overall prediction quality.

While PALM has a massive speed advantage over many competitors in the group, it is

in the middle of the pack for most predictive metrics. Perhaps this could be improved by increasing the number of local experts, N_C at the expense of time. Also note that PALM, in its conception, is designed to leverage the wide area of accuracy of individual LaGP models across the input data (i.e. not to cover large areas of extrapolation). Despite this, the "off the shelf" version of PALM is still competitive with state of the art methods. I have not explored much the effects of inverse variance weighting in extrapolation, or the performance of "local" experts with data that is far removed from the central prediction location.

3.9 Methodological Notes

I have introduced a framework for local model averaging of Gaussian process predictoprs as an exemplar in a potentially wide class of such methods which I have dubbed PALM, for precision aggregated local models. The ingredients are a flexible local fit providing predictive means and variances, which we take from LAGP, and an aggregation scheme, which we take as inverse precision with adjustments for between-model covariance and overall ensemble size. We get all that while remaining in the class of GP regression – the resulting PALM-LAGP is an GP! Code to reproduce all of the results and plots used in this paper is open source and can be found in Gramacy Lab LAGP git repository.

https://bitbucket.org/gramacylab/lagp/src/master/R/boosting

It is easy to imagine other PALM instances. A number of other (locally appled) GP methods could be accommodated directly. We could swap the inverse squared exponential covariance function for other popular choices such as the Matern covariance, tapered or compactly supported covariance functions [26, 27], or a non-stationary options [e.g., 31]. It would not significantly change the method to swap out a traditional Gaussian process for

inducing points methods [42]. Modern sparse spatial processes, such as a nearest neighbor GP [NNGP, 11, 12] can be swapped in.

Even a simple linear model can be used as the local predictor. In the repository, the file `lmPALM.R` contains code that runs a PALM predictor using a third order linear approximation locally. The local experts are made by overlapping partitions of the input space for one and two dimensional examples: a sum of five randomly generated sine waves and Herbie's tooth, respectively. Covariance is determined via empirical predictive correlation between the points used to build two models. These examples are made heteroskedastic by choosing a random point in the input space to be the center for radially increasing variance. Performance on both the mean and variance surfaces can be observed in the plots contained within. Additionally, the boundaries between any existing partition model, such as [19, 33], can be made smooth by applying the PALM weighting scheme.

I envision a number potential opportunities for statistical and computational performance gains. Exploiting independence in inferences in order to parallelize the computational flow is one such aspect, alluded to at length earlier. On statistical efficiency fonts, it may be beneficial to re-select the local designs for PALM's local experts after the hyperparameters have been learned, e.g., using maximum likelihood estimates from LAGP. It may be beneficial to select a space filling design on the LAGP centers that is weighted in the input space based on the lengthscale parameters as well [46].

Finally, there are a number of PALM tuning parameters that I set by intuition, but which could potentially be optimized. First, I selected 50 as the number of points a local model should be trained on, primarily because this is the default in the `laGP` package. Within the provided code, this can be changed by altering the defaults to the `aGP` function. Of course, users should be aware of the inverse quadratic relationship between predictive time and accuracy based on the size of local GP models. While the default I am using works

well, it may be that other local model sizes may result in a better global model. One could imagine choosing such a tuning parameter via cross validation (CV). It may also possible to incorporate varying local model sizes in different areas of the input space that are easier or more difficult to capture. Updating local model size may be an alternative to sequential selection that allows the model to learn the relative complexity of the input space. Selection of the power to which weights are raised is another such tuning parameter. I chose an to use $\log_d N_C$ as a slowly increasing function of the number of centers modulated for dimension of the input space. CV may well be a more data-centric option for that parameter as well. The LAGP repository provides code that that optimizes the predictive power for a specific model, but it runs slowly and offers little benefit to statistical efficiency.

Chapter 4

Sequential Selection

Until this point I have utilized equally-spaced centers when PALM fitting. While this suffices for low-dimensional examples exhibiting highly regular (i.e., stationary) dynamics, bigger problems and more complex (e.g., nonstationary) response surfaces may benefit an uneven spread of computational resources to accurately capture the global surface. Consider the 2d exponential function from [19].

$$f(x) = x_1 \exp\{-x_1^2 - x_2^2\} \tag{4.1}$$

Exponential decay means that the response surface is essentially zero everywhere except near the origin. Although dynamics are smooth, bumps near the origin create subtle nonstationarity. Capturing both bumpy and flat regions, we take inputs in $[-2, 6]^2$.

In Figure 4.1, I have intentionally limited the number of local experts to illustrate how their uneven distribution may improve global accuracy. In the left panel, 16 experts have been assigned to fill out the space. Observe that only one local expert center resides in the interesting region near the origin. In the right panel, 4 points have been assigned to represent the interesting area before the other 12 are filled in by a space filling scheme in the flat area of the function. This adjustment in scheme results in much better coverage of the more complicated area, and an overall reduction in error, as I shall quantify in more detail in due course. For now, notice that the right panel exhibits greater relative isotropy compared to the left panel. Looking at Eq. (4.1), radial decay is evident in $x_1 \times x_2$ space.

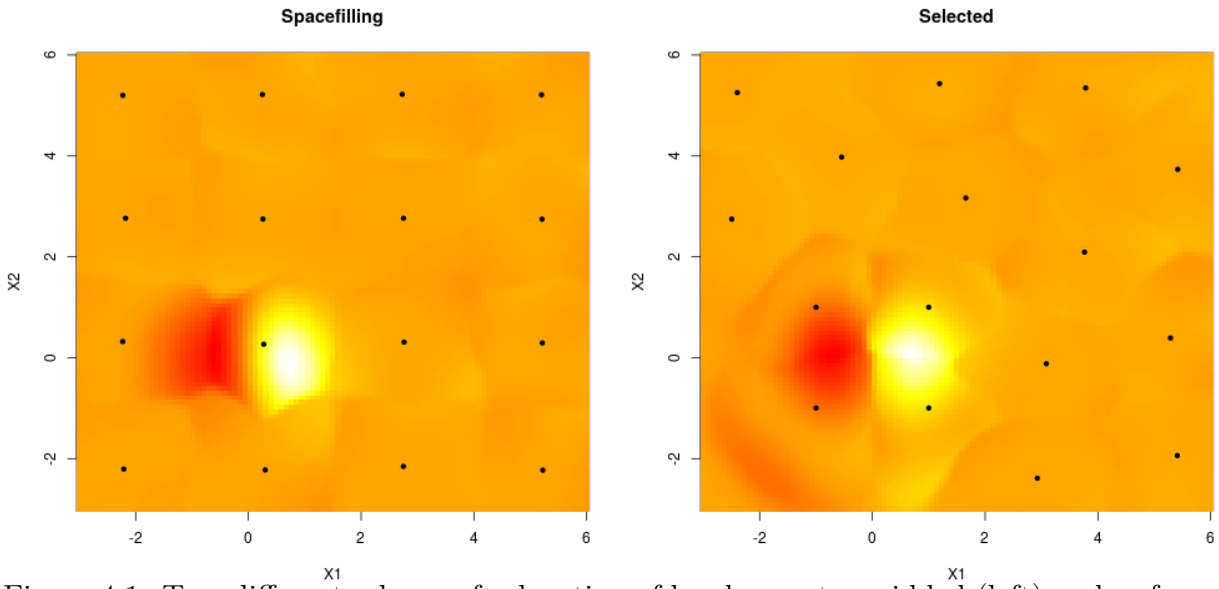

Figure 4.1: Two different schemes for location of local experts: gridded (left) and re-focused on the interesting region (right).

In the following passages I will define an algorithm for unevenly spacing computations throughout the input space, and compare its efficacy to the space filling selection of centers recommended above. Section 4.1 defines an algorithm whereby a PALM can be built by iteratively adding one additional center to an existing set of local experts. In Section 4.2 I benchmark the performance of this algorithm against the non data-driven alternative for increasing final model sizes. Section 4.3 compares the efficacy of this computational allocation to space filling models on a non-stationary 3d function.

4.1 Greedy Selection of PALM Centers

Allocation of expert "centers" in the simple case above is rather straightforward if one has prior knowledge of the surface, which is in general unrealistic. For most functions we will not have prior knowledge of hard to compute areas or, in more than 2 dimensions, an ability to directly visualize them. For similar reasons, optimizing the locations of all centers

simultaneously could be challenging. Instead, I find it advantageous to proceed sequentially. Given an existing PALM, I consider optimization of the selection of one additional local expert. Such a greedy strategy, targeting areas of the input space demonstrating the greatest benefit to expanded modeling fidelity, mirrors LAGP's scheme for choosing local design sights sequentially, e.g., using a variance reduction scheme (i.e., ALC).

In my context of greedy center selection I have the luxury of measuring out-of-sample accuracy directly, rather than relying on a model-based deduction of predictive uncertainty (ALC). Namely, I can calculate residuals between predicted and actual responses under the PALM fit. Therefore, my development of criteria for greedily selecting new centers focuses on identifying areas of the input space suffering from large such (absolute) residuals. Specifically I deploy k-means [30] on pairs (x, r), where $r = |y - \hat{y}|$ and x is the input location(s) of those y-values, in order to find clusters of high residuals where additional centers may be most helpful at increasing accuracy. In practice I find it helpful to scale those absolute residuals to be commensurate in size with the coding of inputs x in order to encourage the algorithm to form spatially contiguous clusters. A PALM with N_C centers should have on the order of N_C "local maxima" for poorly predicted areas, so in practice we set the number of k-means clusters to be equal to the number of local experts in the model. I denote the k-means algorithm, applied to a data set, X, with k clusters as kmeans(X, k). Next, I identify the cluster with highest average absolute residual, place a bounding box around that geographical region of the input space, and perform a local numerical optimization in order to fine-tune the location of the new center.

My choice of that final criteria for local optimization is nuanced. Ideally I would maximize score or aggregated absolute residuals. However, as a function of LAGP sub-design(s), both are discontinuous – score pathologically so – thwarting off-loading of optimization to simple library-based schemes. Instead, I leverage an empirical observation that good predictive

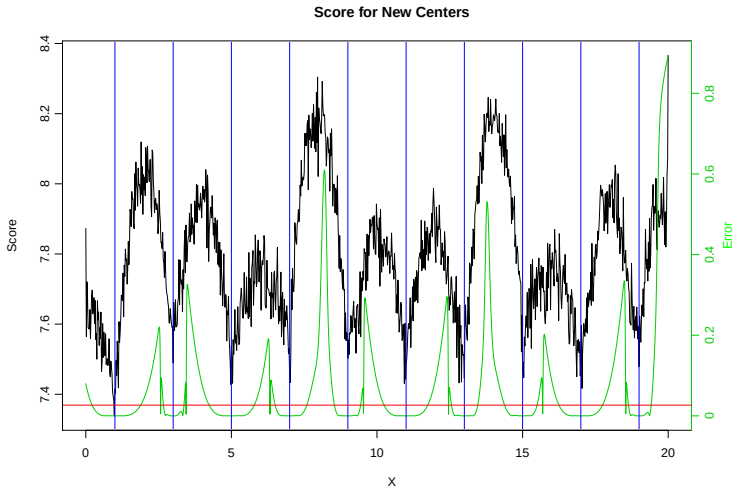

Figure 4.2: Score for potential new local expert placement against the absolute error of the existing model. The horizontal red line represents the score of an existing model, while the black line represents the score for a model updated with a center located at that spot along the number line. The (right) "error" axis indicates the absolute error for predictions using the existing model at that location.

areas tend to be far away (in the input space) from other local models in hard-to-predict areas. In other words, space-filling in a high residual region – e.g., as selected by k-means – offers a decent surrogate for score or residual-optimizing. Figure 4.2 shows the intuition for this idea. The horizontal red line represents the score of a PALM designed to fit a grid of 10000 equally spaced points along a sine wave with existing centers located at the vertical blue lines. The jagged black line is the score for a new center placed at that spot on the x-axis. The green line is the absolute residual at that point in the existing PALM.

Observe in the figure that if we were to select a local area that had high absolute error, and then subsequently choose a space filling location/midpoint within that space, it would correspond with a high score/high absolute residual area of the input space. Also note that the only center locations which make the PALM worse than the existing model are the locations of the current centers. Therefore I have designed our greedy scheme to chose a new center, c, such that $c = \text{argmax}_{c \in B} \min \|c - z\|_{z \in C}$ where c is the location of the new center

Require: existing `PALM` model with N local experts, input locations X, response vector y, matrix of existing center locations C

$\hat{y} \leftarrow \text{predict}(\texttt{PALM}, X)$ {Predict on the known input locations}
r $\leftarrow |y - \hat{y}|$ {compute absolute residuals}
$R \leftarrow \text{bind}(X, r)$ {Combine input space and residuals}
$K \leftarrow \text{kmeans}(R, N)$ {Form into k contiguous clusters}
$X_k \leftarrow K[\bar{r} = \max(\bar{r}), -r]$ {Find the cluster with high residuals}
$B \leftarrow \text{apply}(X_k, \text{range})$ {Compute the bounding box in the input space}
for i in 1 **to** M_s **do**
    $c_{\text{start}} \backsim \text{uniform}(B)$ {Uniformly generate starting point}
    $s_i \leftarrow \text{optim}_{c|c_{\text{start}} \in B} \min \|c - z\|_{z \in C}$ {Optimize from the random start}
end for
$c_{\text{new}} \leftarrow \text{argmax}_{s \in S} \min \|s - z\|_{z \in C}$ {Find overall best solution}
`PALM` $\leftarrow$ `PALM` $\cup$ `laGP`(c_{new}) {Append the `PALM` with a new `laGP`}
$C \leftarrow C \cup c_{\text{new}}$ {Add in the new center location}

Algorithm 2: Computing the location of one more center given an existing PALM fit.

to be added, B is the bounding box around the poorly predicted cluster of residuals, and C is the matrix containing the locations of centers already in the model. This "maximin" criterion is a continuous function which is easily optimized with library methods such as `optim` in `R`, although within the bounding box there may be several local maxima if the box itself contains any of the current PALM centers. To combat this I engage a randomized multi-start scheme. Finally, build an LAGP forming local expert for the PALM centered at the solution. For concreteness, the complete sequential PALM center-updating algorithm, with M_s being the number of multi-start maximin attempts is summarized as a pseudo-code in Algorithm 2.

4.2 Illustration

To get familiar with this greedy scheme, I provide a visualization and empirical comparison on the Gramacy and Lee function (4.1). A regular grid of 40 thousand training points is created deterministically. We begin with a PALM composed of a small space filling design

with 5 centers, seen in the top left of Figure 4.3. None of the points have been placed within

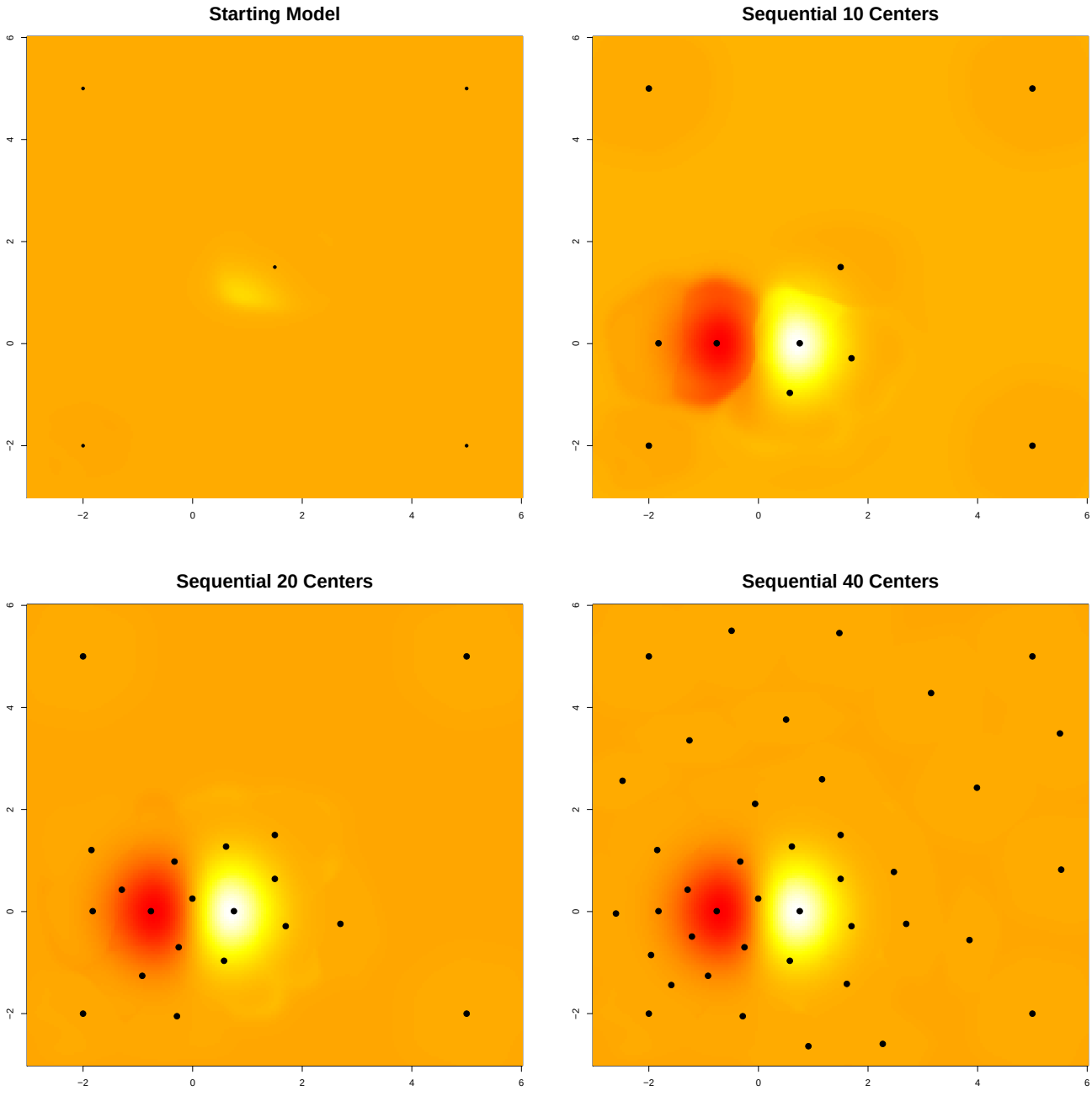

Figure 4.3: An illustration of the sequential selection process. The top left panel has 5 space filling centers. The top right, bottom left, and bottom right follow our sequential algorithm to achieve total sizes of 10, 20, and 40 centers respectively

the area of interest, so when I look at where the residuals are highest, it will be easy to

select the next area to place local experts. To the right, I have placed five more centers, one at a time following the greedy scheme outlined in Algorithm 2. Observe that all have been chosen in the difficult area of the input space, near the origin. The bottom left panel shows a further 10 sequentially selected centers. Now the greedy selector is starting to form a good picture of the interesting region and is exploring putting centers around the edges to determine where the function returns to a flat state. By the time we reach 40 centers, in the bottom right, the PALM has well described the whole region from $(-2, 2)$, and is selecting additional centers in a space filling fashion around the interesting region as well as elsewhere.

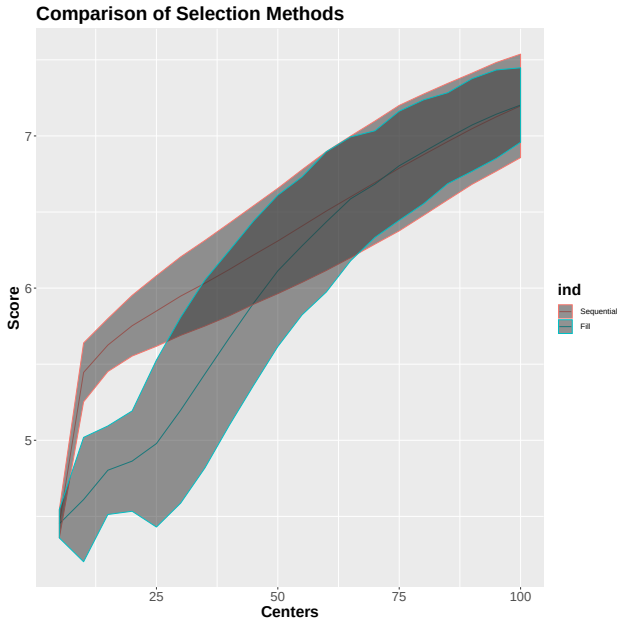

Figure 4.4: The score of models of various sizes when space filling, or applying a sequential selection algorithm on the Gramacy and Lee function.

Figure 4.4 shows how sequentially selecting can potentially drastically improve PALM fits, but will do no worse than a space filling baseline. A grid of 40000 training points were generated from the Gramacy and Lee function (4.1) with a standard deviation of 0.01. For 100 such randomly generated data sets, PALMs were built with 5 to 100 centers incrementing by 5 in a space-filling fashion. The out-of-sample performance of this space-filling comparator is indicated by the blue lines in the figure. The red line, however, represents the above

greedy/sequential selection scheme. An initial/seed spacefilling model of size 5 was built, and all subsequent points were added by greedy selection following Algorithm 2. Notice that the sequential scheme has a sharp spike for smaller models, but levels out to match the space filling models in the long run. This represents a early concentration of centers around the area of interest, $(-2, 2)$ before the sequential scheme starts filling in the rest of the space.

In general, if there is no prior knowledge of the interesting areas of a function, sequential selection of center locations will provide a potential boost in model accuracy by concentrating resources in the active subspace of the model. When the resources allocated are enough to describe the entire surface well, a sequentially selected model will match the performance of a spacefilling scheme. By front loading calculation on determining center locations, the overall performance of a PALM fit can be improved.

4.3 Michalewicz Function

For a higher dimension, direct comparison of sequential design versus space filling centers with a fixed computational budget, consider the Michalewicz function [32] in 3d. This is a good test function for PALM because it is highly non-stationary with large flat areas abutting drastic drops. As the number of dimensions increases, the number and severity of the dips also increases. For any number of dimensions, d, with a steepness parameter, m, the Michalewicz function is defined over $[0, \pi]^d$ as

$$f(\mathbf{x}) = -\sum_{i=1}^{d} \sin(x_i) \sin^{2m}\left(\frac{ix_i^2}{\pi}\right). \tag{4.2}$$

Because it is defined additively over inputs, a slice along any dimension will have the same shape regardless of its position in the other $d-1$ dimensions. This allows one to intuit high

dimensional dynamics visually by overlaying one-dimensional slices. Figure 4.5 shows slices created by the first three inputs. Any output in the 3d surface can be found by adding the x value for the individual dimensions with the corresponding point along that dimension's line. The second panel of the plot shows heat plot rendering in two dimensions, combining the first two slices from the left panel. The shape of the function in either direction remains the same regardless of the slice, although the depth of the slice changes.

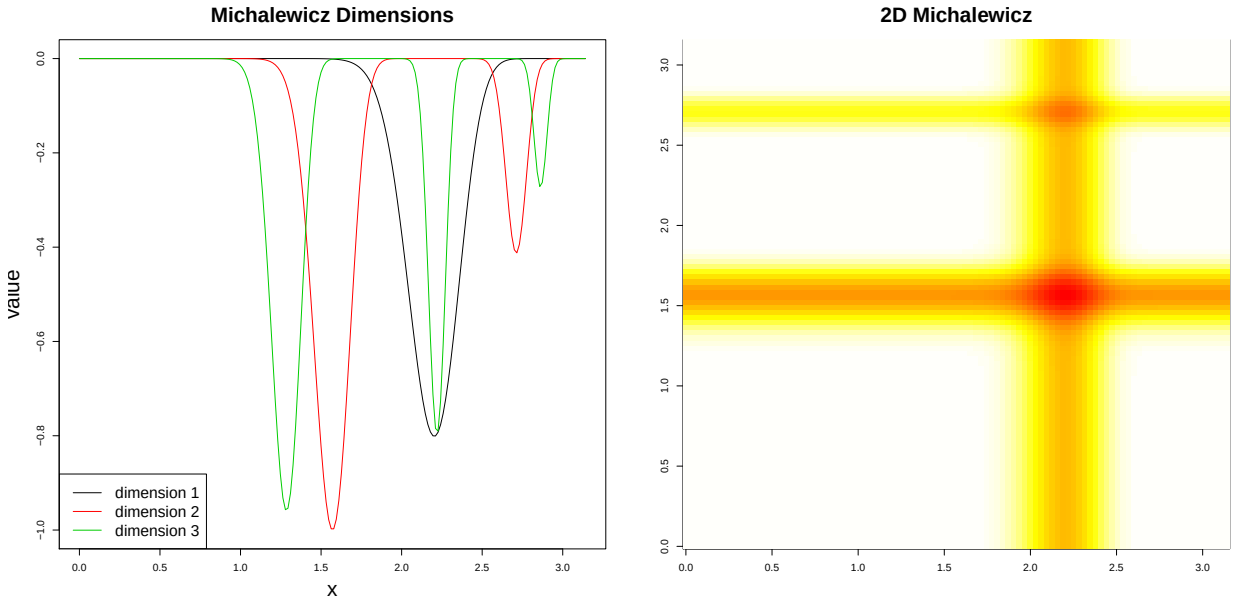

Figure 4.5: Left: The relative values of the first three dimensions of the Michalewicz function. The combined 3d surface can be reconstructed by adding points from these lines together. Right: a visualization of the Michalewicz function in 2d.

To test the effectiveness of space-filling versus sequential designs for local experts in three dimensions, I generated a grid of forty points in 3d from the Michalewicz function, giving a total training data size of $N = 64000$. I added $\mathcal{N}(0, \sigma^2 = 0.0025)$ noise to compare our methods in a stochastic setting. Finally the computational resources of both center selection methods were restricted to budgets of $N_C \in \{100, 200\}$, separately. The testing data set, over which PALMs were compared on RMSE and score, was generated the same way on a grid of size 39^3. The entire process, from noise generation to center location selection, to

fitting and prediction, was repeated one hundred times to get an accurate estimate of the comparison through Monte Carlo averaging.

	100 Centers		200 Centers	
	RMSE	Score	RMSE	Score
Spacefill	0.1237	3.5623	0.0878	4.1012
Sequential	0.1073	3.5704	0.0776	4.1212

Table 4.1: Average RMSE and Score for spacefilling models on 3d Michalewicz.

Focusing first on the $N_C = 100$ case, Table 4.1 shows that despite a notable improvement on RMSE for sequential over space-filling center selection, score is only slightly better. This is because the space-filled variation yields lower average variance. Note that the improvement in RMSE is still enough that the model scores better despite having higher variance, indicating hat we have spent our computational resources wisely. When we increase the number of local experts to $N_C = 200$, the gap in RMSE closes slightly, while the difference in score is still negligible. This confirms our instinct that sequential selection should be used in situations where the computational resources are limited.

Chapter 5

Variations

Here I explain several variations on the basic PALM methodology outlined above. I give the motivation for algorithmically changing the location of PALM centers in Section 5.1, and show the results of a fast method of doing so. Section 5.2 notes an issue PALM has with increasingly large datasets, and proposes a solution that also mitigates some of the effect of increasing dimensionality. Section 5.3 examines PALM with different compentent models, and adds heteroskedasticity to the PALM methodology.

5.1 Cycling

A natural extension of the greedy center-adding scheme described in section 4.1 is to re-evaluate the position of the local experts in the full model as a post-processing step. For this, I introduce the idea of *center cycling*, allowing for improved (re-) selection of model location(s) given a set number of expert models.

5.1.1 Single Center Cycling

For a given PALM, produce a hold-one-out prediction across local experts. This can be done for the entire set of PALM experts with only one prediction made per model, as the only piece of a prediction that needs to be recalculated with one model removed is the weight.

Then choose the local expert that reduces model score the least to remove. Once one center is removed, apply the sequential scheme (Algorithm 2) to select one additional center, as if it were being newly added. This process can be repeated until an equilibrium is reached.

The exact process of cycling out the worst centers to introduce new ones can be found in Algorithm 3. I am introducing the function `PALM_combine` to represent the process of forming weights from a set of local models (denoted here as `LM`) to get a PALM mean and variance. This is an iterative algorithm that can be repeated any number of times.

Require: existing `PALM` model consisting of N_C local models (`LM`), input locations X, response vector y, matrix of existing center locations C

1: **for** i in 1 **to** N_C **do**
2: $(\hat{y}_i, \hat{s}_i^2) \leftarrow$ predict($\texttt{LM}_i, X$) {Predict the known input locations from each model}
3: **end for**
4: **for** i in 1 **to** N_C **do**
5: $(\hat{y}_{-i}, \hat{s}_{-i}^2) \leftarrow$ `PALM_combine`$(\hat{y}_{1:N_C,-i}, \hat{s}_{1:N_C,-i}^2)$ {Get predictions without model i}
6: $S_{-i} \leftarrow$ score$(\hat{y}_{-i}, \hat{s}_{-i}^2)$ {Calculate score without model i}
7: **end for**
8: index $\leftarrow$ argmax$(S_{-1:N_C})$ {Find the local expert that contributes least}
9: `PALM` $\leftarrow$ remove(`PALM`, index) {Remove the indexed model}
10: `PALM` $\leftarrow$ seq(`PALM`, 1) {Sequentially add a model}
11: Go to (1:)

Algorithm 3: The algorithm that cycles from a `PALM`. Here, `PALM_combine`$(\hat{\mu}, \hat{\sigma}^2)$ refers to the weighting and recombination scheme outlined in section 3. seq(`PALM`, n) refers to adding new centers by algorithm 2.

Below we can see that cycling centers can produce marginal performance increases over sequential selection with limited computational resources. I generated a set of 10 thousand points on a grid from the Gramacy and Lee function (4.1) on $[-4, 2]^2$ with a standard deviation of 0.01. To compare performance, I made a spacefilling PALM with 20 centers, and a sequentially selected PALM with a starting size of 3, and a final size of 20. The results seen in the last row are created by running the sequentially selected PALM through the cycling algorithm 20 times. All three models are compared on a testing set of 9801 points generated from the same function for both RMSE and score.

	RMSE	Score
Spacefill	0.0719	3.9200
Sequential	0.0539	4.6113
Cycled	0.0526	4.5230

Table 5.1: Mean RMSE and Score for Spacefilling models, a greedy sequential selection, and a cycling out the worst points from the sequential selection across 100 runs

Notice that while cycling has further decreased the RMSE, the score of the model has actually gone down. This is because the removal step considers score, i.e., a metric that depends on both the mean and variance of the global model, while the sequential addition step only chooses a space filling point within a box determined by poor mean prediction. In effect the removed center contributes the least to the combined mean and variance predictions, while the added expert improves only the mean prediction. This is reflected by the improvement in mean metric and simultaneous drop in proper score. Unfortunately, as previously mentioned, optimizing over proper score of the model with an additional expert at each location, the metric we wish to improve, in the greedy search is not efficient, owing to the pathological discontinuity introduced by LAGP center placement. It is, however, possible to change steps 1 – 6 of Algorithm 2 so the initial bounding box is drawn around an area that is poorly predicted in mean, and/or variance. The addition step can be improved to consider both mean and variance by replacing the metric in step 2 of Algorithm 2, which defines the residual set that will be clustered by Kmeans, with $r \leftarrow (\hat{y} - y)^2 + \hat{\sigma}_X^2$, where $\hat{\sigma}_X^2$ is the vector of point-wise variance estimates. I call this new metric variance inclusive residuals.

Figure 5.1 shows the effect using the variance inclusive residuals has on Kmeans for PALMs of various size. Recall that for the Gramacy and Lee function (4.1) the only interesting area is around $[-2, 2]^2$. The top line shows both residual metrics applied to a small space filling PALM (five experts). Here, both metrics are driven by the high residual area where both models have failed to describe the mean surface in the complicated area. The

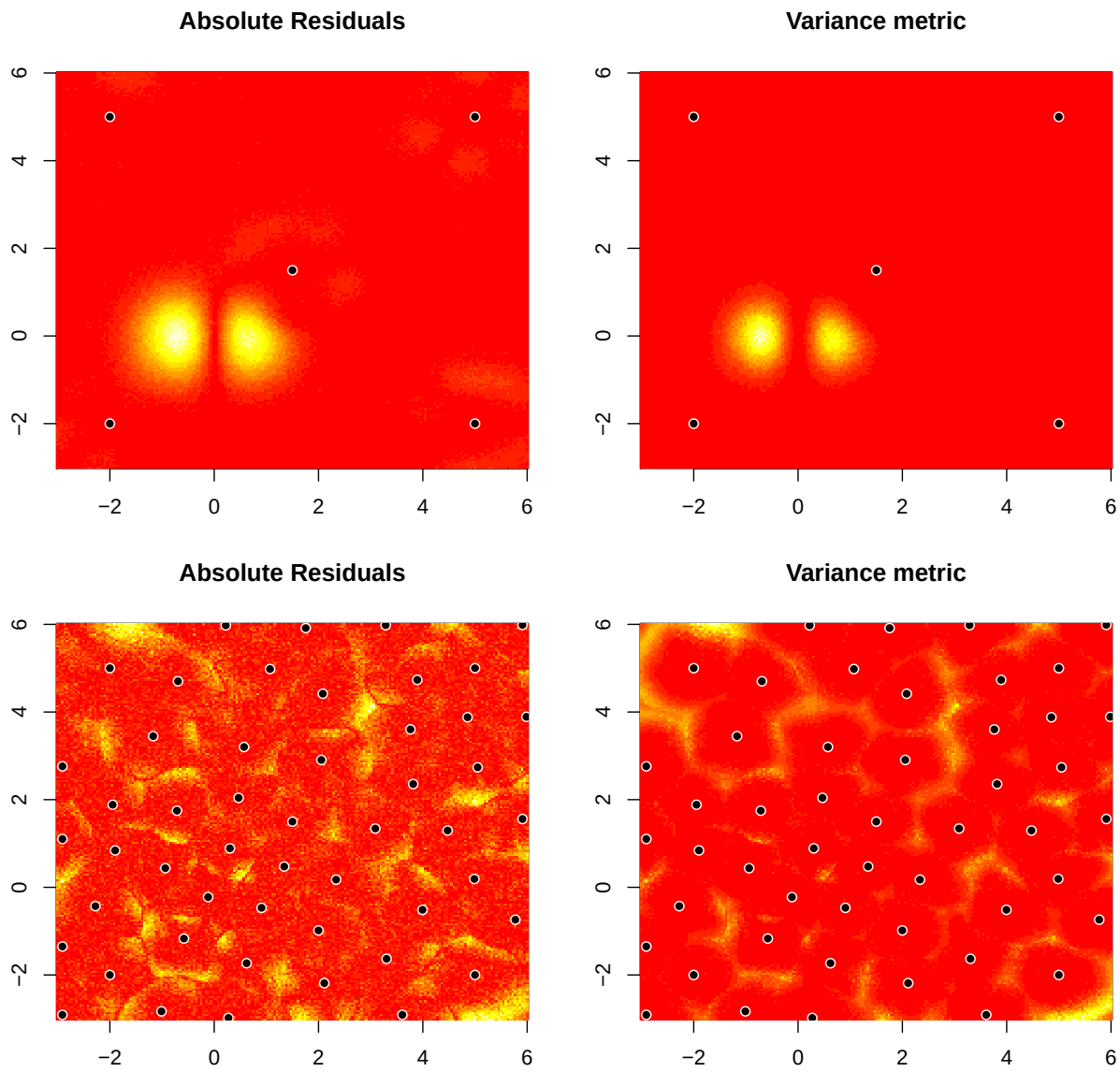

Figure 5.1: Top Left: The surface of absolute residuals from a 5 expert selected PALM on 40 thousand points from the Gramacy and Lee function. Top Right: the variance inclusive residual surface from the same model. Bottom Left: Absolute residuals from a 50 expert sequentially selected PALM trained on the same data. Bottom Right: Variance inclusive residuals shown for the larger PALM.

bottom images show the same surfaces for a 50 expert PALM that has been sequentially selected. On the left, in the absence of any true high residuals, the absolute residual metric focuses on the areas at the periphery of experts that are currently present in the model.

The variance inclusive metric on the right emphasizes areas where the model has not learned anything. Thus, using a surface for Kmeans that considers both the mean and the variance helps keep mean metrics like RMSE low without hurting the model performance on proper scoring metrics.

5.1.2 Fast Cycling

Another hurdle to successful implementation of cycling as a mitigation for the greedy nature of sequential selection is that, as it stands, it is by far the largest computational burden amongst the corpus of PALM methodologies. Despite the fact that predictions for each model need only be made one time, combining them into a mean and variance to determine which model should be removed (step 7 in algorithm 3) is an $\mathcal{O}(N_C^2)$ operation which must be repeated, alternately holding out each center. Because the `PALM_combine` must be performed for each local expert in each cycle, this makes the total computational complexity $\mathcal{O}(N_C^3 \times N_{\text{cycles}})$ for N_{cycles}, a given number of removal/addition pairs. It may also be possible to replace the `PALM_combine` with a more computationally efficient method that produces the same, or approximately the same result. In absence of a more computationally efficient trick, I propose replacing the process of determining which local expert is the best to remove in algorithm 3 by iteratively removing every component model and adding one new one using the new residual surface, $(\hat{y} - y)^2 + \hat{\sigma}_X^2$, in the sequential addition algorithm. Because PALM prediction is a $\mathcal{O}(N_C^2)$ operation, this method is $\mathcal{O}(N_C^2 \times N_{\text{cycles}})$. Cycling all centers in the model (i.e. $N_{\text{cycles}} = N_C$) is therefore $\mathcal{O}(N_C^3)$ rather than $\mathcal{O}(N_C^3)$ per model replaced.

Figure 5.2 shows one step of this fast cycling process. Here, the surface being shown is $(\hat{y} - y)^2 + \hat{\sigma}_X^2$ for the full sequentially selected model with one center removed. The centers that are still part of the model are shown in white, while the removed center is in black. The selected "high residual" area from the Kmeans algorithm is shown in blue. Initialized

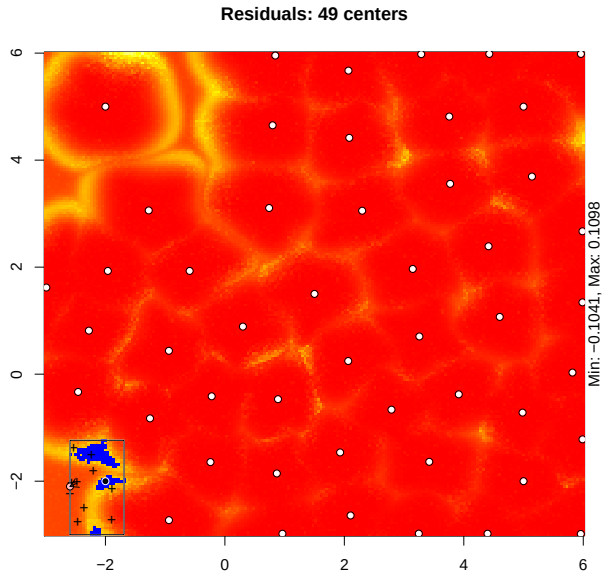

Figure 5.2: One center removal/addition for the fast cycling algorithm. The center of the removed expert is in black, while the added center is within the bounding box shown in pink.

space filling points are indicated by '+', while the eventual new center selection is shown in pink. Notice that the newly selected local expert is placed firmly in an area where the model had high uncertainty before. This is the behavior we want to see if we want to improve the performance of sequentially selected models.

5.1.3 Empirical Results

Table 5.2 shows the effect of fast cycling on various types of models on the Gramacy and Lee function defined on $[-3, 6]^2$. For a PALM consisting of centers chosen by maximin distance (labeled Spacefill), cycling out every point once, or ten times does not substantially change the performance for either the score or the RMSE. This is due to the tendency to replace a removed center with one in roughly the same location. For sequentially selected centers, however, cycling can slightly improve the performance on both. This shows that the greedy algorithm, while producing good results, leaves some room for improvement.

	Spacefill	Sequential	Spacefill C1	Sequential C1	Spacefill C10	Sequential C10
Score	6.9567	6.8961	6.9395	7.0816	6.9203	7.0989
RMSE	0.0209	0.0205	0.0210	0.0184	0.0208	0.0190

Table 5.2: Performance on a 40 thousand observation data set from the Gramacy and lee function on $[-3, 6]^2$. Each PALM is restricted to 50 experts. Results shown for a fully spacefilling selection as well as a sequentially selected model. Each model has all points cycled once (C1), and ten times (C10).

To visually see the effect this has on models, begin with Figure 5.3. On the left the final center selection for 50 experts trained on the Gramacy and Lee function shows a good concentration of computational power in the region of interest. The placement outside that area is clumpy in some regions while spaced out in others. Our knowledge of this function tells us that the whole space outsie $[-2, 2]^2$ is essentially flat, and should be treated equally. This does not happen, however, because center selection in this area is driven by high residuals.

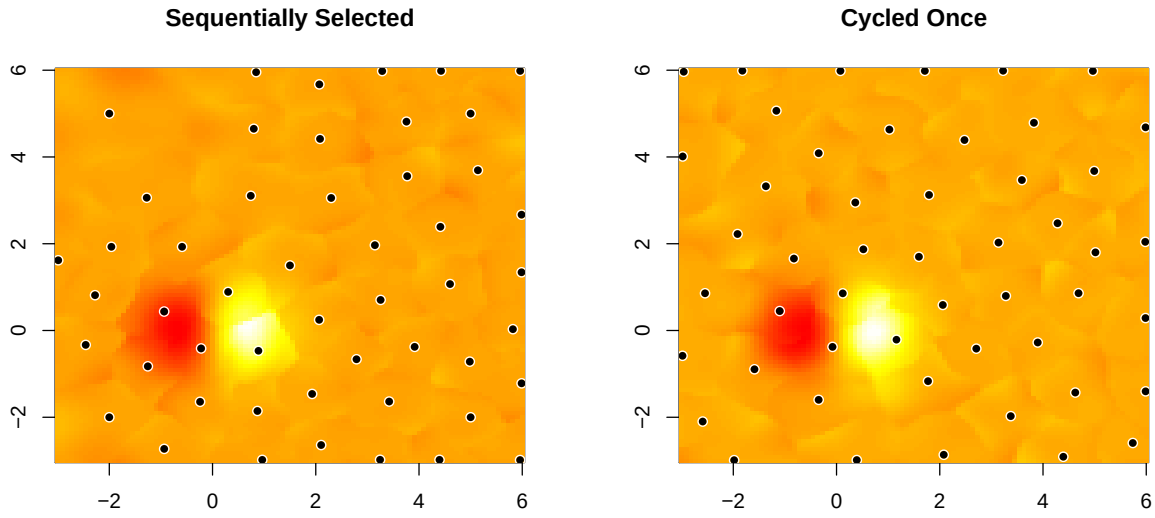

Figure 5.3: Left: Sequential center selection results and surface for a 40 thousand point sample from the Gramacy and Lee function on a regular grid spanning $[-3, 6]$. Right: Results of iteratively cycling out each center and adding a new one by pseudo MSE for the sequentially selected model at left.

On the right, we have cycled the selection locations for the left model one time. While

there is still a concentration of centers surrounding the interesting area, the flat area is covered by a more space filling set of models. This is because, in absence of high residuals, the placement of the bounding box in the sequential algorithm is driven by high variance areas, i.e. areas far away from the experts already included in the model.

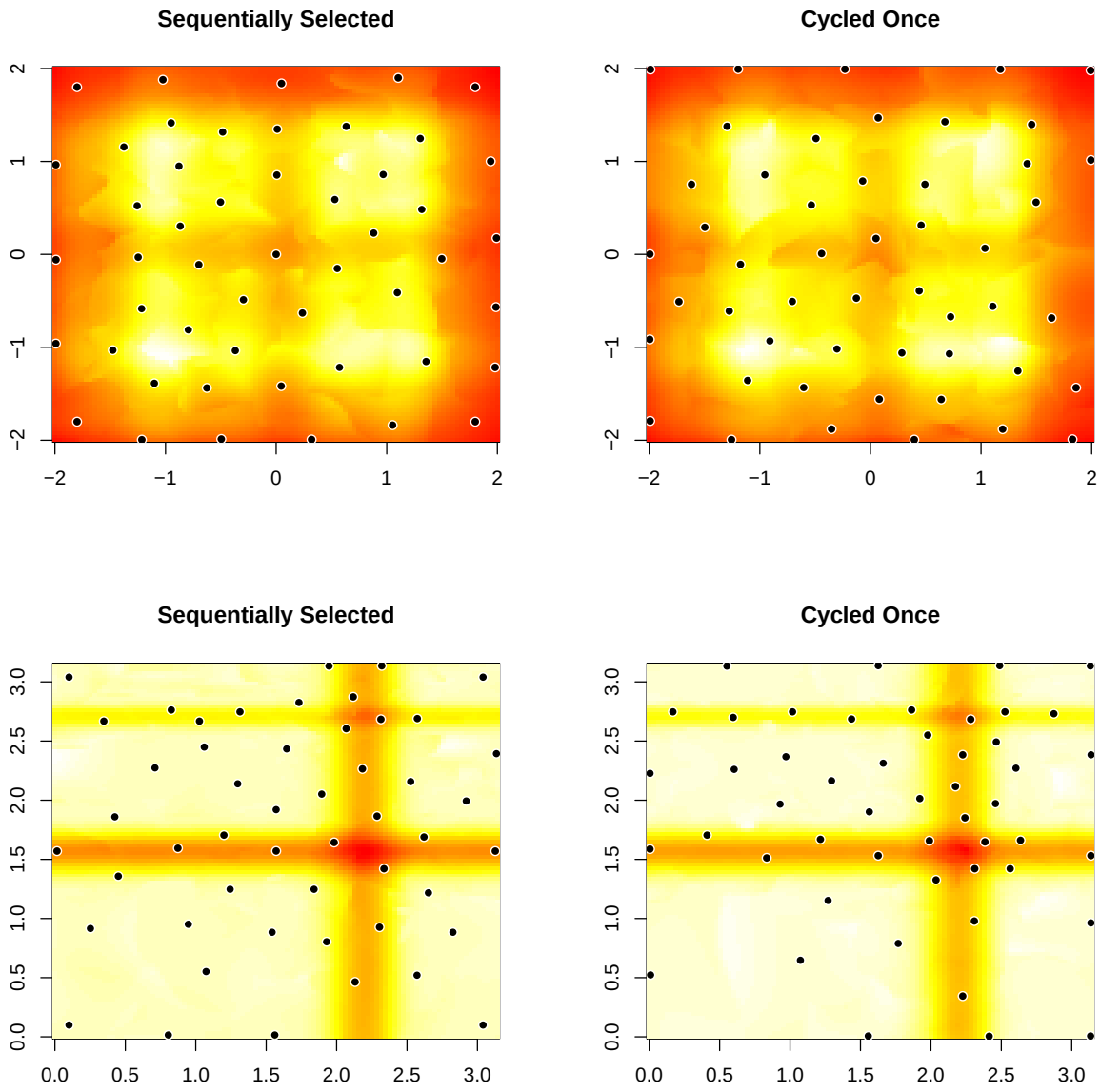

Figure 5.4: Comparisons of sequential versus cycled PALM models for Herbie's tooth (Top) and the 2d Michalewicz function (Bottom). Both sets are generated on a regular 200×200 grid with $\sigma = 0.01$.

Figure 5.4 shows this comparison on other examples used throughout the paper. In the top row, the universally complicated Herbie's Tooth (3.2) shows a relatively spacefilling selection for both methods. This is what we should expect, since is relatively stationary compared to Gramacy and Lee. In the bottom two panels, methods are compared on a two dimensional version of the Michalewicz function (4.2). In this case the point selection appears to be visually similar between the two models, however the two models do not perform the same on proper score.

Data Generator	Sequential	Cycled
Gramacy & Lee	6.90	6.97
Herbie's Tooth	6.96	6.85
Michalewicz 2D	6.45	6.72
Michalewicz 3D	2.52	2.58

Table 5.3: Proper score shown for sequentially selected versus cycled models on various examples. All 2D examples are the result of regular 200×200 grids over their domain. The Michalewicz 3D example is generated on a $50 \times 50 \times 50$ grid.

Table 5.3 shows the effect this process has on proper scoring. Here the 2D functions are all evaluated using 50 expert PALMs on 40 thousand observation training sets. The testing set is over 40401 points generated on the same domain. The Michalewicz 3D set is modeled using a 100 expert PALM on data sets of 125 thousand for training, and 132651 for testing. Note that for the functions where computational resources should be unevenly assigned (Gramacy & Lee, and Michalewicz), cycling improves the allocation of those resources. As we might expect, for Herbie's Tooth, a globally complex function, reassigning the location of computational resources does not improve the global PALM.

For a final note on the change of objective function for identifying poorly predicted areas, the new surface may produce better results in the initial sequential selection as well. Figure 5.5 compares both objective functions on the Gramacy and Lee and Herbie's tooth functions. In both plots, the left box represents $|y - \hat{y}|$, while the right box represents

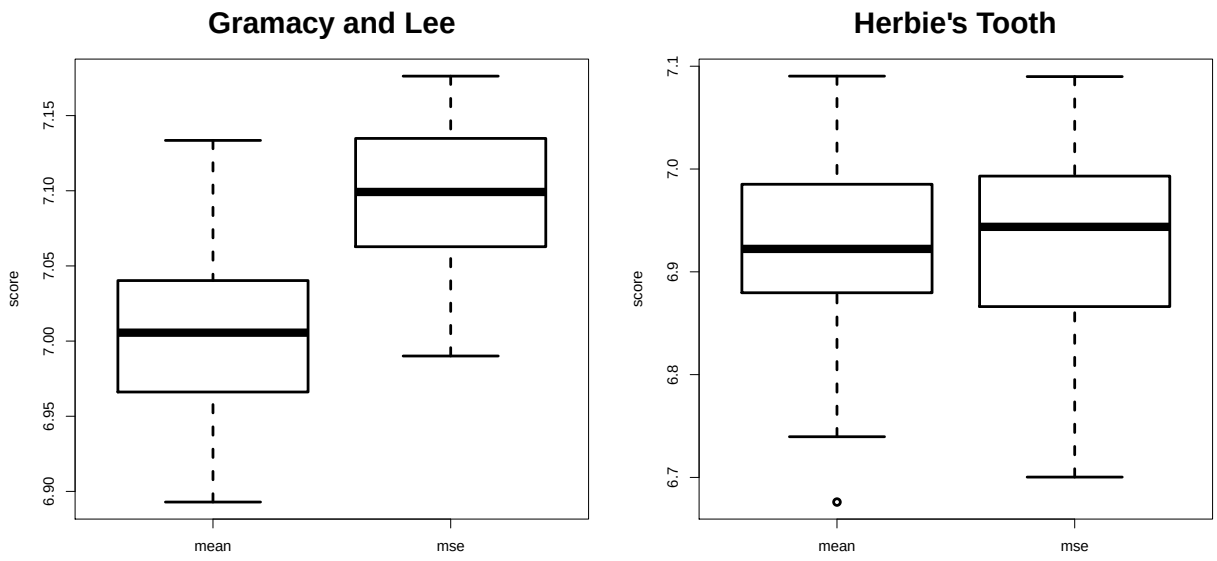

Figure 5.5: Boxplots comparing sequential addition by $|y - \hat{y}|$ and $(y - \hat{y})^2 + \hat{\sigma}^2$. Left: Performance on Gramacy and Lee. Right: Methods compared on Herbie's Tooth.

sequential addition using $(y - \hat{y})^2 + \hat{\sigma}^2$ in the Kmeans evaluation. For Gramacy and lee, where the resources should be unevenly placed, the new objective performs much better. Meanwhile, on the relatively uniformly complex Herbie's tooth function, the change still produces slightly better results on average, although here the variance is much larger.

5.2 PALM with Set Area of Influence

Using LAGP, a model designed to predict at one location, presents an additional issue for creating a global surface with large data sets. Since the subset selection for LAGP selects a large portion of NN points, if the density of the grid increases, the global influence of an LAGP model decreases. The effect for PALM is that more component models will be required if the amount of data increases regardless of the complexity of the underlying function.

The result of this is that the subset chosen for each local expert may be sub-optimal

even in cases where a better selection is available. To illustrate this, see Figure 5.6. In the left panel, 100 local experts are trained to fit a grid of ten thousand points generated deterministically from Herbie's tooth 3.2. The centers of the local models are denoted by large black dots, while the small black dots mark training locations that are used by at least one of the components of the full PALM. This same training regimen is repeated in the right panel for a training set on a grid of 160 thousand locations. Notice that although the centers have not moved, the model for each covers a much smaller area of the input space. The important detail for this example is that the smaller grid is a subset of the larger grid. Thus the model on the left was available to be selected when training the model on the right.

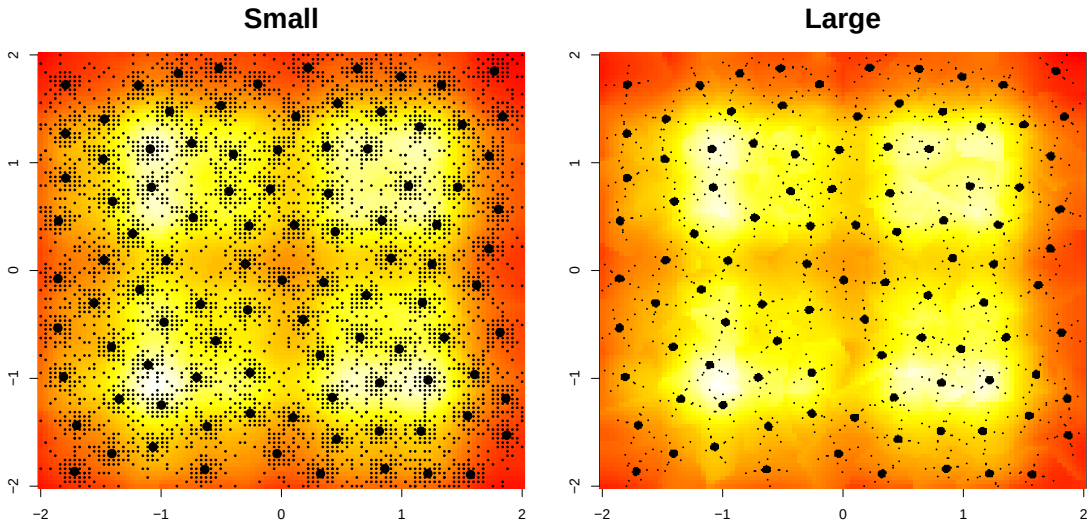

Figure 5.6: Left: PALM trained on 10000 training locations. Right: PALM trained on 160000 locations. In both plots the large dots are the centers and the small dots are the training locations used for each model.

This collapsing area of influence has a nontrivial effect on the predictive performance of the model. Many more local experts, and thus much increased computation, is necessary to learn the same amount of information about the global surface in the large data case.

5.2.1 Algorithmic Data Thinning

To remedy this issue, we can algorithmically thin the data as a preprocessing step. Given an estimate of the maximum global lengthscale (which is already provided by the PALM framework if it is unknown), iterate through each training point and remove all other training points that exceeded some threshold of GP correlation with it. This process eliminates points that may be better for single value prediction, but worse for global prediction, from the larger dataset before PALM training begins. The process is summed up in Algorithm 4.

Require: Training data X, estimates of maximum global lengthscale Θ, threshold value $\rho_{\max}$

$i = 1$, $X_{\text{thin}} = X$ {Initialize a counter, and set the thinned data to the full data}
while $i < |X_{\text{thin}}|$ **do**
  $\text{k} = k_{\Theta}(X_{\text{thin},i}, X_{\text{thin},-i})$ {Calculate the correlation with the remaining points}
  $X_{\text{thin}} = X_{\text{thin}}[\text{k} > \rho_{\max},]$ {Remove rows with higher correlation than $\rho_{\max}$}
  $i = i + 1$ {Iterate to the next point in the thinned set}
end while
return(X_{thin}) {Return the thinned data}

Algorithm 4: Process for thinning training data as a preprocessing step.

The result, when applied to the larger data frame, is a training set that mimics the thinness of the smaller data. This allows the completed PALM model to be a better representation of the true global surface. Figure 5.7 shows the effect that preprocessed thinning has on the 160 thousand point training data shown above. Thinning is performed using $\rho_{max} = 0.99$. A higher threshold will obviously result in more of the training data kept, and less spread for each of the local models. Note that instead of the tight clumps around models from the full data frame, each component local expert now covers a broader area of the input space.

The result this has on model performance is large. Table 5.4 shows the effect of data size and the thinning algorithm on the predictive performance of the resulting PALM. In addition to the 10 thousand point (Small), and the 160 thousand point (Large) training sets

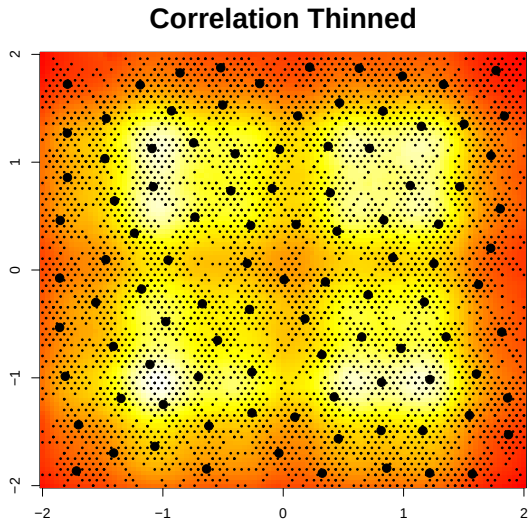

Figure 5.7: A PALM model fit to a very large training set after a preprocessed thinning step.

shows above, I have added a 40 thousand point training set, labeled 'Medium'. The table shows the predictive score of the model on a testing set of 10201 training points generated on a regular grid. Even though Small $\subset$ Medium $\subset$ Large, as the size of the training data increases, the resulting PALM performs worse. After the thinning on is performed on the Big training data, PALM is able to recover the prediction quality of the Small training set despite using barely more than half as many points.

	Score	N
Small	13.75	10000
Medium	12.14	40000
Large	9.46	160000
Thinned	13.96	5360

Table 5.4: Score and training data size for various model fitting schemes. All fitted models are trained on subsets of the 'Large' training data.

5.2.2 Set PALM

Although this method of ensuring a PALM has shown good early results, conceptually it is less than ideal. Not only does it require completely ignoring pieces of the collected data, but

eliminating data based on distance metrics removes the ability of the model to learn close lengthscales in areas of the space where they may be applicable. As discussed at length in the Section 2, farther points can be better for estimating the mean, but are less than ideal for accurate hyperparameter estimation. Additionally, in the stochastic case, we are throw away information about the error at each location by eliminating nearby data.

To fix the issue of worse performance with increasing training size while maintaining the ability to obtain accurate hyperparameter estimates, I propose a different method to guarantee the wider global influence of PALM local experts. In its conception, LAGP restricts the variance minimizing properties of ALC to a single desired predictive location, however, ALC approximation can easily be applied instead to a set. Instead of fitting PALM to predict one point, I propose fitting each local expert to be optimal given size constraints for a set reference set. This could be done in the design phase, however, I will demonstrate the value of so called "`set_PALM`" as a post processing step to perfect an existing PALM model.

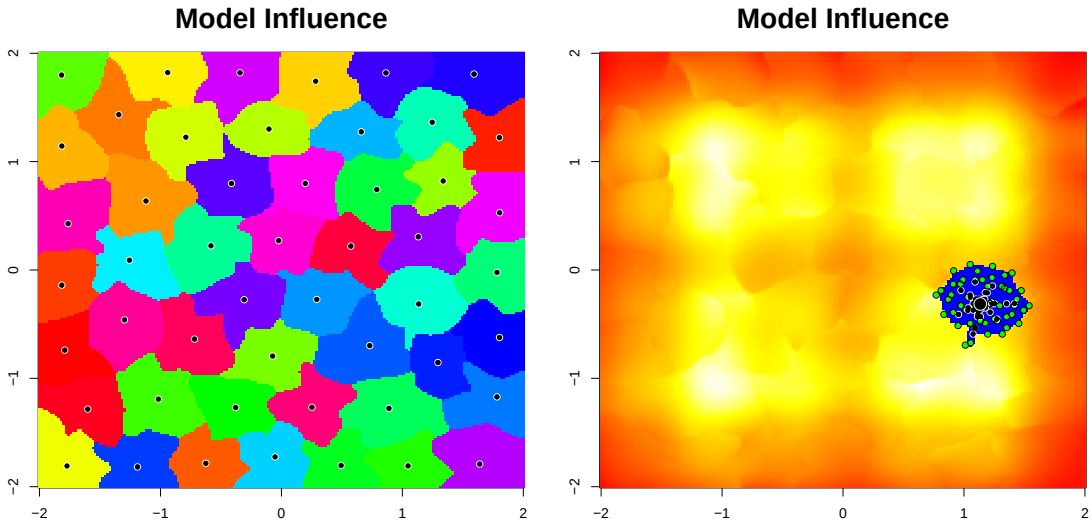

Figure 5.8: Left: Each center in a 50 expert PALM built on Herbie's tooth surrounded by its "area of influence." Right: Refit of one model. The black dot is the original center of the model with the smaller black dots representing the chosen design for that location. The Blue represents the area of influence for that model, and the green dots represent the new design chosen to predict that location.

To see the basic idea of post processing a palm model into a `set_PALM`, observe Figure 5.8. On the left is the influence of a 50 expert space filling PALM model designed to predict Herbie's Tooth. The colored splotches around each center (represented by black dots) represent the area where that model receives higher weight than any other expert in the ensemble. At right see the design effect of refitting one expert. Here, the blue shading represents the area where this model receives at least 90% of the predictive weight. The center, as well as the original X_i locations are shown in black. Notice that although this model will receive the highest portion of weight over the entire blue area, the selected model is very compact around the center location. Green dots represent the design chosen by `laGP` when the entire blue region is used as the reference set. Although the same number of points are chosen, the resulting GP is much better for predicting the entire region it represents within the global PALM model.

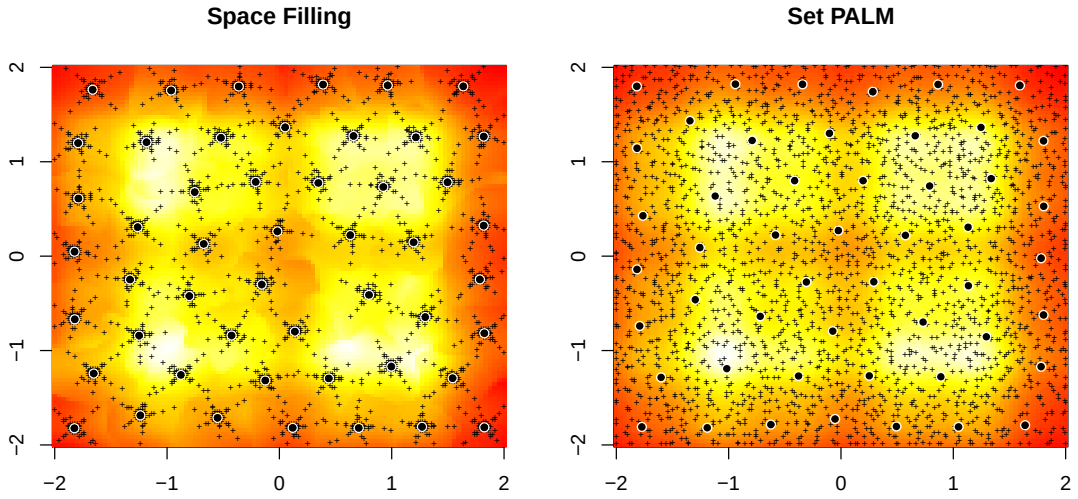

Figure 5.9: Left: Center locations and original design for a 40000 point stochastic draw from Herbie's tooth. Right: Change in design locations after `set_PALM` post processing.

Figure 5.9 shows the effect this post processing step has on the global PALM model. On the left is a space filling PALM created with 50 experts to model data sampled with $\sigma = 0.01$ from a 200×200 regularly spaced grid over $[-2, 2]^2$. Although computational resources should

be evenly allocated across the global surface, the selected data points are focused around the center locations. This leads to some drastic transitions between regions, which negatively affect the score of the model. On the right, observe that after post processing the selected models cover the input space evenly.

The result of spreading out the designs of each component of the global PALM is much improved mean estimation across the entire surface. The effect post processing has on RMSE will be examined in Table 5.5. Additionally, Figure 5.10 shows the effect `set_PALM` has on the variance surface. Both plots show variance surfaces from the experiment described above; basic PALM on the left and sets on the right. For the purpose of visualization, both surfaces have been restricted to the same color palette for variance values. While not entirely flat, the post processing step has made the surface much smoother. When we believe the true surface to be homoskedastic, this greatly improves model performance.

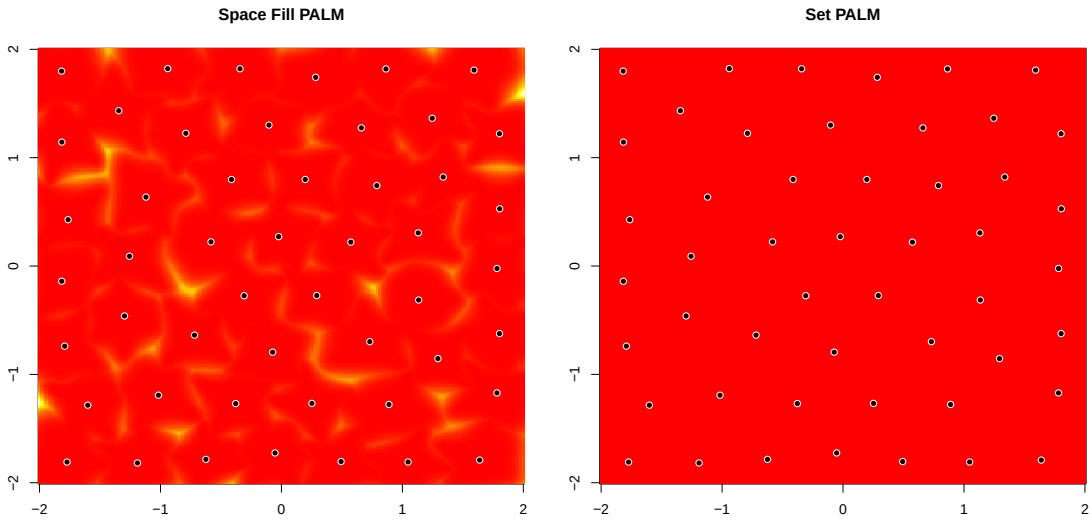

Figure 5.10: Left: Variance surface for a basic PALM on Herbie's Tooth. Right: The variance surface after `set_PALM` post processing pinned to the same color pallet.

Algorithm 5 lays out the process form post processing a PALM into a `set_PALM`. Start with an existing PALM model containing N_C local experts, as well as the data used to train it, (X, y). The model will be refit to be ideal for some predictive set, $X^\star$. In the absence of a

pre-existing set of predictive locations, use $X^\star = X$ to perform a general update of the model based on the observed data. First make predictions $X^\star$ to obtain the weight each model will receive at each location. Next, iterate through each model, select the set of points where the weight exceeds some predetermined threshold, and create a new LAGP model that lowers predictive variance over the entire set. For all examples in this paper the threshold is set to be $w_{\min\, min} = 0.9$.

Require: existing `PALM` model with N_C local experts, a matrix of n predictive locations $X^\star$, input locations X, response vector y, weight threshold $w_{\min}$

1: $\hat{W}_{n \times N_C} \leftarrow \text{weight}(\texttt{PALM}, X^\star)$ {Calculate the wight each model will recieve per location}
2: **for** i in 1 **to** N_C **do**
3: index = which$(\hat{W}[, i] > w_{\min})$ {Select the weights that exceed than the threshold}
4: $X_{\text{ref}} = X^\star[\text{index},]$ {Create a reference set for `laGP`}
5: $\texttt{PALM}_i \leftarrow \texttt{laGP}(X_{\text{ref}}, X, y, \Theta_i)$ {Replace the i^{th} expert using X_{ref} and the existing Θ_i}
6: **end for**

Algorithm 5: Algorithm for post processing a PALM into a `set_PALM`

As seen in Figures 5.9, and 5.10 this has a drastic effect when applied to PALM on large data sets, allowing a small number of models to effectively cover the entire input space. Figure 5.11 Shows how this effect can be applied to sequentially selected PALMs and maintain unequal distribution of computational resources. In both experiments, a sequentially selected PALM with 50 centers is used to model 40000 points sampled on a 200×200 grid with $\sigma = 0.01$. The resulting models have a higher density of local experts around the interesting area. When `set_PALM` is applied, we see generally wider spread of model points, that maintains higher density in the interesting area. This is because each model is adjusted to cover only the area where it received a large portion of weight in the original model. In areas with more densely packed centers, the models do not spread out as far.

The resulting models do not only visually cover the space better. The post processing leads to much better model performance. Table 5.5 shows the score and RMSE of all three examples before and after post processing. Notice the score increases substantially in all

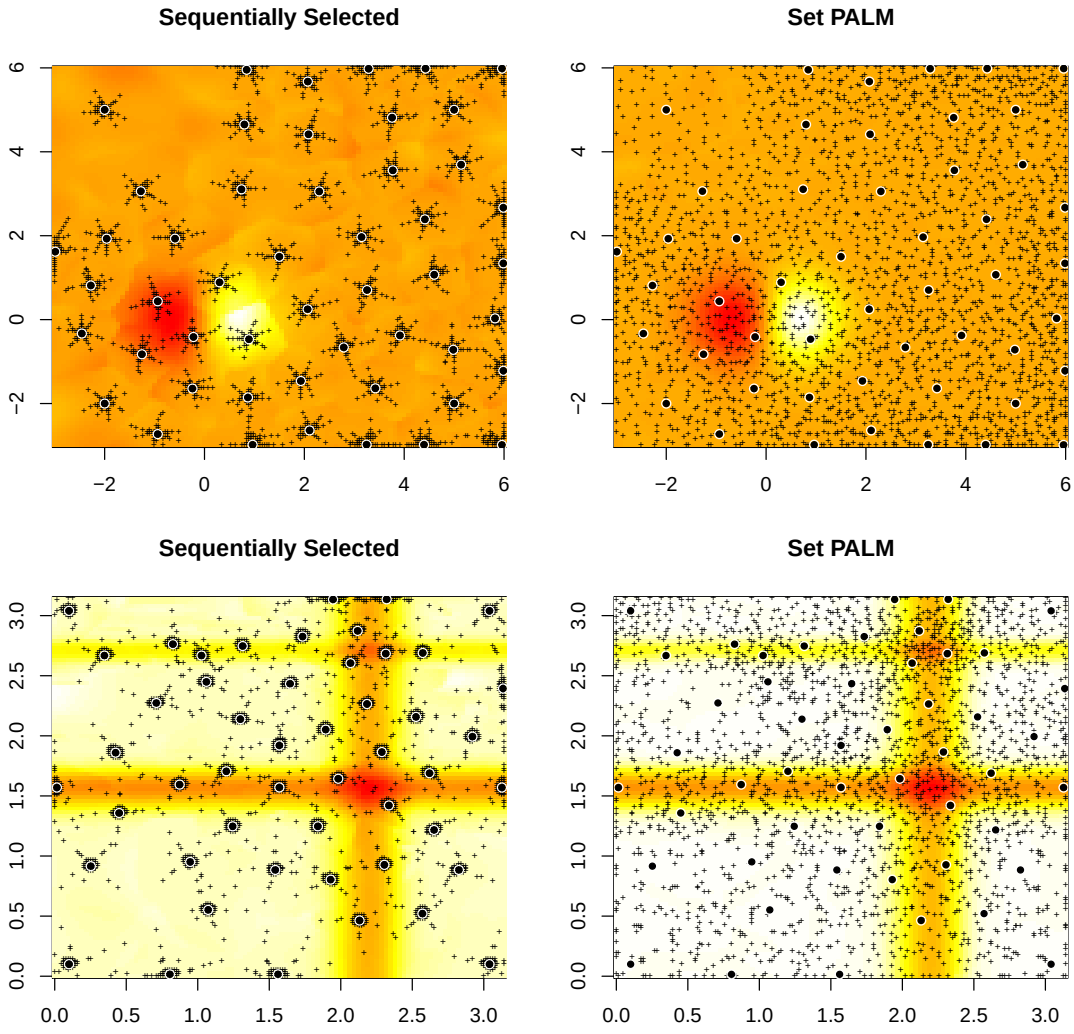

Figure 5.11: Top Row: PALMs on 40000 points from the Gramacy and Lee function sampled on $[-3, 6]^2]$ with $\sigma = 0.1$. Bottom Row: 40000 points from the 2D Michalewicz function with $\sigma = 0.1$. Left Column: sequentially selected PALM with 50 local experts. Right Column: PALM after `set_PALM` post processing.

three cases, and the RMSE is cut by half.

I now apply the set post processing to the motivating example of a 160000 point deterministically sampled, gridded draw from the Herbie's Tooth function. Figure 5.12 shows the effect of post processing on the "Large" data set from figure 5.6. Note that `set_PALM` has matched the spread of the correlation thinned data, but without the clear pattern in points

	Score		RMSE	
	Start	Set	Start	Set
Gramacy	6.9033	7.9951	0.0204	0.0111
Herbie	6.9162	8.0377	0.0246	0.0109
Michalewicz	6.5194	7.4421	0.0283	0.0133

Table 5.5: Comparing starting PALMs to their `set_PALM` equivalent on all of the previously used examples in this paper

kept.

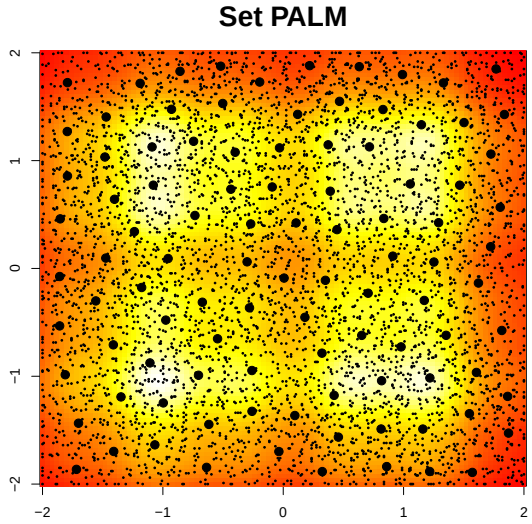

Figure 5.12: Local model point selection for `set_PALM` applied to a 160000 point gridded deterministic draw from Herbie's tooth.

The effect this step has on the score of the model is equally drastic. Table 5.6 shows that with access to the full data set, we can match, and even exceed the score of both the small, and thinned data sets by using the post processing step. The center column contains scores for a stochastic run of the experiment with $\sigma = 0.1$. The results are exactly the same as in the deterministic case, with score decreasing for increasing data size, recovered by thinning the data, and exceeded by `set_PALM`. Finally, in the last column we examine the effect for data that is uniformly sampled stochastically over the range $[-2, 2]^2$ (i.e. not on a regurlar grid). Here we see almost exactly the same results as in the gridded case.

	Deterministic	Stochastic	Non-Gridded
Small	13.74	7.97	7.95
Medium	12.09	7.63	7.66
Large	9.47	6.96	6.99
Thinned	14.02	8.02	7.99
Set	14.98	8.07	8.04

Table 5.6: Comparing Set PALM to other fitting schemes. All fitted models in the deterministic and stochastic case are trained on subsets of the 'Large' training data. The Non-gridded small medium and large sets are uniformly generated with sizes of 10, 40, and 160 thousand data points.

For a final example of post processing on PALM performance we return to the satellite temperature data set of Heaton et al. [24]. I applied the post processing exactly as laid out to both versions of PALM previously mentioned. Table 5.7 shows the results. Unfortunately, `set_PALM` post processing does not appear to be well suited to examples where the predictive set is not well surrounded by observed data points.

	score	crps	int	RMSE	MAE	CVG	Time.elapsed
PALM	-2.81	1.15	11.51	1.93	1.59	0.79	5.79
Set PALM	-3.21	1.25	13.87	2.07	1.71	0.75	5.62
Global PALM	-2.38	1.04	9.19	1.78	1.45	0.85	5.78
Global Set PALM	-2.85	1.19	11.38	1.98	1.65	0.79	5.62

Table 5.7: `set_PALM` applied to the satellite temperature benchmark data set. As previously, both versions built on the raw data, and residuals from a small global GP are shown.

To examine why the models perform worse after post processing, observe Figure 5.13. Here a single reference point is shown in white, while a larger reference set is marked by blue hash marks. The points that are chosen by only the single point model are also shown in white, and the points chosen only by `set_PALM` are shown in blue. Training locations chosen by both models are marked in red. In the case where there are no observed data interior to the predictive location, the effect of a larger reference set is to greatly increase the number of radial points.

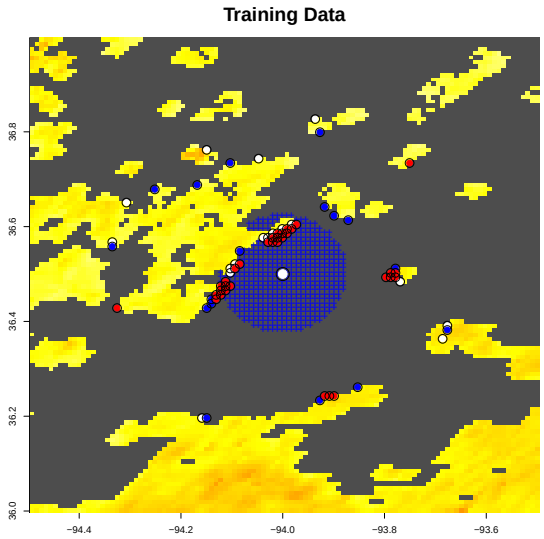

Figure 5.13

5.3 Non-GP PALM

5.3.1 Linear Models

As alluded to earlier, the PALM framework still works when applied to models other than GPs. Figure 5.14 shows one example to apply the PALM methods to combinations of polynomial linear models. On the left is the surface to be predicted, which is generated on X values uniformly sampled over the range $[0, 20]$. The mean surface is a combination of 10 randomly generated sine waves, $y(x) = \sum_{i=1}^{10} \mathcal{A}_i \sin\{\omega_i(x + \xi_i)\}$, with amplitude, $\mathcal{A}$, and period, ω uniformly generated on $[0, 5]$, and offset, ξ, generated on $[0, 2\pi]$. Observations are drawn independently from the resulting curve with $\sigma = 1$.

To model this surface non-parametrically, I first sorted the X locations, and split them into 39 overlapping sections of 50 data points. Thus the X_i contains the sorted observations indexed $(25(i-1)+1)$ to $(25i+25)$. This creates overlapping models, as all the data points except the first and last 25 are contained in 2 models. Each X_i is modeled by a local

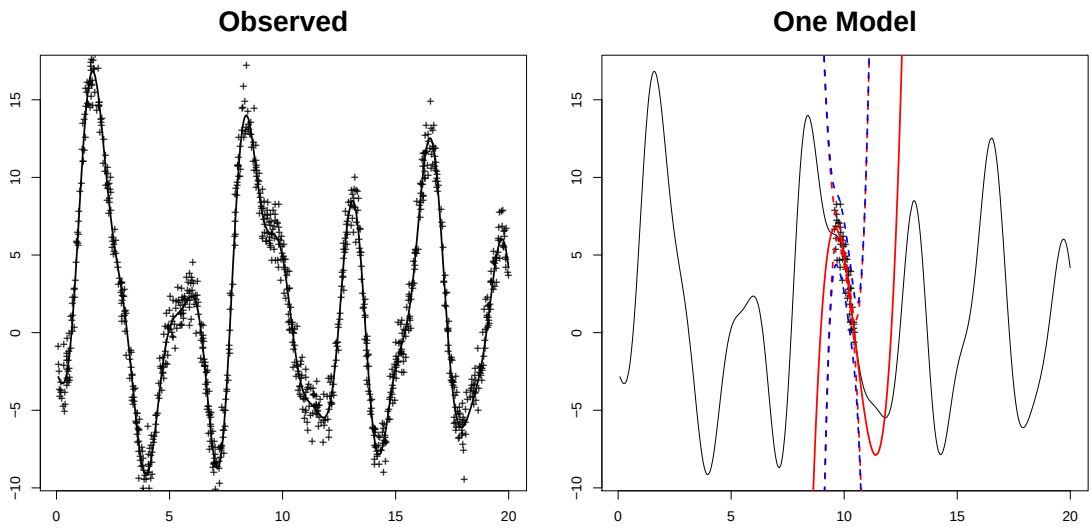

Figure 5.14: Left: A true mean surface created by the linear combination of 10 randomly generated sine waves. Observed points generated with $\sigma = 1$, and marked by '+'. Right: One model ($y = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3$), built on 50 observed points.

polynomial regression, $\hat{y}_i = b_{0i} + b_{1i}X_i + b_{2i}X_i^2 + b_{3i}X_i^3$. One such model is shown in the right panel of Figure 5.14. The estimated mean surface is shown as the solid red line, with two variance surfaces (mean in red, and prediction in blue).

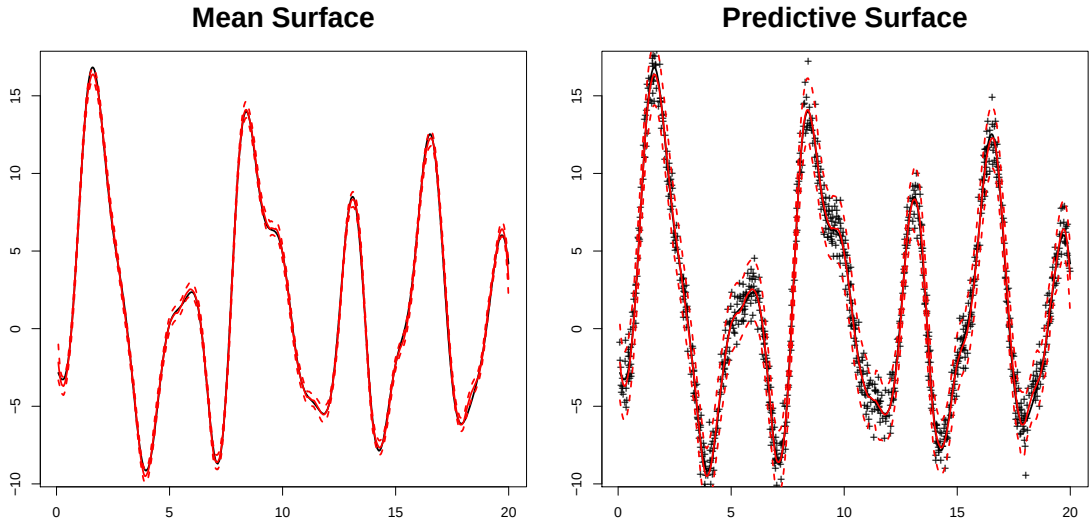

Figure 5.15: Left: The mean surface generated by lmPALM. 95% confidence intervals indicated by the dashed line. Right: Predictive surface. Observed data indicated by '+'. 95% prediction intervals shown by a dashed line.

Variance estimates for mean and prediction come directly from the linear models $\hat{\sigma}_i^2(x^\star(X_i^T X_i)^{-1}(x^\star)^T)$, and $\hat{\sigma}_i^2(1 + x^\star(X_i^T X_i)^{-1}(x^\star)^T)$ respectively. Correlation estimates are calculated as $\max(0, \hat{\rho}_{kj})$, with $\hat{\rho}_{kj}$ as described in section 3.5, using the observed correlation from mean predictions of each model on the training set of the both models. The intuition here is that if the models disagree (i.e. negative correlation), this is probably coincidental, but a positive correlation in overlapping areas should indicate shared knowledge. Weights are calculated using the mean variance.

Figure 5.15 shows the resulting mean and prediction intervals. On the left the mean surface is shown by the solid red line with the dashed lines indicating a 95% confidence interval for mean. The right panel shows the equivalent for prediction intervals, with the observed data shown as small hatch marks. For this example, `lmPALM` is able to capture the mean and cover the predictions well.

5.3.2 Heteroskedastic PALM

Stable variance prediction coming from the linear models allows us to account for some heteroskedasticity with only minor changes to the existing framework. To see how effective this can be, we return to the motorcycle data mentioned in Section 2. The left panel of Figure 5.16 shows four models used to model separate regimes of the acceleration surface. Times from $t \in [0, 15)$ are modeled using $\hat{y} = b_0 + b_1 t$. We cannot use an intercept only model, as the variance would not increase outside the observed range. On $t \in [14, 34)$, a third order polynomial is used. The last two intervals, $t \in [32, 42)$ and $t \in [40, 57.6]$ use quadratic approximations. The resulting models are shown in various colors with solid lines, while the bounds on mean prediction are shown by the dashed lines.

The issue with the current formulation is shown my the middle panel. Mean surface and

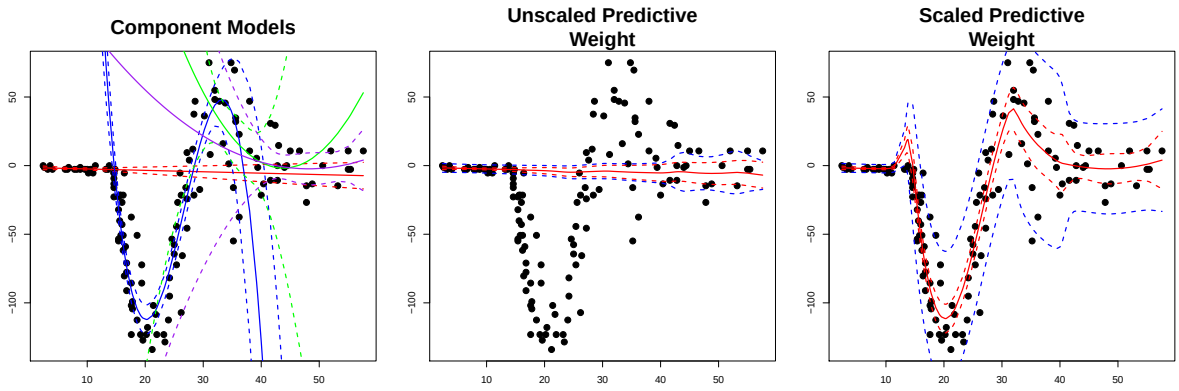

Figure 5.16: Left: Four models used to model pieces of the motorcycle data by lmPALM. Middle: A PALM created by unscaled variance weights from the models at left. Right: Scaled variance weights used to create a heteroskedastic lmPALM.

confidence intervals are shown by the solid and dashed red lines respectively. Prediction intervals are marked by the dashed blue line. The low variance regime on the left receives a significant portion of $w_k(x) = \phi_k(x)^p / \sum_{\ell=1}^{K} \phi_\ell(x)^p$ across the entire surface. To remedy this, I propose replacing ϕ_k with $\sigma_k \phi_k$, where σ_k is an estimate of minimum standard deviation from the model. In this case σ_k is the RMSE from the trained linear model, but in the GP case, this could be $\tau_k^2 \eta_k$. This can be interpreted as how sure the model is about prediction, compared to the highest certainty area. Note that this does not change the weight at all in the homoskedastic case, as all precisions will be multiplied by the same σ.

The result this change has on the resulting PALM is drastic. The right panel of Figure 5.16 shows the same four models from the left panel combined using the new weighting scheme. The mean surface, confidence interval, and prediction interval all match the local models well enough to predict the entire surface.

To observe this change in a case where we know the true surface to be modeled, I repeat the random sine wave experiment with variance increasing from a center uniformly drawn from $[0, 20]$ following $\sigma_i = 0.02(x_i - x_c)^2$, where x_c is the location of the randomly drawn center. The left panel of Figure 5.17 shows the observed data. The local models are built

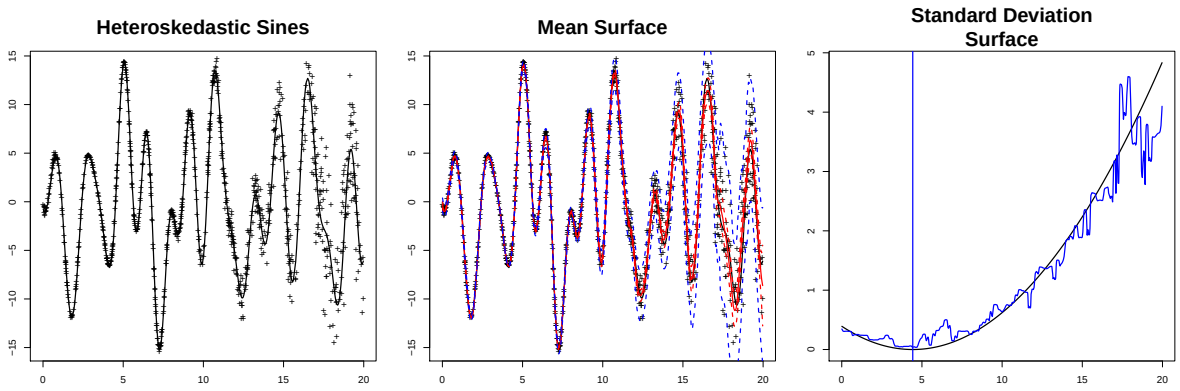

Figure 5.17: Left: Observed data from a linear combination of 10 random sine waves with heteroskedastic variance increasing quadratically from a randomly selected center point. Middle: Predictive surfaces created by lmPALM. Mean confidence intervals are shown in red, while predictive surfaces are shown in blue. Right: The variance surface from lmPALM shown with the true variance surface. The vertical blue line shows where the true center of the increasing variance lies.

exactly as described before, and combined using the new weighting scheme. The resulting model (shown in the middle panel) still captures the mean surface very well, while managing to account for the change in variance regime. The right panel shows the true standard deviation surface in black, and the estimated standard deviation in blue. The vertical blue line indicates the location of the randomly selected center.

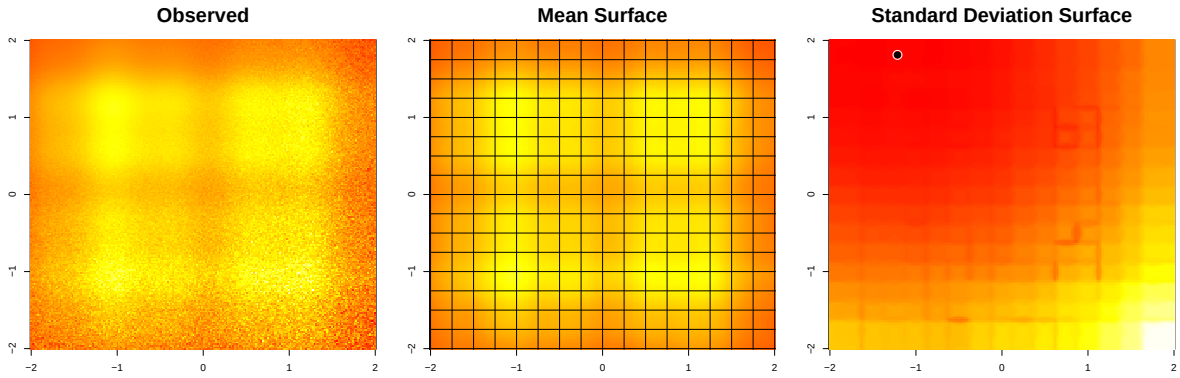

Figure 5.18: Left: Observed data from heteroskedastic herbies tooth with quadratically increasing variance emanating from a randomly selected central location. Modeled 225 lmPALM experts. Right: Observed data. Middle: Predictive surface created by lmPALM. Right: The standard deviation surface from lmPALM. The point indicates the true minimal variance point.

For a final example of heteroskedasticity in the PALM framework, I return to Herbie's Tooth (3.2). The left panel of Figure 5.18 shows observed data generated on a 200×200 regular grid with $\sigma = 0.1 \cdot |x - \text{center}|^2$ with a central location uniformly selected from the domain. Models are built overlapping from evenly divided 16 evenly divided regions in both X dimensions. In the middle panel, each group of 4 squares represents the domain of 1 of 256 local experts. Each local model is represented by an expanded second order polynomial model, $\hat{y}_i = b_0 + b_1 x_1, + b_2 x_2, + b_3 x_1^2 + b_4 x_2^2 + b_5 x_1 x_2$. The estimated standard deviation surface is shown in the right panel, with the true center of the quadratic increase represented by a black dot. In this case, heteroskedastic `lmPALM` has captured both the mean and the variance well.

Chapter 6

Conclusion and Future Work

In this dissertation I have provided a unified framework for both partition and aggregate methods for precision aggregated local models (PALM) featuring Gaussian Processes (GPs) as a showcase. The PALM framework suggest a dynamic weighting scheme for locally focused models as part of a global amalgam, with correlation estimates for accurate uncertainty quantification in prediction. I suggest a sequential selection algorithm for improving computational resource allocation in the PALM surface to mirror functional complexity. Finally I provide several variations on the "base" PALM framework including re-optimizing center location, local models designed for predictive sets, PALM with non-GP component models, and some allowance for smoothly transitioning heteroskedasticity.

The PALM framework is shown to be faster than many state-of-the-art methods while maintaining competitive predictive accuracy. For example, compared to LAGP, PALM is able to eliminate model build time to make comparable, or even better predictions in a fraction of the time. The independent nature of the component models prior to prediction combination supports massive parallelization in both the estimation and prediction phases that could provide additional speed advantages, or support additional, or more complex component experts. This can be viewed as a new area of research both for GP methods, as well as the wider field of non-parametric statistics.

6.1 Extensions

The first, and most obvious extension is applying the heteroskedastic advances of `lmPALM` (Section 5.3) to the GP methodology. The major hurdles to this step involve the somewhat erratic behavior of LAGP subset selection. Current methods smooth the variance by applying both a minimum from aggregated MSE of the component models, and applying a uniform maximum variance of Var(y), the variance of the observed response values. In simple cases, it may be possible to use the LAGP estimates of both hyperparameters, but for most examples, a new smoothing method will be required.

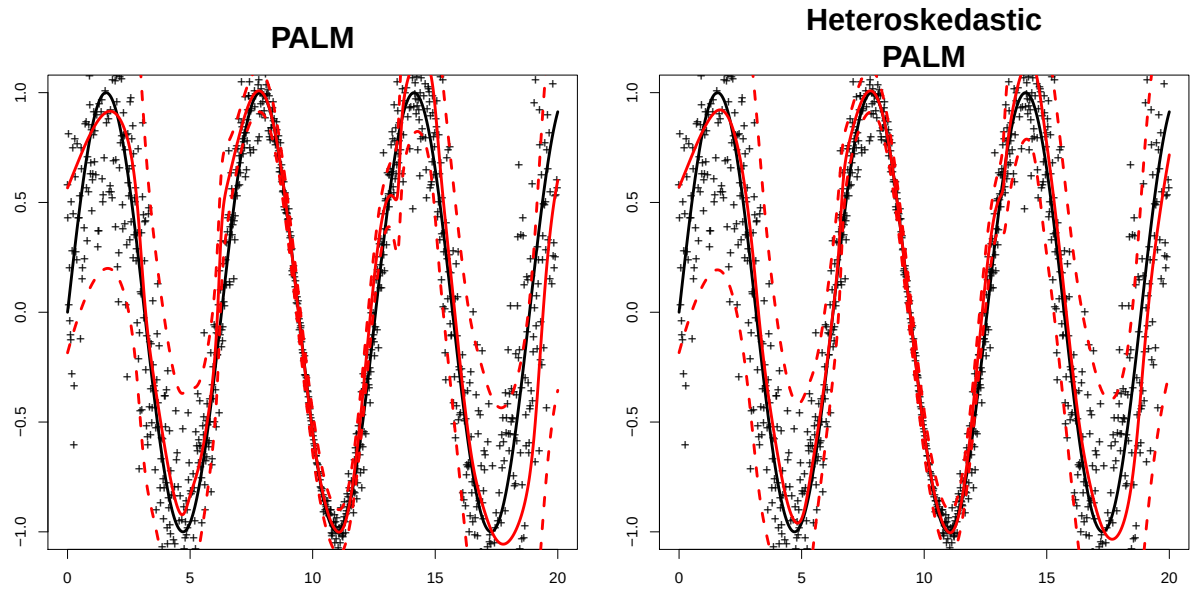

Figure 6.1: Left: Heteroskedastic PALM under powered inverse precision weights. Right: Heteroskedastic PALM using weights scaled by $\sqrt{\tau^2\eta}$

Figure 6.1 shows two PALMs where homoskedastic conditions are not enforced. Both panels are comprised of the same 10 local experts centered at each odd number. The observed values are generated from a basic sine wave with $\sigma = 0.05 \cdot |10 - x|$. On the left, each LAGP local expert is allowed to estimate its own hyperparameters, and is then combined using $w_i = \phi_i^p / \sum \phi_k^p$, to define w_i, the predictive weight for model i. The right panel

uses $w_i = (\phi_i)^p(\tau_i^2\eta_i)^{p/2}/\sum(\phi_k)^p(\tau_k^2\eta_k)^{p/2}$, where τ_i^2 and η_i are the amplitude and nugget parameters for the i^{th} GP model. The two surfaces are very similar, except at the transition points between models. The adjusted weight version is much smoother between models. In this well behaved example, no additional smoothing is necessary, but for real data, some smoothing will probably be necessary.

Additional Future work revolves around establishing a more rigorous basis for many PALM heuristics. Despite the good intuition surrounding the powered weights (Section 3.4), there may be room for improvement. Currently there exists functionality in the Gramacy Lab git repository (`https://bitbucket.org/gramacylab/lagp/src/master/R/boosting`) to optimize the power given an existing PALM and a predictive set. The forthcoming R package (described below) uses NLOPT_LN_COBYLA from the nloptr package to optimize p in $w_i = \phi_i^p/\sum\phi_k^p$, but no closed form solution for a "best" power exists.

It is also currently unknown how many local experts should be used in building a PALM. Examples in this paper were not truly subject to computational or time resources. Developing a cost function for predictive accuracy versus predictive time may help make the number of component models that comprise a global PALM less arbitrary on a case by case basis. The current sequential selection methods maximize predictive accuracy given a set computational budget, but there is no existing method to estimate how much increasing the computational budget will improve PALM performance.

Finally in the several PALM variations there is much left to explore. There should be a formally defined stopping criteria for the cycling algorithm (3). It has also been established that different types of component models cna comprise a PALM. Some exploration into the best types of locally accurate models for a global PALM surface should be conducted.

6.2 R Package

All code to fit PALM models and reproduce the examples shown here is currently available on the Gramacy Lab `laGP` git repository at https://bitbucket.org/gramacylab/lagp/src/master/R/boosting. To make these methods more accessible to the wider body of researchers I plan to write an R package that encompasses the relevant functions.

This will involve significant improvement to the current implementation of PALM. Current implementation involves obtaining the hyperparameters of the fit from the `aGP` function in the `lagp` package, then using those to fit a model that can be accessed in R. The methods elaborated in this dissertation are largely parallelizable, only the initial `aGP` fits of the center locations are currently taking advantage of parallel processing power. The finished package should eliminate computational redundancies, employ efficient matrix libraries, and parallelize computations where possible using R's doParallel and foreach packages.

Bibliography

[1] Binois, M., Gramacy, R. B., and Ludkovski, M. (2018). "Practical heteroscedastic gaussian process modeling for large simulation experiments." *Journal of Computational and Graphical Statistics*, 27, 4, 808–821.

[2] Bradley, J. R., Cressie, N., and Shi, T. (2016). "A comparison of spatial predictors when datasets could be very large." *Statist. Surv.*, 10, 100–131.

[3] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Belmont, CA: Wadsworth.

[4] Chen, T. and Ren, J. (2009). "Bagging for Gaussian process regression." *Neurocomputing*, 72, 7, 1605–1610.

[5] Chipman, H., George, E., and McCulloch, R. (2010). "BART: Bayesian additive regression trees." *The Annals of Applied Statistics*, 4, 1, 266–298.

[6] Chipman, H. A., George, E. I., and McCulloch, R. E. (1998). "Bayesian CART Model Search." *Journal of the American Statistical Association*, 93, 443, 935–948.

[7] Cochran, W. G. (1954). "The Combination of Estimates from Different Experiments." *Biometrics*, 10, 1, 101–129.

[8] Cohn, D. (1994). "Neural network exploration using optimal experiment design." In *Advances in neural information processing systems*, 679–686.

[9] Cressie, N. (1991). *Statistics for Spatial Data,* revised edition. John Wiley and Sons, Inc.

[10] Das, K. and Srivastava, A. N. (2010). "Block-GP: Scalable Gaussian Process Regression for Multimodal Data." In *2010 IEEE International Conference on Data Mining*, 791–796.

[11] Datta, A., Banerjee, S., Finley, A. O., and Gelfand, A. E. (2016). "Hierarchical Nearest-Neighbor Gaussian Process Models for Large Geostatistical Datasets." *Journal of the American Statistical Association*, 111, 514, 800–812.

[12] Datta, A., Banerjee, S., Finley, A. O., Hamm, N. A. S., and Schaap, M. (2015). "Non-separable Dynamic Nearest-Neighbor Gaussian Process Models for Large spatio-temporal Data With an Application to Particulate Matter Analysis." *arXiv:1510.07130 [stat]*. ArXiv: 1510.07130.

[13] Friedman, J. H. (2001). "Greedy function approximation: A gradient boosting machine." *Ann. Statist.*, 29, 5, 1189–1232. Publisher: Institute of Mathematical Statistics.

[14] Fuentes, M. (2001). "A high frequency kriging approach for non-stationary environmental processes." *Environmetrics*, 12, 5, 469–483. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/env.473.

[15] Furrer, R., Genton, M., and Nychka, D. (2006). "Covariance Tapering for Interpolation of Large Spatial Datasets." *Journal of Computational and Graphical Statistics*, 15, 502–523.

[16] Gneiting, T. (2002). "Compactly Supported Correlation Functions." *Journal of Multivariate Analysis*, 83, 2, 493–508.

[17] Gneiting, T. and Raftery, A. (2007). "Strictly Proper Scoring Rules, Prediction, and Estimation." *Journal of the American Statistical Association*, 102, 477, 359–378.

[18] Goldberg, P. W., Williams, C. K., and Bishop, C. M. (1998). "Regression with input-dependent noise: A Gaussian process treatment." In *Advances in Neural Information Processing Systems*, vol. 10, 493–499. Cambridge, MA: MIT press.

[19] Gramacy, R. and Lee, H. (2008). "Bayesian treed Gaussian process models with an application to computer modeling." *Journal of the American Statistical Association*, 103, 483, 1119–1130.

[20] Gramacy, R., Niemi, J., and Weiss, R. (2014). "Massively parallel approximate Gaussian process regression." *SIAM/ASA Journal on Uncertainty Quantification*, 2, 1, 564–584.

[21] Gramacy, R. B., Lee, H. K. H., and Macready, W. G. (2004). "Parameter space exploration with Gaussian process trees." *Proceedings of the International Conference on Machine Learning*, 353–360.

[22] Green, P. J. and Sibson, R. (1978). "Computing Dirichlet Tessellations in the Plane." *The Computer Journal*, 21, 2, 168–173.

[23] Heaton, M. J., Datta, A., Finley, A., Furrer, R., Guhaniyogi, R., Gerber, F., Gramacy, R. B., Hammerling, D., Katzfuss, M., Lindgren, F., Nychka, D. W., Sun, F., and Zammit-Mangion, A. (2017). "A Case Study Competition Among Methods for Analyzing Large Spatial Data." *arXiv:1710.05013 [stat]*. ArXiv: 1710.05013.

[24] Heaton, M. J., Datta, A., Finley, A. O., Furrer, R., Guinness, J., Guhaniyogi, R., Gerber, F., Gramacy, R. B., Hammerling, D., Katzfuss, M., et al. (2019). "A case study competition among methods for analyzing large spatial data." *Journal of Agricultural, Biological and Environmental Statistics*, 24, 3, 398–425.

[25] Johnson, M., Moore, L., and Ylvisaker, D. (1990). "Minimax and maximin distance designs." *Journal of statistical planning and inference*, 26, 2, 131–148.

[26] Kaufman, C. G., Bingham, D., Habib, S., Heitmann, K., and Frieman, J. A. (2011). "Efficient emulators of computer experiments using compactly supported correlation functions, with an application to cosmology." *The Annals of Applied Statistics*, 5, 4, 2470–2492. ArXiv: 1107.0749.

[27] Kaufman, C. G., Schervish, M. J., and Nychka, D. W. (2008). "Covariance Tapering for Likelihood-Based Estimation in Large Spatial Data Sets." *Journal of the American Statistical Association*, 103, 484, 1545–1555.

[28] Kim, H., Mallick, B., and Holmes, C. (2005). "Analyzing nonstationary spatial data using piecewise Gaussian processes." *Journal of the American Statistical Association*, 100, 470, 653–668.

[29] Lee, H., Gramacy, R., Linkletter, C., and Gray, G. (2011). "Optimization subject to hidden constraints via statistical emulation." *Pacific Journal of Optimization*, 7, 3, 467–478.

[30] Lloyd, S. P. (1982). "Least squares quantization in PCM." *IEEE Transactions on Information Theory*, 28, 2, 129–137.

[31] Ma, P., Kang, E. L., Braverman, A., and Nguyan, H. (2019). "Spatial Statistical Downscaling for Constructing High-Resolution Nature Runs in Global Observing System Simulation Experiments." *Technometrics*, 61, 3, 322–340.

[32] Molga, M. and Smutnicki, C. (2005). "Test Functions for Optimization Needs."

[33] Park, C. and Apley, D. (2017). "Patchwork Kriging for Large-scale Gaussian Process Regression." *arXiv:1701.06655 [cs, stat]*. ArXiv: 1701.06655.

[34] Park, C. and Huang, J. Z. (2016). "Efficient Computation of Gaussian Process Regres-

sion for Large Spatial Data Sets by Patching Local Gaussian Processes." *Journal of Machine Learning Research*, 17, 174, 1–29.

[35] Park, C., Huang, J. Z., and Ding, Y. (2011). "Domain Decomposition Approach for Fast Gaussian Process Regression of Large Spatial Data Sets." *Journal of Machine Learning Research*, 12, May, 1697–1728.

[36] Rasmussen, C. E. and Ghahramani, Z. (2002). "Infinite Mixtures of Gaussian Process Experts." 8.

[37] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

[38] Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). "Design and Analysis of Computer Experiments." *Statistical Science*, 4, 409–435.

[39] Santner, T. J., Williams, B. J., and Notz, W. I. (2003). *The Design and Analysis of Computer Experiments*. New York, NY: Springer-Verlag.

[40] Seeger, M., Williams, C. K. I., and Lawrence, N. D. (2003). "Fast Forward Selection to Speed Up Sparse Gaussian Process Regression." In *In Workshop on Ai and Statistics 9*.

[41] Smola, A. J. and Bartlett, P. L. (2001). "Sparse Greedy Gaussian Process Regression." 7.

[42] Snelson, E. and Ghahramani, Z. (2006). "Sparse Gaussian Processes using Pseudo-inputs." In *Advances in Neural Information Processing Systems*, 1257–1264. MIT press.

[43] — (2007). "Local and global sparse Gaussian process approximations." In *Artificial Intelligence and Statistics*, 524–531.

[44] Stein, M. L. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Science & Business Media.

[45] Stein, M. L., Chi, Z., and Welty, L. J. (2004). "Approximating Likelihoods for Large Spatial Data Sets." *Journal of the Royal Statistical Society, Series B*, 66, 2, 275–296.

[46] Sun, F., Gramacy, R. B., Haaland, B., Lawrence, E., and Walker, A. (2019). "Emulating satellite drag from large simulation experiments." *SIAM/ASA Journal on Uncertainty Quantification*, 7, 2, 720–759.

[47] Tresp, V. (2000). "A Bayesian Committee Machine." *Neural Computation*, 12, 11, 2719–2741.

[48] Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science & business media.

[49] Vecchia, A. V. (1988). "Estimation and Model Identification for Continuous Spatial Processes." *Journal of the Royal Statistical Society. Series B (Methodological)*, 50, 2, 297–312. Publisher: [Royal Statistical Society, Wiley].

[50] Zhang, B., Cole, D. A., and Gramacy, R. B. (2019). "Distance-distributed design for Gaussian process surrogates." *arXiv:1812.02794 [stat]*. ArXiv: 1812.02794.