Block-GP: Scalable Gaussian Process Regression for Multimodal Data

Kamalika Das
SGT Inc at NASA Ames Research Center
kamalika.das@nasa.gov

Ashok N. Srivastava
NASA Ames Research Center
ashok.n.srivastava@nasa.gov

***Abstract*—Regression problems on massive data sets are ubiquitous in many application domains including the Internet, earth and space sciences, and finances. In many cases, regression algorithms such as linear regression or neural networks attempt to fit the target variable as a function of the input variables without regard to the underlying joint distribution of the variables. As a result, these global models are not sensitive to variations in the local structure of the input space. Several algorithms, including the mixture of experts model, classification and regression trees (CART), and others have been developed, motivated by the fact that a variability in the local distribution of inputs may be reflective of a significant change in the target variable. While these methods can handle the non-stationarity in the relationships to varying degrees, they are often not scalable and, therefore, not used in large scale data mining applications. In this paper we develop Block-GP, a Gaussian Process regression framework for multimodal data, that can be an order of magnitude more scalable than existing state-of-the-art nonlinear regression algorithms. The framework builds local Gaussian Processes on semantically meaningful partitions of the data and provides higher prediction accuracy than a single global model with very high confidence. The method relies on approximating the covariance matrix of the entire input space by smaller covariance matrices that can be modeled independently, and can therefore be parallelized for faster execution. Theoretical analysis and empirical studies on various synthetic and real data sets show high accuracy and scalability of Block-GP compared to existing nonlinear regression techniques.**

***Keywords*-Gaussian Process, regression, parallel computation**

I. Introduction

In many application domains, it is important to predict the value of one feature based on certain other measured features. For example, in the Earth Sciences, predicting the precipitation at one location given the humidity, sea surface temperature, cloud cover, and other related factors is an important step in climate modeling. For such problems, simple linear regression based on minimizing the mean squared error between the true and predicted values can be used for modeling the relationship between the input and the target features.

In decision support systems which use these predictive algorithms, a prediction with low confidence may be treated differently than if the same prediction was given with high-confidence. Thus, while the predicted value from the regression function is clearly important, the *confidence* in the prediction is equally important. A simple model such as linear regression does not provide us with that information. Also, models like linear regression, in spite of being easy to fit and being highly scalable, fail to capture nonlinear relationships in the data. Gaussian Process regression is one regression model that can capture nonlinear relationships and outputs a distribution of the prediction where the variance of the predicted distribution acts as a measure of confidence in the prediction.

Another issue that arises during such regression based modeling of the data is that often the joint distribution of the input variables and the target variable is not homogeneous. Thus, changes in the local structure of the input variables may lead to dramatic changes in the value of the target variable. Considering our Earth Science example from before, the model for prediction of precipitation would be very different for desert regions compared with the rainforest region. Thus, having a single, global model in these situations would not work as accurately as developing several local models. This potential non-stationary relationship between the input variables and the target variable has led to numerous innovations in regression modeling such as CART, neural networks, piecewise-linear models, mixture of experts, and others over the last three decades. While these methods can handle the non-stationarity in the relationships to varying degrees, they are often not scalable and therefore, not used in large scale data mining applications.

In this paper we propose a novel Gaussian Process regression method that takes into account the multimodal (non-stationary) nature of the joint distribution of the input and the target. Efficient identification of the modes (or clusters) can lead to a significant improvement in prediction accuracy and prediction confidence due to noise suppression. The innovation builds on the fact that the joint distribution of inputs and outputs may likely have regions which have higher intra-mode covariation compared with the inter-mode covariation. The decomposition of the solution to build multiple local models for the different modes also results in high scalability of the method. The approximation introduced by way of the decomposition is compensated for by the introduction of a 'complement set' for modeling noise in the data that improves the accuracy of the algorithm at minimal cost.

II. Background: Gaussian Process regression

Rasmussen and Williams [1] provide an excellent introduction on Gaussian Process regression. The Gaussian Process regression is a generalization of standard linear regression. If $\mathbf{X}$ is the training data set having n multidimensional observations (rows) $\mathbf{x}_1, \ldots, \mathbf{x}_n$, with each $\mathbf{x}_i \in \mathbb{R}^D$ and the corresponding target is represented by a $n \times 1$ vector $\mathbf{y}$, then the standard linear regression model is: $f(\mathbf{x}) = \mathbf{x}\mathbf{w}^T$ and $y = f(\mathbf{x}) + \epsilon$ where $\mathbf{w}$ is a D-dimensional weight vector of parameters and ϵ is additive Gaussian noise such that $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The predictive mean and variance equations for Gaussian Process regression are

$$\widehat{y}^* = K^*(\sigma^2 I + K)^{-1}\mathbf{y} \quad (1)$$

$$C = K^{**} - K^*(\sigma^2 I + K)^{-1}K^{*T} \quad (2)$$

where the ij^{th} entry of K is $k(\mathbf{x}_i, \mathbf{x}_j)$ and K^* and K^{**} are similarly the cross covariance matrices involving the test point $\mathbf{x}^*$. Equations 1 and 2 pose significant computational challenge due to the requirement of inverting the covariance matrix K of size n^2. If the number of observations n is large, the $O(n^3)$ operation can be a bottleneck in the process of using Gaussian Process regression.

Approximations are introduced in the Gaussian Process literature for either finding closed-form expressions for intractable posterior distributions or for gaining computational advantage for large data sets. Smola and Bartlett [2] describe a sparse greedy method that finds an approximation to the maximum aposteriori estimate by selecting an 'active' subset of columns of K, thereby reducing the running time from $O(n^3)$ to $O(nm^2)$ where m ($m \ll n$) is the rank of the matrix approximation. Recent research by Foster et al. [3] has shown that if we use partial Cholesky decomposition to factorize the covariance matrix and perturb the low rank factor such that independent rows and columns form the principal sub-matrix, then the approximation we get is numerically stable. The generalized Bayesian committee machine [4] is another approximate technique that divides the data arbitrarily into M almost equal sized partitions and trains a different estimator on each partition.

For addressing non-stationarity or multimodality of data, the mixture of experts model for neural networks has been extended to a similar concept called mixture of Gaussian Process experts (MGP) in which individual experts model the local dependencies in the data. Tresp proposed the first MGP method where there are k experts for the k modes of the data [5] and the gating network uses another set of Gaussian Process classifiers to decide which test point belongs to which Gaussian Process expert. An alternative formulation for MGP [6] uses a Dirichlet process prior and the posterior is evaluated using MCMC and Gibbs sampling.

Most of the existing mixture of Gaussian Process experts methods suffer from very serious scalability issues due to delayed convergence of iterative procedures and tend not to scale beyond $n = 1000$. The numerical approximation techniques, on the other hand, fail to capture the multimodal nature of the training data. Assuming that multimodal data will have interspersed dense and sparse blocks within its covariance matrix, Block-GP tries to identify and separate the dense blocks from the rest of the (sparse) matrix and builds individual Gaussian Process experts for each block. It also approximately models the sparse blocks of the covariance matrix (not modeled by the experts) by an additional Gaussian Process expert. This is equivalent to approximating the low gating probabilities by zero without explicitly evaluating them using an expensive iterative algorithm. The Block-GP method can scale up to $k+1$ times the scalability of existing numerical approximation techniques with comparable or better accuracy.

III. Block-GP: algorithm description

In this section we describe the proposed algorithm Block-GP for Gaussian Process regression for multimodal data. We start with the intuition behind this algorithm. Let the input data set $\mathbf{X}$ be partitioned into k mutually exclusive groups. If the data is well separated such that covariance between the data points in the different partitions is equal to zero, then the covariance matrix K is block diagonal where K_i of size $n_i \times n_i$ denotes the covariance matrix of all the variables in the i-th partition $\mathbf{X}_i$. This leads to a block diagonal decomposition of $(K + \sigma^2 I)^{-1}$ that allows us to use k different GPs with noise parameters $\sigma_1, \ldots, \sigma_k$. Similarly, we can rewrite the test covariance K^* as $K^* = [K_1^* \quad \ldots \quad K_k^*]$ with K_i^* of size $n^* \times n_i$ denoting the covariance between $\mathbf{x}^*$ and the i^{th} partition $\mathbf{X}_i$. The target $\mathbf{y}$ has the same partitions as the input $\mathbf{X}$. Therefore, $\mathbf{y} = [\mathbf{y}_1 \quad \ldots \quad \mathbf{y}_k]$. Using Equation 1 and the decomposition of K^* and $\mathbf{y}$ we can say that if the test point $\mathbf{x}^*$ is best described by the data in block i, then we can predict y^* using $\widehat{y}^* = K_i^*(K_i + \sigma_i^2 I)^{-1}\mathbf{y}_i$. If we know that a test point is modeled by this set of Gaussian Processes, but do not know which Gaussian Process block (expert) best predicts the test point, then we can modify the prediction equation so that the prediction is a weighted combination of the predictions of the individual experts given by $\widehat{y}^* = \sum_{i=1}^{k} h_i K_i^*(K_i + \sigma_i^2 I)^{-1}\mathbf{y}_i$ where h_i represents the weight of the prediction by the i^{th} expert. This weight can be determined by some *gating* function that assigns a probability of the test point y^* coming from data partition i. Now, if the data is not perfectly sparse, that is, if the off-block diagonal entries of the covariance matrix are not all equal to zero, then we incur some loss of information due to the above decomposition of the prediction equation. Non-zero off-block diagonal elements indicate inter-partition covariance which are not modeled by any of the experts individually. To deal with this scenario we propose to identify all data points that are not modeled well enough by a single Gaussian Process expert, and model them using

a separate expert. Next we describe our proposed Gaussian Process regression framework that consists of two steps, namely, (i) data partitioning and (ii) training. Algorithm 1 provides the pseudo code for the different phases of the Block-GP method.

Algorithm 1: Block-GP algorithm

Input: $\mathbf{X} \in \mathbb{R}^{n \times D}$, $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{x}^* \in \mathbb{R}^{1 \times D}$,$k$, σ, e_{th}, s, $k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathcal{H}$
Output: $\widehat{y}^*$, $Var_{\widehat{y}^*}$,$\mathcal{H}_{est}$
Step 1: Partition the data into k groups
$[\mathbf{X}_1, \ldots, \mathbf{X}_k]$ = spectral_nyström($\mathbf{X}, k, s, \sigma$);
Step 2: Train the Gaussian Process models
for *each data partition* $\mathbf{X}_i, (i = 1 : k)$ **do**
  $[\mu_i, \sigma_i]$=fitGaussian($\mathbf{X}_i$);
  for *each point* $\mathbf{x}_j \in \mathbf{X}_i$ **do**
    p_{ji} = computeProbability($\mu_i, \sigma_i, \mathbf{x}_j$);
    $e_{ji} = \sum_{i=1}^{k} p_i \log(p_i)$;
    if $e_{ji} > e_{th}$ **then** $\mathbf{X}_i^{'} \leftarrow \mathbf{X}_i \setminus \mathbf{x}_j \quad \mathbf{X}_{i+1}^{'} \leftarrow \mathbf{X}_{i+1}^{'} \bigcup \mathbf{x}_j$;
for *each data partition* $\mathbf{X}_i^{'}, (i = 1 : k + 1)$ **do**
  $[\mu_i, \sigma_i]$=fitGaussian($\mathbf{X}_i^{'}$);
for *each data partition* $\mathbf{X}_i^{'}, (i = 1 : k + 1)$ **do**
  $[\mathcal{H}_{est}]$=learnGP($\mathbf{X}_i^{'}, \mathbf{y}_i^{'}, \mathcal{H}$);

A. Data partitioning

Unlike the traditional mixture of Gaussian Process experts, we partition our data into k disjoint sets so that each expert is responsible for modeling the input-target relationship for only that set. We rely on spectral clustering of the input space for identifying blocks in the covariance matrix. Spectral clustering algorithms [7] cluster the data based on the spectral properties or eigen analysis of the similarity matrix S constructed on the data. Spectral clustering suffers from the same scalability issues as Gaussian Process regression due to the requirement of constructing the $n \times n$ similarity matrix. There exists several methods for improving the scalability of spectral clustering. We use Nyström approximation for spectral clustering [8] which computes the first k eigen vectors directly on a smaller subset s ($< n$) of the data points and then extrapolates it to the other $n - s$ values.

B. Training Block-GP

Once the data is partitioned into blocks, Block-GP goes into the training phase where three things are accomplished: (i) identify points in the different data partitions that are not modeled well enough by the distributions fitted to those blocks and create a complement set using those points, (ii) determine the distribution parameters for the data in each block for computing expert weights (h_i) for test points, and (iii) train a set of hyperparameters ($\mathcal{H}_{est}$) for each of the data partitions. For identifying points in the complement set, we fit a Gaussian probability distribution (denoted by $\mathcal{N}(\mu_i, \sigma_i)$) to each data partition and compute the membership probabilities p_{ji} of every point $\mathbf{x}_j$, $j = 1 \ldots n$ in the data set $\mathbf{X}$ for each distribution. We then compute the entropy e_{ji} of each point based on these probabilities and make a $k+1^{st}$ partition using all points that have $e_{ji} > e_{th}$ where the threshold entropy e_{th} is an input to the algorithm. Now we refit Gaussian distributions to the new partitions $\mathbf{X}_i^{'} \ldots \mathbf{X}_{k+1}^{'}$ so that we can compute the weights h_i for a given test point for each data partition during the test phase. Finally, training a Gaussian Process on each of these data blocks requires optimizing a set of hyperparameters $\mathcal{H}$ based on the chosen covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$ appropriate for the data set.

Given a test point $\mathbf{x}^*$, the task during the test phase is to compute the prediction of the corresponding targets y^*. For that we first compute the weights h_i for each Gaussian Process expert. We then compute the prediction mean and variance using the respective equations.

Based on our description of the Block-GP method, we have seen that the performance of Block-GP is dependent on two steps: (i) spectral clustering the data for identifying blocks in the sparse covariance matrix and ii) entropy-based identification of noisy points in the individual blocks for creation of a complement set. The covariance matrix K of a multimodal data set $\mathbf{X}$ having k modes has k non-sparse blocks and using a permutation matrix P on K it is possible to block diagonalize K. We claim that spectral clustering the data $\mathbf{X}$ also allows us to identify the blocks in K. The formal proof is omitted due to lack of space. Since we do not use an expectation maximization based algorithm for learning the *gating* function and use hard clustering for partitioning the data into disjoint sets, it is possible that certain elements in each cluster are not represented accurately by the representative statistics of that cluster. We claim (proof not shown in this paper) that identifying all points from the clusters with high entropy is equivalent to finding points in different data partitions that are have low posterior probability given the cluster. We create a complement set from all points in $\mathbf{X}$ that are not modeled by a single data partition. Since creation of the complement set helps model the effect of some of the non trivial off block diagonal elements, it also improves the prediction accuracy of our method compared to that of using only k experts. The computational cost of Block-GP only increases linearly with the number of data partitions and the number of training instances in each partition. Given a training set $\mathbf{X}$ of size $n \times D$, the Block-GP algorithm has a computational complexity of $O((k+1)n_{\max}D^2)$ for calculating the mean of the prediction where $k+1$ is the number of data partitions with sizes $n_1, \ldots, n_{k+1}$ and $n_{max} = \max\{n_1, \ldots, n_{k+1}\}$.

IV. Experimental results

In this section we present the results of the experiments conducted for Gaussian Process regression on different syn-

thetic and real-world data sets using Block-GP and some of the existing Gaussian Process regression techniques. We compare the accuracy of results obtained from these different methods using the metric normalized mean squared error, defined as: $NMSE = \frac{1}{n\sigma_{\mathbf{y}}^2}\sum_{i=1}^{q}(\hat{y}_i - y_i)^2$ where y_i is the observed value of the target $\mathbf{y}$ having variance $\sigma_{\mathbf{y}}^2$, $\hat{y}_i$ is the prediction of y_i and q is the size of the test set. The methods that we compare our method against are (i) standard Gaussian Process (GP) and (ii) the pivoting-based V-formulation technique (GP-V) proposed by Foster et al. [3]. Since GP-V is an approximation of GP, we only compare against GP-V when standard GP fails to scale to the size of the training data. We do not report the comparison results for Tresp's MGP technique [5] since the expectation maximization iterations fail to converge for most of the data set sizes that we use for demonstrating our results. For demonstrating scalability, we compare the running time of Block-GP for different sizes of training data.

Experimental setup: The GP and GP-V algorithms have been run in a centralized setting using a 64-bit 2.33 GHz quad core Dell Precision 690 desktop running Red Hat Enterprise Linux version 5.4 having 20GB of physical memory. The Block-GP algorithm is parallelizable and has been executed on a 64-bit Linux cluster consisting of 16 slave nodes where each node is a dual processor 1-U server containing two quad-core Intel Xeon 2.66GHz processors totaling 128 cores and 128GB Ram (1Gb/Core). All centralized algorithms are implemented and run in MATLAB R2010a [1]. The Block-GP code uses the Parallel Toolbox in MATLAB R2010a. Also, the individual Gaussian Processes for each partition of Block-GP uses the numerical approximation based GP-V method for maximum scalability.

Dataset description: We demonstrate our results on three different multimodal data sets. The first data set (tanh) is a one dimensional synthetic time series having 900 time points generated by

$$x_{t+1} = \begin{cases} 2(1-x_t^2) - 1 \text{ if } s = 1 \\ tanh(-1.2x_t^2 + \epsilon_{t+1}) \text{ if } s = 0 \text{ and } \epsilon \sim \mathcal{N}(0, 0.1) \end{cases}$$

and then time-embedded to create a three dimensional (two inputs, one target) autoregressive time series where the target is a function of the last two time points. This data set has two modes corresponding to the two functions. The second data set D (http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html#setD) is a computer generated univariate time series describing the motion of a damped driven particle in an asymmetric 4-dimensional four-well potential. It has also been time-embedded to create a two dimensional (one input one target) autoregressive time series having 100,000 observations. We use this data set since it has four modes corresponding to the four wells and predicting a test point means trying to predict the ball's position in one of the 4 wells given any time instance and given the training data describing its motion. The third data set (California) that we have used in our experiments is the MODerate-resolution Imaging Spectroradiometer (MODIS) data providing 500-meter surface reflectance data for the state of California adjusted using a bidirectional reflectance distribution function (BRDF). The data is collected at intervals of every 8 days and stored as 1203×738 image file. Each image data is recorded for seven different wavelengths corresponding to seven different channels. Since these channels observe the same spatial location at the same time instances, there is a high correlation among the different bandwidths. Therefore, Gaussian Process regression can be used to model the relationship between the channels for creating *Virtual* Sensors and detecting changes in land cover. Based on careful exploratory analysis and domain expert feedback, three features (Band1 620 - 670 nm, Band4 545 - 565 nm and Band5 1230 - 1250 nm) have been chosen to model the target (Band6 1628 - 1652 nm). Since surface reflectance is expected to be a function of the land cover of a region, we have chosen the number of modes in the data to be 10 based on the number of different landcover types in the state. [2]. The data set contains nine years worth of data (2001-2009) arranged at the top level by the number of years where each year contains forty six (collected every 8 days) images and has approximately 15 million observations.

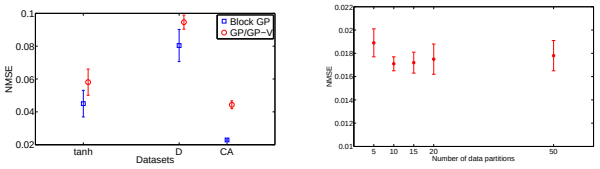
The covariance function used for all three data sets is a sum of two different covariance functions covSEard and covNoise defined as follows: *covSEard*: $k(x_p, x_q) = \sigma_f^2 \exp\left(-(x_p - x_q)P^{-1}(x_p - x_q)/2\right)$ where P is a diagonal matrix with automatic relevance determination (ARD) parameters $l_1^2, ..., l_D^2$ (D is the input dimension), and σ_f^2 is the signal variance. The hyperparameters are: $[\log(l_1), \log(l_2) \ldots \log(l_D), \log(\sigma_f)]$. *covNoise*: $k(x_p, x_q) = \sigma_n^2\delta(p, q)$ where σ_n^2 is the noise variance and $\delta(p, q)$ is a Kronecker delta function which is 1 iff $p = q$ and zero otherwise. The hyperparameter is $log(\sigma_n)$.

A. Accuracy

For our accuracy analysis, we report the results of two different experiments. The first experiment uses the three multimodal data sets tanh, D and California. For the tanh dataset we compare the results of Block-GP with standard GP. However, for data sets D and California, we cannot compute the covariance matrix K due to memory limitations and use GP-V as the baseline for comparison. Figure 1(a) shows the plot of the mean NMSE and the standard deviation of NMSE for these three data sets for 50 trials of the experiment. For each trial we take a random sample (75%) of the training examples to build the model and test it on the test data, which is fixed for all trials. We can notice in

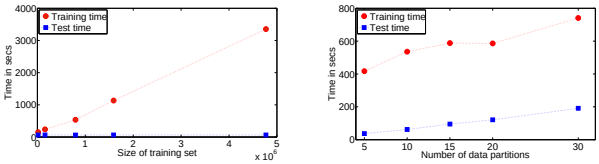
[1]For training hyperparameters we use minimize.m available at http://www.GaussianProcess.org/gpml/code A version of GP-V has been obtained from https://c3.ndc.nasa.gov/dl/algorithm/stableGP/

[2]http://casoilresource.lawr.ucdavis.edu/drupal/files/images/GIS_veg1.jpg



(a) NMSE for different data sets (b) NMSE for varying modes

Figure 1. Mean and standard deviation of NMSE of Block-GP for different data sets and data partitions



(a) Increasing training data size (b) Increasing number of modes

Figure 2. Scalability results demonstrated on the California data set

Figure 1(a) that for both tanh and D data sets the Block-GP performs marginally better than GP and GP-V respectively. For the California data set there is a 48% improvement in the prediction result since

$$\frac{\text{NMSE}_{\text{Block-GP}}}{\text{NMSE}_{\text{GP-V}}} = \frac{0.0229}{0.0443} \approx 52\%$$

. The variance is also very low for the California data set indicating that the hyperparameter training is sufficient and the model is stable. The variance is comparatively higher for the autoregressive time series data sets. This can be attributed to the inclusion of boundary (switching) points in the test sets, which are difficult to model in the absence of prior time points in the training set due to the autoregressive nature of the time series.

Figure 1(b) demonstrates the dependence of the accuracy of Block-GP on the choice of the number of blocks in the data partitioning step, given a fixed number of modes of the data. We use the California data set for this purpose and report accuracy results for 10 trials of the experiment. Knowing that the data has 10 modes, we vary the number of modes in our experiment from 5 to 50 and study the variation in the NMSE. We can see in Figure 1(b), that the mean NMSE is lowest when we use $k = 10$ for our data partitioning step. This performance is in accordance with the domain expert's feedback of partitioning the California surface reflectance data into 10 groups. We also notice that although the NMSE does not change considerably as we experiment with other number of modes, the variance in the performance increases indicating instability of the model which might be attributed to lack of sufficient representation of each data partition for training the Block-GP model. In summary, we see that Block-GP has higher prediction accuracy than standard GP and GP-V for the different multimodal data sets and this performance is dependent on the choice of the covariance function and the correct knowledge of the number of modes in the data.

B. Scalability

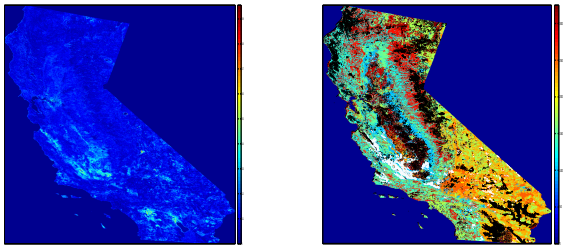
We test our algorithm for scalability on the California data set for 2001 which has 15 million training points with 3 input dimensions. Figure 2 shows the plot of the training time and test time separately for the Block-GP method in the distributed experimental setup described earlier. For demonstrating the dependence of Block-GP on the size of the training data, we vary the training set size from 15,000 observations to 5 million observations and keep the size of the test set fixed at 1500 test points. Figure 2(a) shows the plot of the time required for training the model and testing a fixed sized test set. As expected, the test time is almost constant for all the cases and we see a linear increase in the training time as we vary the size of the training set. This is because the California data set has 3 input dimensions which makes the $O(n_{max}D^2)$ Block-GP method an $O(n_{max})$ algorithm for this data set. Figure 2(b) shows the increase in the running time of Block-GP in the same distributed experimental setup as we increase the number of data partitions, keeping both the size of the training and the test set fixed. Comparing the y axis scales of Figures 2(a) and 2(b), we see that the increase in the running time for increasing number of partitions is negligible compared to the increase in the training set size. However, the running time in Figure 2(b) is not a constant, as is expected in a parallel setup, since the book-keeping tasks for creating distributed jobs and submitting them to the task schedulers take an amount of time that increases as the number of parallel blocks increase.

The running times reported in Figure 2 do not account for the time required for data partitioning since different implementations of spectral clustering will lead to different running times. Our experiments on the California data set that use 5% of the training data and the results of which are discussed in details in Sections IV-A and IV-C, use a Nyström approximation based spectral clustering implementation [9]. We do not compare the the running times of Block-GP with GP-V since ones executes in a centralized setup while the other in a distributed setup and has factors like network latency and disk i/o influencing the running time.

C. Analysis of California data set results

In this section we analyze the performance of Block-GP in comparison to GP-V on the California data set. For all results reported here we have used the 2001 data for training and 2002 for testing. A close study of the absolute error distribution of the two methods show that Block-GP has lower prediction error (color map shown in Figure 3(a)) for Band 6 than GP-V. GP-V has a maximum error value of

0.103 and a mean error of 0.06 while the same quantities for Block-GP are 0.09 and 0.05 respectively. A similar study of the predictive variances also indicate that GP-V is less confident about predicting the target than Block-GP. This is expected, since Block-GP has separate *experts* in charge of predicting different regions of California whereas GP-V has just one model for predicting the different variations and is unable to capture the specific characteristics of the different regions. The comparative performance of the two methods is captured in Figure 3(b) which shows the top 5% of cases where GP-V performs better than Block-GP as white dots and the top 5% of cases where Block-GP performs better than GP-V in black dots overlayed on the map of California color coded into ten different clusters identified by our spectral clustering algorithm. As is evident from the error plot, there is a small region in central California that Block-GP fails to predict as accurately as GP-V. This however, cannot be attributed to inaccurate modeling of that region since we know from the color map of clustering that other test points that belong to that cluster have been modeled more accurately by Block-GP than GP-V. One possible reason can be a noisy target which is modeled better by a not-so-accurate model (GP-V in this case). There is another region of white dots near the Mojave desert area of California where GP-V performs better than Block-GP. This might be indicative of the partitioning of the data into one too many clusters in that region leading to poor representation of data for the smaller cluster. It is possible that the two clusters in that region should be merged to a single cluster due to very high similarity in their characteristics. All black dots in Figure 3(b) represent points where Block-GP performs significantly better than GP-V. Most of these points lie in smaller clusters identified by our spectral clustering algorithm and can be explained by the lack of representation of training points from those regions while learning a single model for GP-V.



(a) Block-GP error map (b) Block-GP vs. GP-V

Figure 3. Performance of Block-GP on California data set

V. Conclusion

In this paper we have presented a new method for scaling up Gaussian Process regression for multimodal data. Compared to existing techniques, our proposed method shows comparable or improved prediction accuracy for different multimodal data sets. The confidence in prediction is also higher compared to a single model. The theoretical bounds on the scalability improve the existing state-of-the-art by a factor of $k + 1$, for a data set with k modes. Since the proposed solution is decomposable, the training of the individual Gaussian Process blocks is parallelizable, thereby speeding up the total training time of the method to match that of training a single Gaussian Process expert. This method offers interpretability of the sparse solution since the different blocks identified by the spectral clustering algorithm can be traced back to the different modes of the actual data set providing better insight into the model. Our experimental results validate the performance claims of our method on different synthetic and real-world multimodal data sets. Depending on the kernel function used, this method allows the model to capture the non-linear relationship in the data in a highly scalable and accurate fashion. In our future research we intend to explore how our method performs compared to existing regression techniques that exploit data sparsity for scalability.

Acknowledgment

This work was supported by the NASA Terrestrial Observation and Prediction System and the Integrated Vehicle Health Management projects. The authors would like to thank Rama Nemani and Petr Votava for the the MODIS dataset and Nikunj Oza and Santanu Das for valuable feedback on the paper.

References

[1] C. E. Rasmussen, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[2] A. J. Smola and P. Bartlett, "Sparse greedy gaussian process regression," in *Proc. of NIPS 13*, 2000, pp. 619–625.

[3] L. Foster, A. Waagen, N. Aijaz, M. Hurley, A. Luis, J. Rinsky, C. Satyavolu, M. Way, P. Gazis, and A. Srivastava, "Stable and Efficient Gaussian Process Calculations," *JMLR*, vol. 10, pp. 857–882, 2009.

[4] V. Tresp, "The generalized bayesian committee machine," in *Proc. of KDD*, 2000, pp. 130–139.

[5] ——, "Mixtures of gaussian processes," in *Proc. of NIPS 13*, 2000, pp. 654–660.

[6] C. E. Rasmussen and Z. Ghahramani, "Infinite mixtures of gaussian process experts," in *NIPS 14*, 2001, pp. 881–888.

[7] U. Luxburg, "A Tutorial on Spectral Clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[8] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral Grouping Using the Nyström Method," *IEEE PAMI*, vol. 26, no. 2, pp. 214–225, 2004.

[9] W. Chen, Y. Song, H. Bai, C. Lin, and E. Y. Chang, "Parallel spectral clustering in distributed systems," *IEEE PAMI*, 2010.