IIE Transactions
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/uiie20

Bayesian site selection for fast Gaussian process regression

Arash Pourhabib [a], Faming Liang [b] & Yu Ding [a]
[a] Department of Industrial and Systems Engineering, Texas A&M University, College Station, TX, 77843-3143, USA
[b] Department of Statistics, Texas A&M University, College Station, TX, 77843-3143, USA
Accepted author version posted online: 25 Oct 2013.Published online: 05 Feb 2014.

To cite this article: Arash Pourhabib, Faming Liang & Yu Ding (2014) Bayesian site selection for fast Gaussian process regression, IIE Transactions, 46:5, 543-555, DOI: 10.1080/0740817X.2013.849833

To link to this article: http://dx.doi.org/10.1080/0740817X.2013.849833

IIE Transactions (2014) **46**, 543–555

ISSN: 0740-817X print / 1545-8830 online
DOI: 10.1080/0740817X.2013.849833

Bayesian site selection for fast Gaussian process regression

ARASH POURHABIB[1], FAMING LIANG[2] and YU DING[1,*]

[1]*Department of Industrial and Systems Engineering and* [2]*Department of Statistics, Texas A&M University, College Station, TX 77843-3143, USA*
E-mail: yuding@iemail.tamu.edu

Received April 2012 and accepted June 2013

Gaussian Process (GP) regression is a popular method in the field of machine learning and computer experiment designs; however, its ability to handle large data sets is hindered by the computational difficulty in inverting a large covariance matrix. Likelihood approximation methods were developed as a fast GP approximation, thereby reducing the computation cost of GP regression by utilizing a much smaller set of unobserved latent variables called *pseudo points*. This article reports a further improvement to the likelihood approximation methods by simultaneously deciding both the number and locations of the pseudo points. The proposed approach is a Bayesian site selection method where both the number and locations of the pseudo inputs are parameters in the model, and the Bayesian model is solved using a reversible jump Markov chain Monte Carlo technique. Through a number of simulated and real data sets, it is demonstrated that with appropriate priors chosen, the Bayesian site selection method can produce a good balance between computation time and prediction accuracy: it is fast enough to handle large data sets that a full GP is unable to handle, and it improves, quite often remarkably, the prediction accuracy, compared with the existing likelihood approximations.

Keywords: Bayesian model averaging, Gaussian process computation, large data sets, reversible jump MCMC

1. Introduction

Since its introduction, Gaussian Process (GP) regression has gained popularity among experts ranging from computer scientists and statisticians to engineers. GP's flexibility, nonlinearity, and inherent nonparametric structure are the key features that have made it of use to a wide range of researchers (Rasmussen and Williams, 2006). GP regression has proliferated in recent years due to the widespread availability of data. On the other hand, however, the vast amount of data, while furnishing adequate information to train the model, could hamper computationally efficient implementation of GP regression. As the Gaussian distribution is central to the GP regression, in almost all methods of full GP regression one needs to invert matrices of size equal to the number of data points; this could be a burdensome task as its complexity is of order $\mathcal{O}(N^3)$; it should be noted that most methods require executing this matrix inversion many times to guarantee successful implementation of the algorithms.

Spurred on by GP's popularity, research has been conducted in recent years to address the computational issue of handling large data sets. There are two main schools of thought: sparse approximation and low-rank approximation. The sparse approximation methods employ a compactly supported covariance function in a way that results in a sparse covariance matrix, still of size N, but inverting this sparse matrix using sparse matrix algorithms (Gneiting, 2002; Furrer *et al.*, 2006) can lead to a substantial reduction in computational effort. Although the theoretical complexity of this method is difficult to determine, Furrer *et al.* (2006) observed, through a number of numerical case studies, that the training computation increases almost linearly in N. This class of algorithms, however, suffers from a high order of complexity during the test stage, which is also linear in N, whereas the low-rank approximation, as we will briefly review below, can do faster than $\mathcal{O}(N)$ during testing.

The second school of thought, the low-rank approximation, tries to reduce the computational complexity by inverting a matrix of reduced rank instead of the original covariance matrix. Utilizing different techniques to produce the reduced-rank matrix, the low-rank approximation can be categorized into three groups: matrix approximation, localized regression, and likelihood approximation. The algorithms based on matrix approximation seek substitutions for the original covariance matrix, which has truncated bases (e.g., Nyström method), resulting in a rank reduction, and can therefore be handled less expensively (Quiñonero-Candela and Rasmussen, 2005). Localized regression assumes that data points far from each other do

*Corresponding author
Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/uiie.

not share any measure of similarity and one can employ a GP on a local region by merely taking the data points in the region of interest into account (Park *et al.*, 2013). The likelihood approximation methods (Seeger *et al.*, 2003; Snelson and Ghahramani, 2007; Snelson, 2007) try to reduce the computational cost by making use of a set of unobserved latent variables called *pseudo points*. Assuming the conditional independence of the observed variables given the pseudo outputs, one needs to invert matrices of a size equal to that of the pseudo points M, and doing so can save significant time if one chooses $M \ll N$ (Seeger *et al.*, 2003; Snelson, 2007; Snelson and Ghahramani, 2007). Specifically, assuming that each training and test point in the data set is independent from others given the pseudo points, one can achieve a computational complexity of $\mathcal{O}(NM^2)$ for the training stage and $\mathcal{O}(M^2)$ for testing (Snelson and Ghahramani, 2006). This complexity expression is generally true for other methods in the school of low-rank approximation methods, although the meaning of M in a specific method differs.

Although the methods in the low-rank approximation class typically mitigate the computational burdens, they suffer from other problems. For instance, the matrix approximation algorithms may lead to poor estimation and lack of interpretability (Snelson and Ghahramani, 2006), and its prediction variance is not guaranteed to be positive (Park *et al.*, 2013). In localized regression algorithms, it is not a straightforward task to select independent subsets. Localized regression predictions, in general, lack continuity on boundaries, and the existing methods to address this problem cannot handle data sets other than those from one- or two-dimensional spaces (Park *et al.*, 2013). In likelihood approximation, despite the fact that the accuracy and computation of the algorithm rely heavily on the number of pseudo points selected, there are no strict guidelines regarding how to choose them and, currently, M is typically fixed *a priori*.

In this article, we choose to make a further improvement for the likelihood approximation methods because they are easy to use and do not have problems such as higher complexity in testing (associated with the spare approximation) or discontinuity in prediction (with the localized GP). However, as we previously mentioned, one major improvement needed for the likelihood approximation methods is a more flexible way of deciding the number of pseudo points. The current inflexibility often causes the likelihood approximation methods to have a higher prediction error (measured by the mean squared errors using a testing data set).

For the purpose of improving the likelihood approximation, we propose a Bayesian Site Selection (BSS) method that allows the data to decide simultaneously the number and locations of pseudo inputs. Specifically, BSS considers the pseudo inputs as a new set of parameters in the model and selects them from a subspace of the training data. Then, BSS tries to estimate the posterior predictive distribution via a Markov Chain Monte Carlo (MCMC) method. We call the new set of parameters in the BSS *sites*, which are the counterparts of the pseudo inputs in the likelihood approximation methods. We generate an artificial GP defined on the sites and condition our real response on those artificially generated outputs to reduce the order of computational complexity. Note that Park and Liang (2013) also used the same name for their method but the methods as well as the targeted applications are different.

Compared with the current likelihood approximation methods, BSS tries to systematically discover the number of sites used as the pseudo inputs. The efforts spent in finding the appropriate sites are valuable as those lead to a less subjective algorithm and produce more accurate results. In addition, BSS chooses the locations of sites based on an MCMC algorithm, and by applying MCMC, BSS employs more than one GP approximation and can thereby provide more accurate prediction results through the mechanism of Bayesian model averaging (Hoeting *et al.*, 1999).

Understandably, performing all of these tasks in BSS comes at a higher computational cost than the existing likelihood approximation methods. The theoretical computational complexities of the likelihood approximation methods and BSS, if using the same number of sites, are of the same order; for training, it is at $\mathcal{O}(LNM^2)$, where L is the number of iterations a method employs to fit the model—the model fitting process is also known as *hyperparameter learning* in GP research (Rasmussen and Williams, 2006). Because the current likelihood approximation methods use a deterministic gradient-based optimization method, its L in practice is smaller than that in BSS; L in BSS is the number of iterations of the MCMC. However, we would like to point out that BSS, with appropriately chosen priors, can produce a practically sensible balance between computation time and prediction accuracy; it is fast enough to handle large data sets that a full GP is unable to handle, while it improves, quite often remarkably, the prediction accuracy, compared with deterministic likelihood approximations.

The remainder of this article is organized as follows. Section 2 gives the GP regression formulation and uses the method proposed in Snelson and Ghahramani (2006) to explain the basic thoughts behind the likelihood approximation. Section 3 presents the details of the BSS approach, including discussions about the implementation of the method. In Section 4, we implement our method on several simulated and real data sets and show that the BSS method outperforms the existing methods for several test cases. Finally, we conclude the article in Section 5 with additional discussions and comments.

2. Likelihood approximation based on pseudo inputs

A Gaussian process is a continuous stochastic process in which any finite number of variables have a joint Gaussian distribution (Rasmussen and Williams, 2006). That is, if $\mathbf{f} = \{f_1, f_2, \ldots, f_N\}$ represents a finite collection of these variables, where each f_i is defined over its corresponding

index $\mathbf{x}_i$, then we have

$$\pi(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{K}), \tag{1}$$

where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, and $\mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$ denotes a multivariate Gaussian distribution over $\mathbf{x} \in \mathbb{R}^d$ with mean $\boldsymbol{\mu}$ and covariance matrix $\mathbf{K}$ whose entries are defined by the covariance function $K(.,.)$:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left[-\frac{1}{2}\sum_{\ell=1}^{d}\left(\frac{x_{i\ell} - x_{j\ell}}{\eta_\ell}\right)^2\right], \tag{2}$$

where $\boldsymbol{\eta}_\ell$ is the ℓth component of the vector of the length-scale parameter $\boldsymbol{\eta} = \{\eta_1, \eta_2, \ldots, \eta_d\}$, $x_{i\ell}$ and $x_{j\ell}$ are the ℓth components of $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively, and d is the dimension of the input space. As we can always subtract a constant from the response values before using the data, without loss of generality we can assume $\boldsymbol{\mu} = 0$.

Having defined a GP, we are interested in employing the GP for the regression task. We assume that we observe $\mathbf{y} = \{y_1, y_2, \ldots, y_N\}$ as a sum of the GP $\mathbf{f} = \{f_1, f_2, \ldots, f_N\}$ and some additive white Gaussian noise:

$$y_i = f_i + \epsilon_i \quad \text{for } i = 1, 2, \ldots, N, \tag{3}$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, and $\text{cov}(\epsilon_i, \epsilon_j) = 0$ for any $i \neq j$. To estimate the model parameters, we can optimize the marginal likelihood $\pi(\mathbf{y}|\mathbf{X}, \theta)$ with respect to θ where $\theta = \{\sigma, \sigma_f, \boldsymbol{\eta}\}$ (Rasmussen and Williams (2006), p. 112). If we want to make a prediction at some test input $\mathbf{x}_*$, the distribution of the corresponding response (i.e., y_*) will be Gaussian with mean μ_* and variance σ_*^2, where

$$\begin{aligned}\mu_* &= \mathbf{K}_{*N}(\mathbf{K}_N + \sigma^2\mathbf{I})^{-1}\mathbf{y},\\ \sigma_*^2 &= K_* - \mathbf{K}_{*N}(\mathbf{K}_N + \sigma^2\mathbf{I})^{-1}\mathbf{K}_{N*} + \sigma^2.\end{aligned} \tag{4}$$

In the above, the subscript of the covariance matrix implies the data points for which the covariance matrix is formed. For example, $\mathbf{K}_N$ is the $N \times N$ covariance matrix of all training data points, $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. For a covariance matrix between a single point and a set of points, such as the test point $\mathbf{x}_*$, we explicitly denote the point in the covariance notation. Therefore, $\mathbf{K}_{*N}$ denotes the $1 \times N$ covariance matrix between the test input $\mathbf{x}_*$ and the training points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. Finally, K_* is used to denote $K(\mathbf{x}_*, \mathbf{x}_*)$, the prior variance associated with the test site $\mathbf{x}_*$. The same symbolism is used throughout this article.

The computational issue mentioned earlier is related to the inversion of $(\mathbf{K}_N + \sigma^2\mathbf{I})$. This inversion happens during the learning stage of the parameters at the order of $\mathcal{O}(N^3)$. Note that once the parameter learning is done, the calculation of the mean and variance of a test point $\mathbf{x}_*$ costs $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$, respectively.

The likelihood approximation method we aim at improving in this article is the one using a Sparse Pseudo input Gaussian (SPGP) process (Snelson and Ghahramani, 2006). The SPGP method works as follows. Instead of using the N observations of y directly, which are too numerous and cause the computational problem, one can consider using a much smaller set of inputs of size $M \ll N$ to approximate the full GP model. If this smaller set of inputs is a subset of the actual N data points, that method is called the Subset of Data Approximation (SDA; Quiñonero-Candela and Rasmussen (2005)). However, researchers have realized that the simple SDA usually does poorly in approximating the full GP because the possible subsets are restricted to the locations where the data were observed. If that restriction is lifted, meaning that if the input locations can be strategically selected to be at places where there are not necessarily any observations, the resulting GP approximation can be considerably improved, and the unrestricted inputs are then called pseudo inputs.

Following the notations in Snelson (2007), let $\bar{\mathbf{X}} = (\bar{\mathbf{x}})_{m=1}^{M}$ denote the pseudo inputs and $\bar{\mathbf{f}} = (\bar{f})_{m=1}^{M}$ denote the pseudo outputs. The bar notation shows that the pseudo inputs (and outputs) reside in the same spaces as those of real data, but they cannot be observed. Another important point is that since pseudo outputs are not actually observed, it does not make sense to include observation noise (i.e., ϵ) in them, which is why $\bar{\mathbf{f}}$ is used instead of $\bar{\mathbf{y}}$. Based on the same reasoning presented for the selection of the GP prior in Equation (1), we can assume the following prior for pseudo outputs:

$$\pi(\bar{\mathbf{f}}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_M), \tag{5}$$

and if assuming the outputs are independently and identically distributed given the inputs, one can have:

$$\begin{aligned}\pi(\mathbf{y} \mid \bar{\mathbf{f}}, \mathbf{X}, \bar{\mathbf{X}}, \theta) &= \prod_{n=1}^{N}\pi(y_n \mid \bar{\mathbf{f}}, \mathbf{X}, \bar{\mathbf{X}}, \theta)\\ &= \mathcal{N}\big(\mathbf{K}_{NM}\mathbf{K}_M^{-1}\bar{\mathbf{f}}, \text{diag}(\mathbf{K}_N - \mathbf{Q}_N) + \sigma^2\mathbf{I}\big),\end{aligned} \tag{6}$$

where $\mathbf{K}_{NM}$ is the $N \times M$ covariance matrix between the N training points and the M pseudo inputs (Snelson (2007), p. 38). The matrix $\mathbf{Q}_N$ is the low-rank covariance matrix whose entries are defined by the low-rank covariance function $Q(.,.)$,

$$Q(\mathbf{x}, \mathbf{x}') = \mathbf{K}_{\mathbf{x}M}\mathbf{K}_M^{-1}\mathbf{K}_{M\mathbf{x}'}. \tag{7}$$

Next, integrating out the pseudo outputs produces the marginal likelihood

$$\begin{aligned}\pi(\mathbf{y} \mid \mathbf{X}, \bar{\mathbf{X}}, \theta) &= \int \pi(\mathbf{y} \mid \bar{\mathbf{f}}, \mathbf{X}, \bar{\mathbf{X}}, \theta)\,\pi(\bar{\mathbf{f}} \mid \bar{\mathbf{X}}, \theta)\text{d}\bar{\mathbf{f}}\\ &= \mathcal{N}(\mathbf{0}, \mathbf{Q}_N + \text{diag}(\mathbf{K}_N - \mathbf{Q}_N) + \sigma^2\mathbf{I}).\end{aligned} \tag{8}$$

The predictive distribution can be obtained by first writing the joint distribution of $\pi(y_*, \mathbf{y})$, which is multivariate normal and takes the same form as in Equation (8). From the joint distribution of $\pi(y_*, \mathbf{y})$, the prediction distribution $\pi(y_*|\mathbf{y})$ can be obtained using the conditional normal distribution formula (Rasmussen and Williams (2006), p. 200).

The resulting $\pi(y_*|\mathbf{y})$ is as follows:

$$\begin{aligned}
&\pi(y_* \mid \mathbf{y}, \mathbf{X}, \bar{\mathbf{X}}, \theta) = \mathcal{N}(\mu_*, \sigma_*^2),\\
&\mu_* = \mathbf{Q}_{*N}(\mathbf{Q}_N + \text{diag}(\mathbf{K}_N - \mathbf{Q}_N) + \sigma^2\mathbf{I})^{-1}\mathbf{y},\\
&\sigma_*^2 = K_* - \mathbf{Q}_{*N}(\mathbf{Q}_N + \text{diag}(\mathbf{K}_N - \mathbf{Q}_N) + \sigma^2\mathbf{I})^{-1}\mathbf{Q}_{N*} + \sigma^2.
\end{aligned} \tag{9}$$

Despite entailing $N \times N$ matrix inversions in Equation (9), using the matrix inversion lemma one can show that the computational complexity is actually $\mathcal{O}(NM^2)$ (Snelson (2007), p. 40). The reason is simply that after using the matrix inversion lemma, the $N \times N$ matrices will become diagonal whose inversion is $\mathcal{O}(N)$ and, consequently, the computation is no longer dominated by inverting those matrices.

The parameters in the above model can be categorized into two groups:

1. The so-called hyperparameters that are also used by other GP models, usually denoted by θ. Here $\theta = \{\sigma, \sigma_f, \eta\}$.
2. The locations of pseudo inputs $\bar{\mathbf{X}}$.

To estimate all of the model parameters $(\theta, \bar{\mathbf{X}})$ together, one can use gradient ascent methods to optimize the marginal likelihood in Equation (8). The details of the optimization procedure as well as how to take the gradients can be found in Snelson ((2007), pp. 126–129). For the SPGP method, the cost for computing the marginal likelihood in Equation (8) once is $\mathcal{O}(NM^2)$, due to the fact that $\mathbf{Q}_N$ is of rank M (lower than N). If the optimization method takes l steps to converge, then the training cost—i.e., that for hyperparameter learning—is $\mathcal{O}(lNM^2)$. We will refer to the SPGP method as Deterministic Site Selection (DSS) hereafter. This name is chosen because it helps highlight the difference between the existing likelihood approximation and our proposed Bayesian method.

3. BSS

One drawback of the DSS mechanism in SPGP is that the number of pseudo inputs—i.e., the cardinality of $\bar{\mathbf{X}}$—is fixed at M. Given the important role that the number of pseudo inputs plays in both computation and prediction accuracy, it would be desirable that the number of pseudo inputs can change in the algorithm and be decided by the data. That is indeed the objective of the research presented in this article, through a method we label as BSS.

To establish a Bayesian framework for this problem, we begin by emphasizing that the goal is to make an inference about the posterior predictive distribution $\pi(y_*|\mathbf{y}, \mathbf{X}, \theta)$. As we are interested in utilizing the information inherent in the pseudo inputs, we consider the pseudo inputs as a set of new parameters through which we can represent the predictive distribution, namely:

$$\pi(y_*|\mathbf{y}, \mathbf{X}, \theta) = \int \pi(y_*|\mathbf{y}, \mathbf{X}, \bar{\mathbf{X}}, \theta)\, \pi(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \theta) \mathrm{d}\bar{\mathbf{X}}. \tag{10}$$

Equation (10) requires specifying the posterior distribution $\pi(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \theta)$, which can be expressed as

$$\pi(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \theta) \propto \pi(\bar{\mathbf{X}})\, \pi(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}}, \theta). \tag{11}$$

Note that $\pi(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}}, \theta)$ follows a normal distribution according to Equation (8). Moreover, as the new observations y_* and $\mathbf{y}$ are jointly normally distributed, $\pi(y_*|\mathbf{y}, \mathbf{X}, \bar{\mathbf{X}}, \theta)$ is also normally distributed with parameters shown in Equation (2). Therefore, the only term to be determined in order to fully specify the model is the prior distribution of the pseudo inputs. A suitable prior distribution should take into account the number of pseudo inputs so that we can update our belief about their number in light of the observed data. Considering $\bar{\mathbf{X}}$ as an $M \times d$ matrix, one reasonable choice for the prior could be

$$\pi(\bar{\mathbf{X}}) \propto \frac{\lambda^{|\bar{\mathbf{X}}|}}{|\bar{\mathbf{X}}|} \times I(k_\text{l} \le |\bar{\mathbf{X}}| \le k_\text{u}), \tag{12}$$

where $|\bar{\mathbf{X}}|$ denotes the number of rows in $\bar{\mathbf{X}}$ (i.e., the number of pseudo inputs), and $I(.)$ is the indicator function. The prior considers a range for the number of locations $(k_\text{l} \le |\bar{\mathbf{X}}| \le k_\text{u})$, which describes the smallest and the largest number of pseudo inputs we would like to consider in our model. The new hyperparameter λ reflects our belief regarding the average number of pseudo inputs in the prior.

Having specified the terms constituting the integrand in Equation (10), we want to evaluate the value of the integral. Unfortunately, the integral in Equation (10) cannot be solved analytically and, consequently, we need to resort to numerical methods to approximate the integral. In fact, if we can generate Markov samples $\{\bar{\mathbf{X}}_1, \bar{\mathbf{X}}_2, \ldots, \bar{\mathbf{X}}_T\}$ from $\pi(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \theta)$, then we can approximate the integral in Equation (10) by

$$\hat{\pi}(y_*|\mathbf{y}, \mathbf{X}, \theta) = \sum_{t=1}^{T} \pi(y_*|\mathbf{y}, \mathbf{X}, \bar{\mathbf{X}}_t, \theta). \tag{13}$$

Then, the problem is reduced to how to draw samples from $\pi(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \theta)$.

The difficulty associated with drawing samples from $\pi(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \theta)$ is that a direct application of MCMC fails as it requires the state space of the Markov chain to be of a fixed dimension, but the dimension of $\bar{\mathbf{X}}$ may actually vary. To overcome this issue, we can use the Reversible Jump Markov Chain Monte Carlo (RJMCMC) algorithm (Green, 1995), which allows the dimension of the state space of the Markov chain to vary. The underlying idea is that RJMCMC introduces three types of moves: exchange, birth, and death. Exchange means that the chain remains in the space with the same dimension but moves into a new

state. Birth and death are the moves that change the dimension of the state space. Intuitively, a birth step augments the state space by adding new states, whereas a death step reduces the dimension of the state space. At each iteration, the type of move, whether an exchange, birth, or death, is randomly chosen, and one accepts the new state using a Metropolis–Hastings rule. To see full details and examples, refer to Green (1995).

3.1. *Algorithm*

Specifically, to generate the sample $\{\bar{\mathbf{X}}_1, \bar{\mathbf{X}}_2, \ldots, \bar{\mathbf{X}}_T\}$, we build a Markov chain whose space is a subset of $\mathbb{R}^{d\times s}$ for varying values of s, so that at stage t the corresponding set of pseudo inputs $\bar{\mathbf{X}}_t$ may have a number of elements different from previous stages. This allows the number of pseudo inputs to change so that we can seek simultaneously the number and location of the pseudo inputs.

To employ the RJMCMC, we need to introduce some new notations and make some extra assumptions. We restrict the space from which the pseudo inputs can be chosen by imposing it to be finite. Specifically, let $\mathbf{S}$ denote the whole space of explanatory variables, so if $\mathbf{x}$ is an element in $\mathbf{X}$ or $\bar{\mathbf{X}}$, then $\mathbf{x} \in \mathbf{S}$. Let $\tilde{\mathbf{S}} \subset \mathbf{S}$ denote a finite discretized subspace in the sense that $|\tilde{\mathbf{S}}| < \infty$. For an $\mathbf{x} := \{x_1, x_2, \ldots, x_d\} \in \tilde{\mathbf{S}}$, $x_i \in \{\tilde{x}^i_{\min}, \tilde{x}^i_{\min} + \xi, \tilde{x}^i_{\min} + 2\xi, \ldots, \tilde{x}^i_{\max}\}, \forall i \in \{1, 2, \ldots, d\}$, where $\tilde{x}^i_{\min}$ and $\tilde{x}^i_{\max}$ are the minimum and maximum values to consider in the ith dimension of $\mathbf{S}$, respectively, and ξ is the discretization step. We choose the locations of the pseudo inputs from $\tilde{\mathbf{S}}$ instead of from the original $\mathbf{S}$ for the sake of ease of computation. As such, $\bar{\mathbf{X}} \subset \tilde{\mathbf{S}}$ and we refer to the $\bar{\mathbf{X}}$ as "sites" in our approach. Note that if ξ is taken to be a small number and $|\tilde{x}^i_{\min}|$ and $|\tilde{x}^i_{\max}|$ are large enough, then $\tilde{\mathbf{S}}$ can approximate $\mathbf{S}$ reasonably well.

As in other RJMCMC-based algorithms, we also use three types of moves: birth, death, and exchange. Birth and death are used to add or remove points to the current $\bar{\mathbf{X}}$, and exchange is used to update the locations of sites with their number being kept unchanged. Recall that the set of parameters for this model is $(\theta, \bar{\mathbf{X}})$, where $\theta = \{\sigma, \sigma_f, \boldsymbol{\eta}\}$. In order to optimize θ, a full Bayesian approach is to incorporate both θ and $\bar{\mathbf{X}}$ in the RJMCMC algorithm and update them both as the chain evolves. Based on our numerical analysis, this approach, although theoretically appealing, does not provide stable results numerically; at least we have not found a robust enough numerical procedure that leads to that outcome. Therefore, we choose to employ the gradient ascent method to find a θ that maximizes the marginal likelihood (8) for a fixed set of $\bar{\mathbf{X}}$. That is to say, after every few iterations we seek for the optimal value of θ conditioned on the current value of $\bar{\mathbf{X}}$, and then we update $\bar{\mathbf{X}}$ using RJMCMC moves, conditioned on the newly found value of θ. We denote the number of RJMCMC iterations between two consecutive optimizations of θ by κ. We also want to note that the algorithm used here to maximize the marginal likelihood (8), for a given $\bar{\mathbf{X}}$, is the same as that in the full GP.

Next, we present the specific formulations of the RJMCMC moves. In the following, $q(A \to B)$ is the proposal distribution denoting the probability of going from a set $A \subset \tilde{\mathbf{S}}$ to $B \subset \tilde{\mathbf{S}}$, and a and b are the probabilities of performing birth and death operations, respectively; both will be explained after the formulations.

Birth: Choose a point from $\tilde{\mathbf{S}}\backslash\bar{\mathbf{X}}$, say $\mathbf{x}^*$, and add it to the current site set $\bar{\mathbf{X}}$ with probability p so that the new site set is $\bar{\mathbf{X}} \cup \{\mathbf{x}^*\}$, where

$$p = \min\left(1, \frac{\pi(\bar{\mathbf{X}} \cup \{\mathbf{x}^*\}|\mathbf{y}, \mathbf{X}, \theta)\, q(\bar{\mathbf{X}} \cup \{\mathbf{x}^*\} \to \bar{\mathbf{X}})}{\pi(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \theta)\, q(\bar{\mathbf{X}} \to \bar{\mathbf{X}} \cup \{\mathbf{x}^*\})} \times \frac{b}{a}\right).$$

Death: Choose a point from $\bar{\mathbf{X}}$, say $\mathbf{x}^*$, and remove it from the current site set $\bar{\mathbf{X}}$ with probability p so that the new site set is $\bar{\mathbf{X}}\backslash\{\mathbf{x}^*\}$, where

$$p = \min\left(1, \frac{\pi(\bar{\mathbf{X}}\backslash\{\mathbf{x}^*\}|\mathbf{y}, \mathbf{X}, \theta)\, q(\bar{\mathbf{X}}\backslash\{\mathbf{x}^*\} \to \bar{\mathbf{X}})}{\pi(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \theta)\, q(\bar{\mathbf{X}} \to \bar{\mathbf{X}}\backslash\{\mathbf{x}^*\})} \times \frac{a}{b}\right).$$

Exchange: Choose a point from $\tilde{\mathbf{S}}\backslash\bar{\mathbf{X}}$, say $\mathbf{x}^*$, and a point from $\bar{\mathbf{X}}$, say $\mathbf{x}^{**}$, and exchange the two points with probability p where

$$p = \min\left(1, \frac{\pi(\bar{\mathbf{X}} \cup \{\mathbf{x}^*\}\backslash\{\mathbf{x}^{**}\}|\mathbf{y}, \mathbf{X}, \theta)\, q(\bar{\mathbf{X}} \cup \{\mathbf{x}^*\}\backslash\{\mathbf{x}^{**}\} \to \bar{\mathbf{X}})}{\pi(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \theta)\, q(\bar{\mathbf{X}} \to \bar{\mathbf{X}} \cup \{\mathbf{x}^*\}\backslash\{\mathbf{x}^{**}\})}\right).$$

Note it is straightforward to show that the Jacobian of transformation for the Birth and Death moves collapses to 1. Regarding the choice of the proposal distribution $q(A \to B)$ used in the moves, we choose to use a uniform function that assigns equal weights to all of the points in the sets $\tilde{\mathbf{S}}\backslash\bar{\mathbf{X}}$ and $\bar{\mathbf{X}}$. For example, $q(\bar{\mathbf{X}} \cup \{\mathbf{x}^*\} \to \bar{\mathbf{X}})$ can be expressed as $1/(|\bar{\mathbf{X}}| + 1)$ and $q(\bar{\mathbf{X}} \to \bar{\mathbf{X}} \cup \{\mathbf{x}^*\})$ is equal to $1/(|\tilde{\mathbf{S}}| - |\bar{\mathbf{X}}|)$. Choosing the uniform proposal and expressing the posteriors as the product of the priors and the likelihoods, we will get the following forms for the acceptance probabilities of birth, death, and exchange processes, respectively:

$$p_{\mathrm{B}} = \min\left(1, \frac{\lambda|\bar{\mathbf{X}}|(|\tilde{\mathbf{S}}| - |\bar{\mathbf{X}}|)\, f(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}} \cup \{\mathbf{x}^*\}, \theta)}{(|\bar{\mathbf{X}}| + 1)^2 f(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}}, \theta)} \times \frac{b}{a}\right),$$

$$p_{\mathrm{D}} = \min\left(1, \frac{|\bar{\mathbf{X}}|^2 f(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}}\backslash\{\mathbf{x}^*\}, \theta)}{\lambda(|\tilde{\mathbf{S}}| - |\bar{\mathbf{X}}| - 1)(|\bar{\mathbf{X}}| - 1) f(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}}, \theta)} \times \frac{a}{b}\right),$$

$$p_{\mathrm{E}} = \min\left(1, \frac{f(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}} \cup \{\mathbf{x}^*\}\backslash\{\mathbf{x}^{**}\}, \theta)}{f(\mathbf{y}|\mathbf{X}, \bar{\mathbf{X}}, \theta)}\right).$$

Other valid proposal distributions could be used as well; for a few other proposal distributions, see Liang *et al.* (2010).

Algorithm 1 presents the procedure of this RJMCMC, which generates a Markov chain of sites whose stationary distribution is $\pi(\bar{\mathbf{X}}|\mathbf{y}, \mathbf{X}, \theta)$. Once we have $\{\bar{\mathbf{X}}_t\}_{t=1}^T$ from the

posterior distribution of $\bar{\mathbf{X}}_t$, we can use Equation (13) to make a prediction at any untried points. In Algorithm 1, the values a and b can be selected as $a = b = 1/3$, which means that the probability of performing each of the operations birth, death, and exchange is equal.

Algorithm 1 BSS for SPGP

Choose $\bar{\mathbf{X}}_0$ as a uniformly random draw from $\tilde{\mathbf{S}}$ where $|\bar{\mathbf{X}}_0| = M$
Optimize the value of θ for a fixed $\bar{\mathbf{X}}_0$
repeat
  Draw u uniformly from [0, 1]
  if $u \leq a$ **then**
    Perform *Birth*
  else
    if $a < u \leq a + b$ **then**
      Perform *Death*
    else
      Perform *Exchange*
    end if
  end if
  After each κ steps optimize the value of θ for a fixed $\bar{\mathbf{X}}_t$
  After burn-in steps store the values of $\bar{\mathbf{X}}_t$
until The number of RJMCMC iterations reaches the pre-specified value of *MCMC Length*

To choose the optimal number and locations of the pseudo inputs, BSS's order of complexity for the training stage is $\mathcal{O}(LMN^2)$, and for the test stage it is $\mathcal{O}(LM^2)$, where L is the length of the MCMC chain. Comparing the BSS's computational complexity with that of the full GP, which is $\mathcal{O}(N^3)$ for the training stage and $\mathcal{O}(N^2)$ for the test stage, one can see a considerable cost reduction as long as M is chosen such that $M \ll N$. Comparing the BSS with the DSS, BSS costs more because L in the MCMC is longer than its counterpart in a gradient-based optimization, namely, the number of optimization iterations l. However, BSS generally produces results with better accuracy than those of DSS, as will be demonstrated in Section 4.

Finally, it should be noted that since the BSS method is a Bayesian approach, it naturally provides the posterior distribution that inherently contains information about the uncertainty associated with the method. Once the method is established, one only needs to sample from its posterior distribution enough times to get the mean prediction and the confidence intervals.

3.2. *Computational details*

As the new algorithm employs GP and MCMC methods, the reader can consult Rasmussen and Williams (2006) for general advice related to GP implementation and Liang *et al.* (2010) concerning MCMC.

The bottleneck for the computation comes from inverting the matrix of size N: (i) when we evaluate the likelihood in the birth, death, and exchange steps in RJMCMC and (ii) when we evaluate the posterior predictive distribution after burn-in steps. The computation of evaluating the likelihood in RJMCMC is on the order of $\mathcal{O}(NM^2)$ (Snelson (2007), p. 40), but the computation can be further reduced. Since the low-rank covariance in Equation (7) after each move changes moderately, we can exploit the similar structures of last obtained matrix to calculate the new one. To illustrate, assume that we want to perform the birth step. Let $\mathbf{x}_\mathrm{b}$ be the newly added point to the site set $\bar{\mathbf{X}}$. As such, the new low-rank covariance is

$$\mathbf{Q}^{\mathrm{new}} = \mathbf{K}_{N(M+1)}\mathbf{K}_{M+1}^{-1}\mathbf{K}_{(M+1)N}, \tag{14}$$

which can be written as

$$\mathbf{Q}^{\mathrm{new}} = \begin{bmatrix}\mathbf{K}_{NM} & \mathbf{K}_{N\mathbf{x}_\mathrm{b}}\end{bmatrix}\begin{bmatrix}\mathbf{K}_M & \mathbf{K}_{M\mathbf{x}_\mathrm{b}} \\ \mathbf{K}_{\mathbf{x}_\mathrm{b}M} & K_{\mathbf{x}_\mathrm{b}\mathbf{x}_\mathrm{b}}\end{bmatrix}^{-1}\begin{bmatrix}\mathbf{K}_{MN} \\ \mathbf{K}_{\mathbf{x}_\mathrm{b}N}\end{bmatrix}. \tag{15}$$

Let $k = K_{\mathbf{x}_\mathrm{b}\mathbf{x}_\mathrm{b}}$ and use the matrix inversion formula for a partitioned matrix (Rasmussen and Williams (2006), p. 201), we get

$$\mathbf{Q}^{\mathrm{new}} = \begin{bmatrix}\mathbf{K}_{NM} & \mathbf{K}_{N\mathbf{x}_\mathrm{b}}\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma}_M & -\frac{1}{c}\mathbf{K}_M^{-1}\mathbf{K}_{M\mathbf{x}_\mathrm{b}} \\ -\frac{1}{c}\mathbf{K}_{\mathbf{x}_\mathrm{b}M}\mathbf{K}_M^{-1} & \frac{1}{c}\end{bmatrix} \times \begin{bmatrix}\mathbf{K}_{MN} \\ \mathbf{K}_{\mathbf{x}_\mathrm{b}N}\end{bmatrix}, \tag{16}$$

where $c = k - \mathbf{K}_{\mathbf{x}_\mathrm{b}M}\mathbf{K}_M^{-1}\mathbf{K}_{M\mathbf{x}_\mathrm{b}}$ and $\boldsymbol{\Sigma}_M = \mathbf{K}_M^{-1} + (1/c)\mathbf{K}_M^{-1}\mathbf{K}_{M\mathbf{x}_\mathrm{b}}\mathbf{K}_{\mathbf{x}_\mathrm{b}M}\mathbf{K}_M^{-1}$. Therefore, to evaluate $\mathbf{Q}^{\mathrm{new}}$ instead of inverting $\mathbf{K}_{M+1}$ as in Equation (14), Equation (16) allows us to use the inverse of $\mathbf{K}_M$ from the previous step. Similar actions can be taken to facilitate the computation in the death and exchange steps.

3.3. *Choices of other parameters*

We share our thoughts on choosing the parameters when initializing the BSS algorithm. The first is what to choose for λ in the site prior (12). The effect of λ can be understood as follows. A small value of λ forces the algorithm to choose a smaller number of sites, whereas a large value of λ has the reverse effect. The latter typically results in a more accurate prediction but at the expense of a longer computation time. Taking this trade-off into consideration, one can decide the value of λ by selecting a subset of data to train the model for different values of λ. Then the trained model can be used to predict the responses of another unused subset of data to observe the prediction accuracy. As the computation time can approximately be extrapolated over the whole data set, this method can provide us with the information about the accuracy and computation time trade-off as a result of choosing different values for λ. In our implementation, the value of λ is selected to be $1.5(M/10^t d)$, where M is the initial number of sites, d is the dimension of the input space, and t is selected from the interval [1, 4].

The second parameter is the size of the discretized subspace, $\tilde{\mathbf{S}}\backslash\bar{\mathbf{X}}$, from which we choose the sites. The discretized subspace should be constructed based on the trade-off between computation time and accuracy. A larger $|\tilde{\mathbf{S}}\backslash\bar{\mathbf{X}}|$ yields more accurate results but could slow the computation. As a rule of thumb, one can choose $|\tilde{\mathbf{S}}\backslash\bar{\mathbf{X}}| = 10N$.

The next set of parameters is the initial locations of sites, decided in three steps: first, we randomly choose a subset of the training points; second, we find the new locations (with the number of pseudo inputs being kept fixed) by maximizing the marginal likelihood; and, finally, we find the closest points in the discretized steps to those locations.

The other set of parameters is the range of the pseudo inputs, namely, the lower and upper bounds k_{l} and k_{u} used in the site prior of Equation (12). Our analysis shows that the values of k_{l} and k_{u} do not affect the method as long as the range is selected wide enough. However, if a user wants to prevent the algorithm from choosing a large number of locations (for instance, due to the time constraint), he or she can choose a relatively small number for the upper bound k_{u}. On the other hand, if the user wants to make sure that the accuracy of the method is over some threshold, he or she can choose a relatively large number for the lower bound k_{l}, so that the algorithm will not choose a number of locations less than that limit. In the current implementation, the range is decided based on the initial number of pseudo inputs M and the number of data points in the training set N. Specifically, $k_{\mathrm{l}} = M/t_{\mathrm{l}}$, and $k_{\mathrm{u}} = t_{u_1} M + (N/t_{u_2})$ for some constants t_{l}, t_{u_1}, and t_{u_2}.

Finally, we want to note that the stopping criterion for this algorithm is decided based on the trade-off between computation time and accuracy, not necessarily based on the probabilistic convergence of a Markov chain. Doing so can be justified by noting two facts: first, the BSS is a Bayesian model averaging approach and, theoretically speaking, as the chain evolves, the results get more accurate. Second, letting the chain run for a very long time is counterproductive to the original purpose of the algorithm, which is to approximate the GP regression and provide a reasonable result in a relatively short period of time. As such, we would recommend that the stopping criterion be decided based on the presumed reduction in the mean-squared error in a specified period of time, which can be evaluated through cross-validation.

4. Experimental results

In this section we present the results obtained by applying the proposed method to some real and simulated data sets. First, we compare BSS with the Full GP (FGP) and Treed Gaussian Processes (TGP; Gramacy and Lee (2008)) for some small- to moderate-sized data sets. Then, we compare BSS with the DSS (Snelson, 2007) for some large data sets. Through both comparisons, we reinforce our claim that BSS provides a good trade-off: on the one hand, it can handle the large data sets that FGP and TGP usually can-not and, on the other hand, it is more accurate than DSS in terms of mean squared prediction errors. All of the numerical studies were performed on a computer with two 3.16 GHz quadcore CPUs.

4.1. *Data sets and performance criterion*

We used four real data sets: the first two data sets are available at the UCI Machine Learning Repository (UCI, 2010), and the other two are satellite data from NASA. The first set is the Abalone data set, which consists of 4177 points and each data record has an input vector $\mathbf{x}$ of dimension $d = 7$. The response in the Abalone data set is the abalone age, and its inputs in $\mathbf{x}$ are related to different properties in an abalone's body. The second set is the Sarcos data set, which consists of 48 933 data points and $d = 27$. The data are related to the dynamics of a robot. The two NASA data sets, the third and fourth real data sets, are spatial data; both have $d = 2$. The third data set is TCO, which consists of 48 331 measurements of the total column of ozone around the globe, collected by the NIMBUS-7/TOMS satellite on October 1, 1988. The fourth data set is MOD08-CL, which is the data collected by the Moderate Resolution Imaging Spectroradiometer on NASA's Terra satellite. The data points, 64 800 in total, are the measurements of the average of cloud fractions around the globe from January to September in 2009.

We also tested the proposed method on a set of simulated data sets generated using the revised Ackley's path function (Joseph and Kang, 2011), which is defined as

$$f(\mathbf{x}) = -\alpha \exp\left(-\beta \sqrt{\sum_{\ell=1}^{d} \frac{x_\ell^2}{d}}\right) - \exp\left(\sum_{\ell=1}^{d} \frac{\cos(\gamma x_\ell)}{d}\right) + \alpha + \exp(1), \quad \mathbf{x} \in [-2, 2]^d, \tag{17}$$

where $\alpha = 2d$, $\beta = 0.2$, and $\gamma = 2\pi$. To generate a set of data, d and N need to be specified. Additionally, the locations of the data points need to be selected, and following Joseph and Kang (2011), we used the Latin hypercube designs (Wu and Hamada, 2009) for this purpose.

To evaluate the performance of a method, we partitioned each simulated data set so that 80% was for training and the remaining 20% was for testing, and for the real data sets we used a five-fold cross-validation.

The primary evaluation criterion was the Mean Squared Error (MSE), defined as

$$MSE = \sum_{i=1}^{N_{\mathrm{t}}} \frac{(y_i - \hat{y}_i)^2}{N_{\mathrm{t}}},$$

where y_i is the observed value, $\hat{y}_i$ is the predicted value, and N_{t} is the number of test cases. When comparing BSS and DSS on the real data sets, in addition to MSE, which measures the accuracy of the mean prediction, we employed

a Predictive Log Score Measure (PLSM; Hoeting *et al.* (1999)), which takes into account the predictive uncertainty of a method. PLSM is formally defined as

$$PLSM = -\sum_{\delta \in D^{\mathrm{T}}} \log \left\{ \sum_{M \in \mathcal{A}} \Pr(\delta | M, D^{\mathrm{B}}) \Pr(M | D^{\mathrm{B}}) \right\}, \quad (18)$$

where D^{B} and D^{T} are the build data (i.e., training data) and test data, respectively. Specifically, $D^{\mathrm{B}} = \{(\mathbf{x}_i, y_i); i = 1, 2, \ldots, N\}$ and $D^{\mathrm{T}} = \{(\hat{\mathbf{x}}_i, \hat{y}_i); i = 1, 2, \ldots, N_{\mathrm{t}}\}$, where $(\mathbf{x}_i, y_i)$ is the ith input–output pair for training and $(\hat{\mathbf{x}}_i, \hat{y}_i)$ is the ith input–output prediction pair in which $\hat{\mathbf{x}}_i$ is the test input and $\hat{y}_i$ is the corresponding prediction. The set $\mathcal{A}$ contains all of the models used in a prediction; for the BSS, $\mathcal{A}$ is the set of all pseudo input-based approximations based on the set of pseudo inputs $\bar{\mathbf{X}}_t$ for $t >$ burn-in, and for the DSS $\mathcal{A}$ is simply one approximation based on a fixed number of pseudo inputs. A smaller PLSM indicates an overall better predictive performance of a method, considering all uncertainties involved.

4.2. *A one-dimensional example*

To illustrate how the algorithm works, we first generated a simulated data set from Equation (17) for $d = 1$ and $N = 5000$ and applied the BSS algorithm with $\lambda = 0.1$. Figure 1 presents four plots showing the results. Figure 1(a), the top left plot, displays half of the 4000 points selected as the training data set. Here we plot only the positive half as the revised Ackley's path is symmetric around the y-axis. Figure 1(b), the bottom left plot, illustrates how the sites (locations of the pseudo inputs) change as the MCMC chain evolves. Figure 1(c), the top right plot, displays the number of sites versus the number of MCMC iterations.

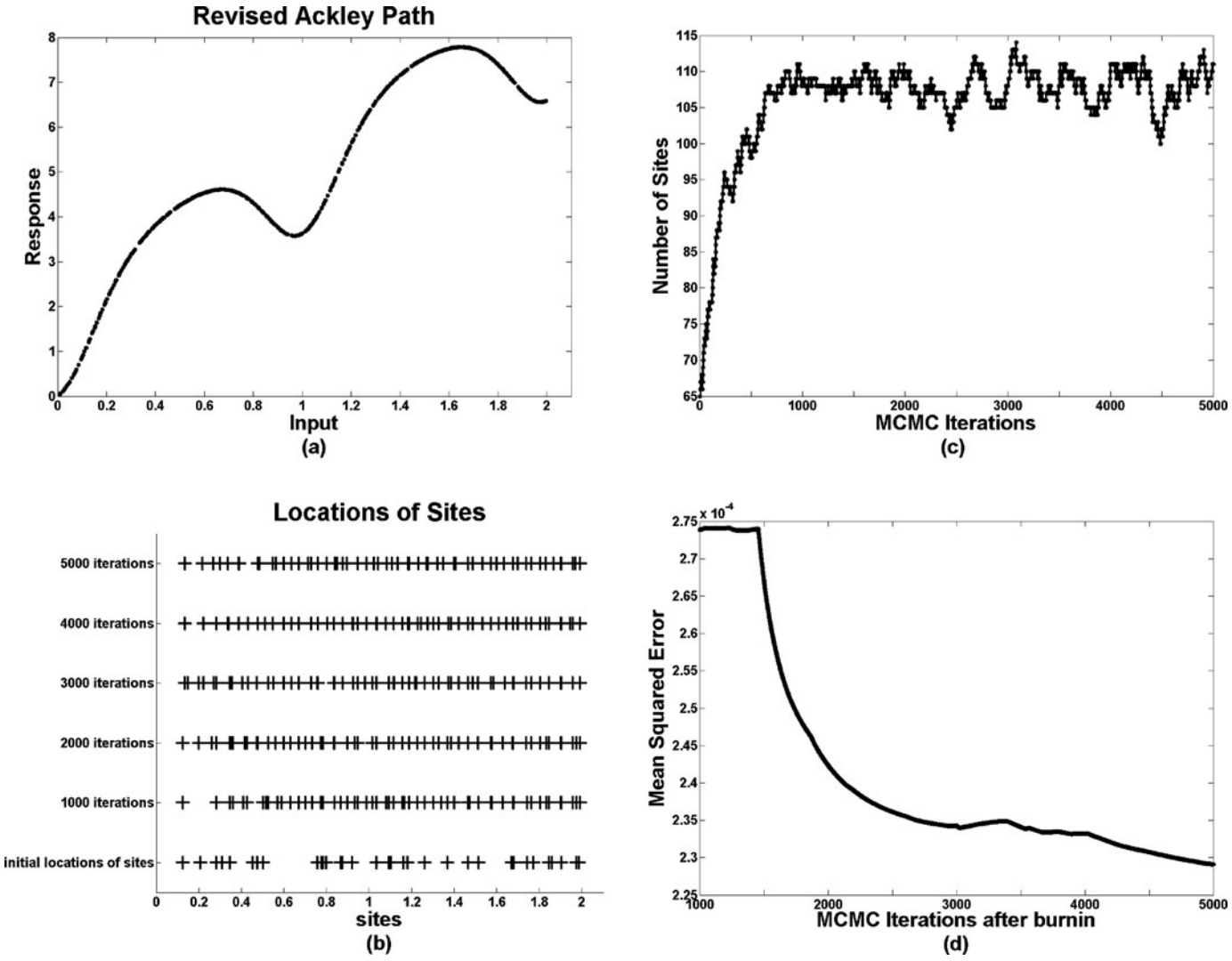

Fig. 1. (a) The revised Ackley's path when $d = 1$ and $N = 5000$ (the plot only shows 2000 training points whose inputs are positive), (b) the initial locations of sites and their new locations after every 1000 MCMC iterations (on the positive side of the axis), (c) number of sites verus MCMC iterations, and (d) MSE versus MCMC iterations after the burn-in period.

Considering the plots in Figs. 1(b) and 1(c), one can observe how the number and locations of sites change; during the first 1000 iterations, the number of sites increases to around 110. After that, the number of sites oscillates around 110, and the locations change as a combined result of the birth, death, and exchange operations. Although the plot indicates that the number of sites at a given iteration might be the same as that of a few iterations before, the locations of sites are not necessarily the same. The last plot, Fig. 1(d), displays how the MSE changes after the 1000 burn-in iterations. Initially, we observe a significant reduction in MSE and, then the decrease levels off. This example provides insights regarding how the BSS works: the method is initialized by selecting a number of sites, and through RJMCMC, BSS chooses different sites in terms of both location and quantity. In general, the predictive MSE has a decreasing trend and the number of sites converges toward a specific range, depending on the value of λ used.

4.3. *Performance comparison*

We first used the simulated data set from the revised Ackley's path in experiments to compare BSS with FGP, TGP, and DSS. The dimension of the revised Ackley's path was fixed at $d = 10$ but two data set sizes were used: $N = 1000$ and $N = 5000$. These data set sizes are moderate, so that FGP and TGP could handle them. Figure 2 illustrates the results of the proposed method as well as that of the other three methods.

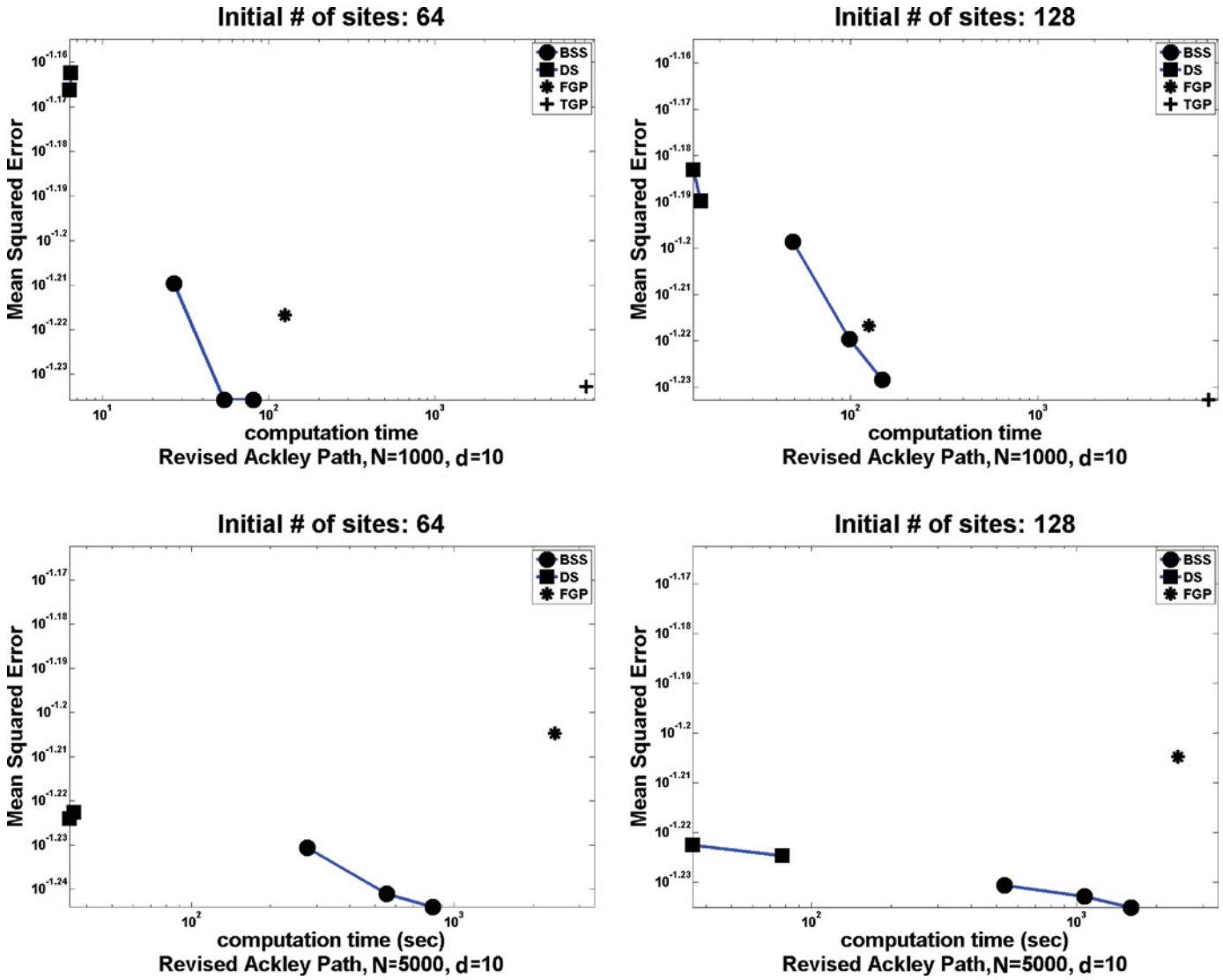

Fig. 2. Top: the results of BSS comparing with DSS, FGP, and TGP for the revised Ackley's path with $d = 10$ and $N = 1000$. In the left-hand-side plot, the number of initial sites for BSS and the (fixed) number of sites for DSS are 64, and on the right-hand side they are 128. Bottom: the results of the BSS comparing with DSS and FGP for the revised Ackley's path with $d = 10$ and $N = 5000$. In the left-hand-side plot, the number of initial sites for BSS and the (fixed) number of sites for DSS are 64, and on the right-hand side they are 128.

Table 1. Summary of the results presented in Fig. 2

	$N = 1000, d = 10$		$N = 5000, d = 10$	
Algorithm	*Computation time (s)*	*MSE*	*Computation time (s)*	*MSE*
BSS (64)	26.9	0.0617	277.1	0.0588
	53.9	0.0581	554.1	0.0574
	80.8	0.0581	831.1	0.0570
DSS (64)	6.4	0.0685	14.4	0.0656
	6.3	0.0688	15.8	0.0646
BSS (128)	49.2	0.0633	534.3	0.0588
	98.5	0.0603	1068.5	0.0585
	147.6	0.0591	1602.8	0.0582
DSS (128)	35.8	0.0597	77.6	0.0596
	35.8	0.0599	78.2	0.0593
FGP	125.2	0.0607	2417.5	0.0624
TGP	8145.0	0.0585	N/A	N/A

For BSS, three MSEs are presented in Fig. 2, each of which corresponds to some pre-determined RJMCMC chain length. As the length of the chain increases, the MSE in general decreases. For DSS, the different results are due to the different initial locations (randomly selected) used in the gradient-based optimization. For TGP, we used the default parameter settings in its `R` package. The top part of the figure presents the results for $N = 1000$. As evident in the figure, BSS produces MSE results close to those of FGP while requiring less computer time. When we allow BSS to spend nearly the same amount of time as FGP, it can produce smaller MSE values. DSS here uses a fixed number of sites, the same as the initial number of sites used in BSS, and then tries to find the optimal locations of those sites. As we see for this case, BSS obviously spends more time but produces more accurate predictions than does DSS. TGP produces very competitive MSE results but takes the longest time. The bottom part of the figure presents the results for $N = 5000$, at which data set size TGP takes too long to run, so the figure includes only the results of BSS, DSS, and FGP. The observations made earlier also apply here. In fact, compared with FGP, BSS performs noticeably better in less time. This could be due to the fact that the dimension of the data set ($d = 10$) is relatively high, which makes it difficult for FGP to learn the hyperparameters and, consequently, a poor estimate of hyperparameter hinders its performance. A summary of these results is also presented in Table 1.

We also ran an experiment to compare BSS with DSS using the real, generally larger-sized data sets; for the data sets larger than 1000 data points, FGP and TGP were too computationally expensive to run, so that we do not include FGP and TGP in the subsequent comparisons. To reach a more definite conclusion, we ran a five-fold cross-validation. The five-fold cross-validation provides average MSE/PLSM values as well as their standard deviations from the five trials.

Table 2 presents the MSE results for BSS and DSS when both are applied to the four real data sets. As shown in the table, BSS always produces a smaller average MSE than DSS: on the two spatial data cases the reduction in MSE is impressive, around two-fold smaller than that of DSS; on Sarcos data, BSS provides a remarkable 35% decrease in the average MSE, and on Abalone data, the two methods performed similarly, especially considering the standard deviation of the MSE.

Table 3 shows the PLSMs of the two methods for the data sets used in this article. It is observed that except for the Abalone data set, the BSS has a significantly smaller average PLSM than that of the DSS. Recall that PLSM measures the combined effect from the accuracy of the mean prediction and the predictive variance. Smaller PLSM and MSE values for the Sarcos, TCO, and MOD08-CL data sets are strong indicators that BSS outperforms DSS not only in terms of mean prediction but also with lower overall uncertainty. On Abalone data, the two methods perform similarly: BSS and DSS have almost indistinguishable MSEs but BSS has a slightly worse PLSM.

Admittedly, the improvement in accuracy by BSS comes at the cost of more computation time. The computation times of BSS, however, still reside in a region desirable for practical purposes. For example, BSS produces the results in less than 8 minutes for the Abalone data, 230 minutes for the Sarcos data, and less than 100 minutes for both the MOD08-CL and TCO data sets. Corresponding computation times for DSS are 10 seconds for Abalone, 8 minutes for Sarcos, and around 3 minutes for the MOD08-CL and TCO data sets. Should FGP be applied to a data set of a size similar to TCO, based on extrapolation from FGP's run times for solving smaller data sets, it would take more than 20 days.

4.4. *Sensitivity analysis*

Recall that we embed an optimization procedure within the RJMCMC moves. We ran experiments to investigate how

Table 2. Comparing BSS with DSS in terms of MSE. The numbers in the parentheses are standard deviations

Data set	*Dimension (d)*	*Number of data points (N)*	*BSS*	*DSS*
Abalone	7	4177	4.4081 (0.2018)	4.4454 (0.2008)
Sarcos	27	48 933	0.0558 (0.0082)	0.0754 (0.0059)
MOD08-CL	2	64 800	0.0058 (0.0004)	0.0147 (0.0010)
TCO	2	48 331	197.8 (37.4)	337.2 (37.5)

Table 3. PLSM values for $M = 32$, and for BSS, the MCMC chain runs 2000 iterations with 1000 burn-in iterations. The numbers in the parentheses are standard deviations

Data	*PLSM-BSS*	*PLSM-DSS*
Abalone	8600.1 (40.13)	8461.1 (23.1)
Sarcos	−7039.3 (1155.8)	−2123.2 (1007.6)
MOD08-CL	−92 458.3 (2608.6)	−44 563.2 (1793.5)
TCO	177 695.2 (4112.2)	206 905.6 (3040.9)

the running of the optimization would affect the results of BSS.

There were two parameters involved: one was the number of MCMC iterations between two consecutive optimizations of hyperparameters, which we denoted by κ, and the second parameter, intuitively speaking, concerned how "well" we performed the optimization, which can be characterized by the number of gradient steps used in the optimization procedure. The number of gradient steps was denoted by l in DSS; here we adopt the same notation. Using the revised Ackley's path case with $d = 2$ and $N = 10\,000$, we ran an ANalysis Of VAriance (ANOVA) taking κ and l as the factors. The value of κ was chosen from seven levels {1, 5, 10, 25, 50, 75, 100} and l was chosen from six levels {10, 20, 40, 100, 150, 200}. Note that in DSS, the number of gradient search iterations was generally fixed around 200. The response values were the MSE value under each combination of factors. The value of λ was randomized so it would not have a significant effect on the responses. We ran three replications in a full factorial design. Table 4 shows the resulting ANOVA table.

The result in Table 4 based on 5000 MCMC iterations. This ANOVA table suggests none of the factors is significant under $\alpha = 0.05$. However, among the parameters investigated, the parameter l has the smallest p-value. Based on other experiments conducted on the same data set, l can be found to be significant under $\alpha = 0.05$ when

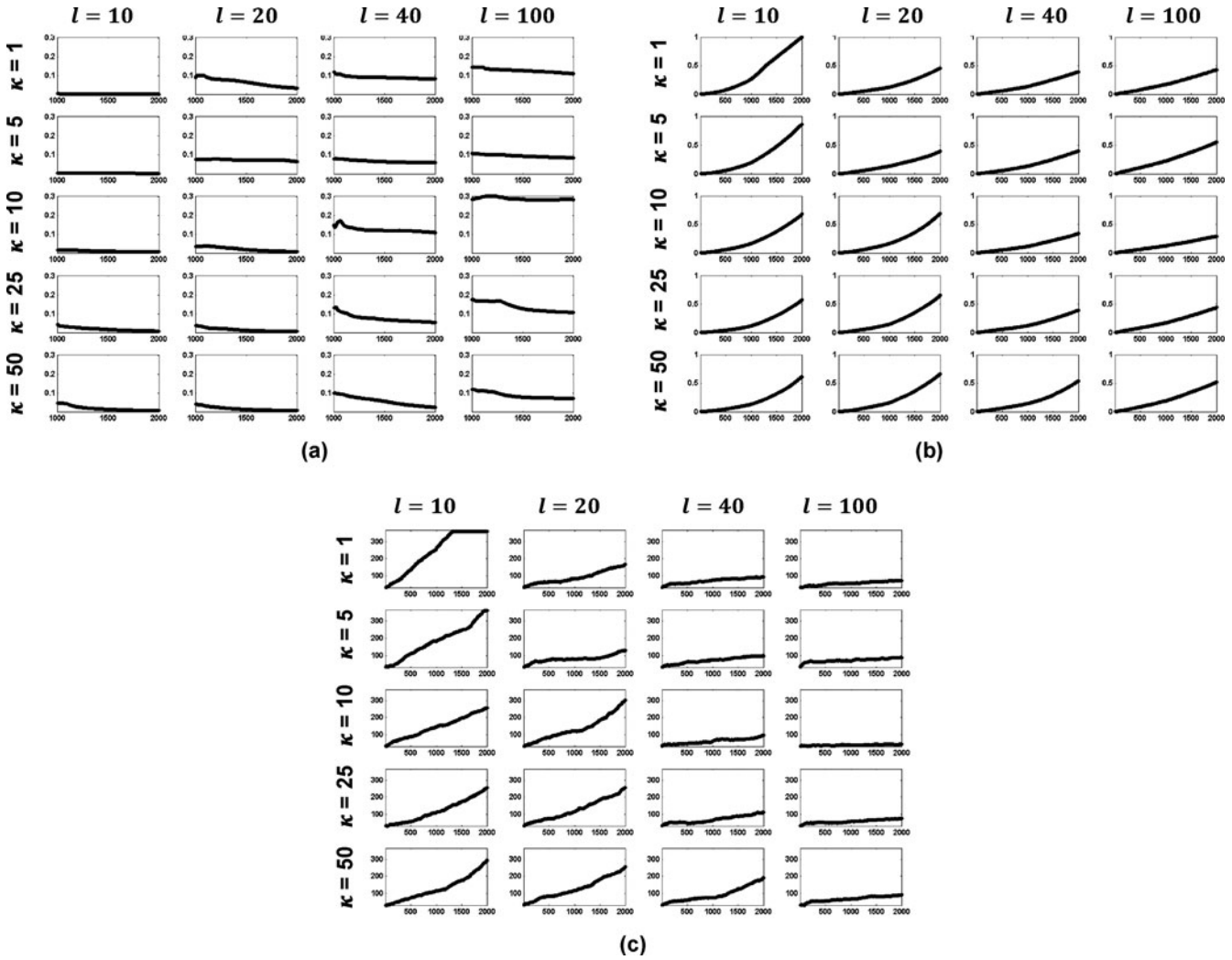

Fig. 3. (a) MSE versus MCMC iterations (after burn-in period) for different values of κ and l for the revised Ackley's path with $d = 2$ and $N = 10\,000$, (b) normalized computation time versus MCMC iterations, and (c) number of sites versus MCMC iterations.

Table 4. ANOVA analysis for revised Ackley path with $d = 2$, $N = 10\,000$ and 5000 MCMC iterations

Source	*Sum of squares*	*Degrees of freedom*	*Mean of squares*	*F*	*Prob > F*
κ	0.0046	6	0.0008	0.9850	0.4350
l	0.0053	5	0.0011	1.3397	0.2467
$\kappa \times l$	0.0156	30	0.0005	0.6630	0.9140
Error	0.2963	378	0.0008		
Total	0.3218	419			

shorter MCMC chains are used. We believe that this analysis suggests that if the length of the MCMC chain is long enough, the BSS method becomes less sensitive (or insensitive) to the change in parameters. This conclusion is supported by most of the other data sets we used.

To gain more insight concerning the effect of the parameters, we decided to look further into the behavior of the algorithm for different values of κ and l. Figure 3 shows the change in the MSE, the normalized computation time, and the number of sites for different values of κ and l as the chain evolves, based on an MCMC chain truncated at 2000 for the revised Ackley's path with $d = 2$ and $N = 10\ 000$. The normalized computation time is the computation time under a combination of κ and l, normalized by dividing the longest computation time among all possible κ–l combinations.

We would like to make a number of observations about Fig. 3.

1. Intuitively people might think that a larger l leads to a smaller MSE since a larger l means a deeper optimization of θ at each iteration. In reality, it turns out that a smaller l helps reduce the MSE more. The reason behind this is because a model's MSE depends much more on the number and locations of the sites than the optimization of θ. Long iterations in the optimization routine could overfit the data with a smaller number of sites and a refined θ, which may very well end up with a higher MSE.
2. Given the previous observation, in order to get smaller MSE values, people would understandably use a smaller l (e.g., $l = 10$). The other factor to consider is the computation. On the surface, a smaller l could mean a fast computation, and pairing with the small l, a large κ should be chosen to further reduce the computation. Although the large κ choice is generally correct, a small l does not necessarily lead to fast computation. For example, for the combination $\kappa = 1$ and $l = 10$ the algorithm has a longer computation time and results in a smaller MSE, compared with $\kappa = 1$ and $l = 100$. The reason is again the number of sites the algorithm chooses, and the computation depends much more on the number of sites than the value of l. Generally speaking, a smaller l causes the selection of a larger number of sites, leading to a smaller MSE but causing a longer computation. This can be seen by comparing Figs. 3(b) and 3(c). Our experience indicates that a general practice is to choose a relatively large l, say $l = 40$, paired with a large κ, say $\kappa = 25$, that can arrive at a good compromise between prediction quality and computation expense.

5. Conclusions and summary

This article presents an approximation algorithm to reduce the computation time of GP regression when dealing with large data sets. We tackle the problem by trying to approximate the likelihood function using a set of artificial data points and labeling them as "sites." We devise a BSS method and solve it using the RJMCMC algorithm to simultaneously find the number of sites and their locations. Our method can handle large data sets with general dimensions and outperforms the DSS method, which decides the locations of sites but with the site number fixed *a priori*.

As evident in the case studies presented in Section 4, the proposed BSS method produces similar or even smaller MSE values compared with FGP, while being able to do so faster. It can handle large data sets that FGP is not practically able to handle, while producing MSEs smaller than the DSS method. The computation time related to BSS can be reduced further by reducing the number of MCMC iterations in RJMCMC. Understandably, this may come at the expense of a decrease in prediction accuracy. Our current analysis indicates that with appropriate priors chosen, BSS generally provides a good trade-off between the two conflicting objectives.

There is still room for improvement in the proposed method. One possible improvement, as we mentioned before, is a full Bayesian treatment that updates θ at the same time as the site locations $\tilde{\mathbf{x}}$s, as opposed to using the gradient method for θ within the RJMCMC iterations. That is to say, in each birth, death, or exchange step, in addition to proposing a value for $\tilde{\mathbf{x}}$, we can also propose a value for θ and use the ratio test to either accept or reject it. One issue that we have not yet resolved is what proposal distribution to use. For the typical ones we have tested, they have not worked effectively. That is certainly an issue worthy of continuing efforts of exploration. Another improvement is for BSS to choose the sites from a continuous subspace, as opposed to from a discretized subspace in its current version. We believe that lifting the site selection restriction can particularly help the performance of BSS in the high-dimensional data problems.

Funding

Arash Pourhabib and Yu Ding were supported in part by NSF grants CMMI-0926803 and CMMI-1000088; Yu Ding was also supported by the NSF grant CMMI-0726939; Faming Liang's research was partially supported by NSF grants CMMI-0926803, DMS-1007457,

and DMS-1106494 and an award (KUS-C1-016-04) made by King Abdullah University of Science and Technology.

References

Furrer, R., Genton, M.G. and Nychka, D. (2006) Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, **15**, 502–523.

Gneiting, T. (2002) Compactly supported correlation funtions. *Journal of Multivariate Analysis*, **83**(2), 493–508.

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

Green, P.J. (1995) Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, **82**, 711–732.

Hoeting, J.A., Madigan, D., Raftery, A.E. and Volinsky, C.T. (1999) Bayesian model averaging: a tutorial. *Statistical Science*, **14**(4), 382–417.

Joseph, V. and Kang, L. (2011) Regression-based inverse distance weighting with applications to computer experiments. *Technometrics*, **53**, 254 –265.

Liang, F., Liu, C. and Carroll, R. (2010) *Advanced Markov Chain Monte Carlo Methods: Learning from Past Samples*, John Wiley & Sons, Hoboken, NJ.

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**, 1697–1728.

Park, J. and Liang, F. (2013) A Prediction-oriented Bayesian site approach for large spatial data. *Journal of Statistical Research*, **47**, 11–30.

Quiñonero-Candela, J. and Rasmussen, C.E. (2005) A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, **6**, 1939–1959.

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

Seeger, M., Williams, C.K.I. and Lawrence, N.D. (2003) Fast forward selection to speed up sparse Gaussian process regression, Presented at the Workshop on Artificial Intelligence and Statistics 9, January 3–6, 2003, Key West, FL.

Snelson, E. (2007) Flexible and efficient Gaussian process models for machine learning. Ph.D. Thesis, Gatsby Computational Neuroscience Unit, University College London, London, UK.

Snelson, E. and Ghahramani, Z. (2006) Sparse Gaussian processes using pseudo-inputs. *Advances in Neural Information Processing Systems*, **18**, 1257–1264.

Snelson, E. and Ghahramani, Z. (2007) Local and global sparse Gaussian process approximations. Presented at the International Conference on Artifical Intelligence and Statistics 11, March 21–24, 2007, San Juan, Puerto Rico.

UCI. (2010) UCI machine learning repository. Available from http://archive.ics.uci.edu/ml (accessed January 2014).

Wu, C.F.J. and Hamada, M.S. (2009) *Experiments: Planning, Analysis, and Optimization*, second edition, John Wiley & Sons, Hoboken, NJ.

Biographies

Arash Pourhabib received his B.S. in Industrial Engineering from Sharif University of Technology in 2008. He is currently a Ph.D. candidate in the Department of Industrial and Systems Engineering at Texas A&M University. He has been a Research Assistant in the Advanced Metrology Laboratory since January 2009. His research interests are in the area of system informatics and control and statistical machine learning. He is a member of INFORMS.

Faming Liang received a B.S. degree in Statistics from Fudan University, Shanghai, China, in 1992, and a Ph.D. degree in Statistics from the Chinese University of Hong Kong, Hong Kong, China, in 1997. He is currently a Professor in the Department of Statistics, Texas A&M University. He is a fellow of the American Statistical Association and the Institute of Mathematical Statistics and an elected member of the International Statistical Institute. His current research interests include Markov chain Monte Carlo methods, big data analysis, bioinformatics, and machine learning.

Yu Ding received his B.S. degree in Precision Engineering from the University of Science and Technology of China in 1993; his M.S. degree in Precision Instruments from Tsinghua University, China, in 1996; his M.S. degree in Mechanical Engineering from Pennsylvania State University in 1998; and his Ph.D. degree in Mechanical Engineering from the University of Michigan in 2001. He is currently the Mike & Sugar Barnes Professor of Industrial and Systems Engineering and Professor of Electrical and Computer Engineering at Texas A&M University. His research interests are in the area of system informatics and quality and reliability engineering. He currently serves as a Department Editor of *IIE Transactions*. He is a member of IIE, a senior member of IEEE, and a member of INFORMS and ASME.