IEICE TRANS. INF. & SYST., VOL.E93–D, NO.10 OCTOBER 2010

PAPER Special Section on Data Mining and Statistical Science

Gaussian Process Regression with Measurement Error

Yukito IBA[†a)] ***and*** **Shotaro AKAHO**[††b)]**,** ***Members***

SUMMARY Regression analysis that incorporates measurement errors in input variables is important in various applications. In this study, we consider this problem within a framework of Gaussian process regression. The proposed method can also be regarded as a generalization of kernel regression to include errors in regressors. A Markov chain Monte Carlo method is introduced, where the infinite-dimensionality of Gaussian process is dealt with a trick to exchange the order of sampling of the latent variable and the function. The proposed method is tested with artificial data.

key words: *measurement error, errors in input variables, kernel, Gaussian process, Bayes, Markov chain Monte Carlo*

1. Introduction

Regression analysis that incorporates errors in both input and output variables is often required in the analysis of experimental data; a recent example in the brain science is found in [1]. It is also useful in a variety of situations where only partial observation of input variables is allowed; an illustrative example is when the input is an image observed by cameras with limited resolution. Another example is inference based on "binned" or "coarse grained" data, where only a summary of data is disclosed; it is often appeared in practical applications and becoming important in relation to data anonymization.

Models for such an analysis are called as *measurement error models* in the literature of statistical science [2]–[8]. This subject, however, seems not fully treated in the context of machine learning. Also, treatment of measurement error models for the multi-dimensional input variables still presents an open problem.

If we ignore errors in input variables, it can cause biases in estimated parameters and functional relations between inputs and outputs. Specifically, using models without errors in input variables, homogeneous noises in input variables are interpreted as noises whose levels are dependent on the local slope of the estimated functions; in dealing with such noises, explicit modeling of errors in input variables seems the best way. In reference [6], examples are given where conventional regression is outperformed by the one with errors in input variables, even with a cross-validated choice of the hyperparameter; a corresponding result is shown by numerical experiments in Sec. 4 in this paper.

The aim of this paper is to develop an algorithm for Gaussian process regression that allows for errors in input variables. This can also be regarded as a generalization of *kernel regression* to include errors in regressors. Artificial discretization is not required in the proposed algorithm. This property makes the algorithm computationally efficient in problems with missing observations and opens the possibility of treating problems with multi-dimensional input variables. In addition, priors only represented by infinite-order lag operators can be introduced in a natural way; for example, a process associated with a Gaussian kernel can easily be handled.

A few studies [1], [6] in the literature treat measurement error problems in a Bayesian framework using smoothness priors*. However, they are essentially limited to smoothness priors with second-order derivatives; additional complexity should be introduced to treat infinite-order priors. Also, applications to multi-dimensional problems will be difficult with these methods.

Considering great flexibility of kernel regression, one might think that adaptation of kernels can easily handle errors in input variables. It will not be always true, as we already discussed in this section and will be shown in Sec. 4. Construction of kernels becomes even more difficult, when noises in input variables have levels varying with the values of inputs or have correlations to noises in outputs; in some cases, the conditional probability of input variables may be non-Gaussian. On the other hand, representing everything as a kernel, it spoils an advantage of Gaussian process regression that likelihood and function space of the regressor are separately specified. Using measurement error models, tailor-made modeling becomes easier at the expense of increased computational costs.

It is straightforward to construct measurement error models with a generic Gaussian process. On the other hand, it is not trivial to make inference about these models, because a Gaussian process is infinite-dimensional in the sense that infinite number of points are required to specify a sam-

Manuscript received November 30, 2009.
Manuscript revised April 2, 2010.

[†]The author is with the Department of Statistical Modeling, The Institute of Statistical Mathematics, and Department of Statistical Science, Tachikawa-shi, 190–8562 Japan.
[††]The author is with the Human Technology Research Institute, National Institute of Advanced Industrial Science and Technology, Tsukuba-shi, 305–8568 Japan.
a) E-mail: iba@ism.ac.jp
b) E-mail: s.akaho@aist.go.jp
DOI: 10.1587/transinf.E93.D.2680

*For other attempts to treat measurement error problems in nonparametric frameworks, see, [4], [6], [9] and references therein. The term "kernel" used in these references does not necessarily mean reproducing kernels; it is usually used in a broader sense.

ple. Hence this paper mostly focuses on computational problems, especially how we can perform the computation using finite-dimensional representations.

In conventional kernel regression, the representer theorem is used to reduce a functional regression problem to a finite-dimensional one; if we denote the values of the input variables by $\{x_i\}$, the maximum of the posterior distribution defined by the model and data is found in the space of a linear combination $\sum_i a_i k(\cdot, x_i)$, where k is the reproducing kernel associated with the Gaussian process. In a measurement error problem, $\{x_i\}$ are in themselves variables to be estimated; if we denote them as $\{z_i\}$, the space spanned by $\{k(\cdot, z_i)\}$ is no more finite-dimensional, when we consider all possible values of $\{z_i\}$. If we are interested only in the joint MAP estimate of the input variables $\{z_i\}$ and the function f (i.e., the pair of $\{z_i\}$ and f that maximize the posterior density (8) in Sec. 2), it is possible to use the representer theorem within an alternate maximization of $\{z_i\}$ and f. However, the joint MAP estimate is often not optimal in measurement error models [5], [6]. Then, we want to compute, for example, the posterior mean of f averaging over $\{z_i\}$ as an estimate of f, but the representer theorem is not directly applicable in this case, because we cannot specify the values of $\{z_i\}$.

To avoid this difficulty, we develop a Markov chain Monte Carlo algorithm (MCMC) specialized to the problem. A naive application of MCMC prohibited by apparent infinite-dimensionality of the problem. We will show, however, that a simple trick of exchanging the order of samplings makes it possible to reduce the problem to an effectively finite-dimensional one; this idea is explained in Sec. 3.1.

The rest of the paper is organized as follows: In Sec. 2, a statistical model for Gaussian process regression is introduced. In Sec. 3, an MCMC algorithm is proposed for the inference with the model. In Sec. 4, results of numerical experiments using the proposed algorithm are shown. Section 5 is devoted to summary and discussions.

2. Model

Let us consider estimation of a function f from samples $\{(x_i, y_i)\}$; observed values of the input variables and output variables are denoted by $x = \{x_i\}$ and $y = \{y_i\}$, and the index i of the samples is assumed to take a value $1, \cdots, n$. Unobserved true values of input variables are denoted by $z = \{z_i\}$, which are latent variables of the model.

Here we formulate the problem within a hierarchical Bayesian framework [10]–[12]; the model is defined by the combined probability density as

$$p(x, y, z, f) = p(y|f, z)p(x|z)p(z)p(f), \tag{1}$$

whose components are given in the following subsections Secs. 2.1 and 2.2. With this combined density, the posterior density is derived by the Bayes formula in Sec. 2.3.

2.1 Regression with Measurement Error

As in a usual regression problem, we assume a Gaussian density

$$p(y|z, f) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(y_i - f(z_i))^2}{2\sigma_y^2}\right) \tag{2}$$

to represent an observation process of the output variables.

In this study, another density $p(x|z)$ is introduced to represent errors in the input variables. Hereafter, we assume independence of the observations and express $p(x|z)$ as

$$p(x|z) = \prod_{i=1}^{n} p(x_i|z_i). \tag{3}$$

Here, we restrict ourselves to the cases, where $x_i \in \mathbb{R}^d$, $z_i \in \mathbb{R}^d$, and a Gaussian density

$$p(x_i|z_i) = \frac{1}{(2\pi\sigma_x^2)^{d/2}} \exp\left(-\frac{\|x_i - z_i\|^2}{2\sigma_x^2}\right) \tag{4}$$

is assumed, here $\|\ \|$ is the Euclidean norm. In principle, however, any case with a non-Gaussian density $p(x_i|z_i)$, or even with discrete or graphical input variables x and z can be treated by using suitable modifications of the algorithm in Sec. 3. An example of other assumptions on $p(x_i|z_i)$ is given in Sec. 5, which corresponds to coarse-grained data.

The variance σ_x^2 and σ_y^2 in (2) and (4) is assumed to be known. Difficulty arises, when we estimate *both* of σ_x^2 and σ_y^2 using the same set of data from which we estimate f; basics of identifiability and consistency problems for models with errors in input variables are explained in [2]. In general, problems caused by large number of hidden variables $\{z_i\}$ have been studied in statistics since Neyman and Scott [13]; their information geometrical aspects are discussed in [5]. In this paper, however, we will not discuss them further and focus on computational aspects of the problem.

The prior $p(z)$ is assumed to be the uniform density, although it is also not essential for the algorithm. Combined with the prior $p(f)$ discussed in the next subsection, the densities (2), (3) and $p(z)$ specify a measurement error model.

2.2 Gaussian Process Prior

The prior $p(f)$ is defined as a Gaussian process [12] with zero mean $\mathrm{E}[f(s)] = 0$ and the covariance function

$$\mathrm{Cov}[f(s), f(s')] = k(s, s'). \tag{5}$$

An inner product $(\cdot, \cdot)_{\mathcal{H}}$ associated with the process can be introduced, which satisfies

$$(f, g)_{\mathcal{H}} = \sum_{i,j} a_i b_j\, k(\xi_i, \zeta_j), \tag{6}$$

when $f(\cdot) = \sum_i a_i k(\cdot, \xi_i)$ and $g(\cdot) = \sum_i b_i k(\cdot, \zeta_i)$. A norm $\|\cdot\|_{\mathcal{H}}$

is also defined by $\|f\|_{\mathcal{H}}^2 = (f, f)_{\mathcal{H}}$. The Hilbert space $\mathcal{H}$ that consists of f satisfying $\|f\|_{\mathcal{H}} < +\infty$ is called a reproducing kernel Hilbert space (RKHS) [12], [14].

Typically, $\|f\|_{\mathcal{H}} = +\infty$ with probability one for a random sample f from a Gaussian process. However, we can apply some regularization[†] to f that makes the norm finite, $\|f\|_{\mathcal{H}} < +\infty$, and redefine the measure on the set $\tilde{\mathcal{H}}$ of regularized f's. In most of applications[††], we can safely identify $\tilde{\mathcal{H}}$ with $\mathcal{H}$; we will write $\mathcal{H}$ in place of $\tilde{\mathcal{H}}$ through the rest of the paper.

With this identification, the Gaussian process prior is effectively written as

$$p(f) \propto \exp\left(-\frac{1}{2}\|f\|_{\mathcal{H}}^2\right), \ f \in \mathcal{H}, \tag{7}$$

which is used to construct the posterior density in the next subsection.

2.3 Posterior Density

Applying the Bayes formula to the joint density (1) defined by the densities (2), (3), and (7), the posterior density is expressed as

$$p(f, z|x, y) \propto \exp\left(\sum_i \left\{-\frac{(y_i - f(z_i))^2}{2\sigma_y^2} + \log p(x_i|z_i)\right\} - \frac{1}{2}\|f\|_{\mathcal{H}}^2\right). \tag{8}$$

Thus, our problem is reduced to the sampling of $f \in \mathcal{H}$ and $z \in \mathbb{R}^d$ from the posterior (8).

3. Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) is widely used in modern Bayesian statistics for sampling posterior distributions [8], [10], [15], [16]; applications in measurement error models are found in [1], [4], [6], [8]. In this section, we introduce a method for realizing MCMC for the model on a computer without artificial discretization or truncation.

3.1 Basic Idea

In the present case, a naive version of MCMC for sampling the posterior (8) is defined by alternate updates of f and z. Starting from an initial value $z^{(0)}$ of z, the algorithm produces a series $(z^{(1)}, f^{(1)}), (z^{(2)}, f^{(2)}), \cdots$ of samples; the MCMC algorithm is designed that averages over them coincide with the corresponding posterior averages.

Given the density $q(\cdot|z^{(t)})$ of the proposal distribution used in the Metropolis-Hastings step, a step of the naive MCMC is described as

1. Sample $f^{(t+1)}$ from the density $p(f|x, y, z^{(t)})$.
2. Generate a candidate z^* of the next value of z from the density $q(\cdot|z^{(t)})$.
3. Calculate the Metropolis-Hastings ratio
$$r = \frac{p(z^*|f^{(t+1)}, x, y)\, q(z^{(t)}|z^*)}{p(z^{(t)}|f^{(t+1)}, x, y)\, q(z^*|z^{(t)})}. \tag{9}$$
Draw a uniform random number $rnd \in (0, 1]$.
If $rnd < r$ then $z^{(t+1)} = z^*$, else $z^{(t+1)} = z^{(t)}$.

In this case, however, the sampling of $f^{(t+1)}$ is not feasible on a computer, because f is virtually infinite-dimensional.

A key observation is that we do not need entire f in the procedure 3. Let us consider a case that we change a single component z_k at one time and the probability of the candidate z_k^* at step t is defined by the density $q(z_k^*|z_k^{(t)})$. Then, the computation of r using (9) in the procedure 3 requires only the values $f^{(t+1)}(z_k^{(t)})$ and $f^{(t+1)}(z_k^*)$, because r is expressed as

$$\exp\left(-\frac{\left\{(y_k - f^{(t+1)}(z_k^*))^2 - (y_k - f^{(t+1)}(z_k))^2\right\}}{2\sigma_y^2}\right) \times \frac{p(z_k^*|x_k)\, q(z_k^{(t)}|z_k^*)}{p(z_k^{(t)}|x_k)\, q(z_k^*|z_k^{(t)})}. \tag{10}$$

Then, the exchange of the order of procedures 1 and 2 resolves the difficulty, that is, we can produce exactly the same sequence of $z^{(1)}, z^{(2)}, \cdots$ using the following procedures.

1'. Choose k randomly and generate a candidate z_k^* of the next value of z_k from the density $q(\cdot|z_k^{(t)})$.
2'. Sample $f^{(t+1)}(z_k^{(t)})$ and $f^{(t+1)}(z_k^*)$ from the density $p(f(z_k^{(t)}), f(z_k^*)|x, y, z^{(t)})$.
3'. Calculate the Metropolis-Hastings ratio r using (10).
Draw a uniform random number $rnd \in (0, 1]$.
If $rnd < r$ then $z^{(t+1)} = z^*$ else $z^{(t+1)} = z^{(t)}$.

Here, procedures 1' and 2' correspond to the procedures 2 and 1 in the previous version of the algorithm, respectively. In this scheme, we do not need to compute the function f; resampling of just two values $f^{(t+1)}(z_k^{(t)})$ and $f^{(t+1)}(z_k^*)$ is enough for our purpose of updating z_k. This makes the algorithm finite and feasible on a computer. The explicit form of the density $p(f(z_k^{(t)}), f(z_k^*)|x, y, z^{(t)})$ will be derived in the next subsection.

Now that we can generate a sequence $z^{(1)}, z^{(2)}, \cdots$ by MCMC, how we can sample required values of f? Assume that we want to sample the values $\{f(\tilde{x}_i)\}$ of f on a set of points $\tilde{x} = \{\tilde{x}_i\}$, $i = 1, \cdots, m$; hereafter we call $\{\tilde{x}_i\}$ as *observation points*. Then, the sampling of $\{f(\tilde{x}_i)\}$ is realized using the fact that the joint density of $\{f(\tilde{x}_i)\}$ and $\{f(z_i)\}$ is multivariate Gaussian; similar methods are found in the literature [17].

[†]It is realized by the truncation of an orthogonal expansion of f, which introduces a cutoff of high frequency part; an arbitrary large cutoff is enough for our purpose of defining the expression (8).

[††]Precisely speaking, the constraint that $|x_i - x_j| > \epsilon$ for all pairs of i and j is required for measurement error models, where ϵ is a small constant. It will, however, not affect the results in most cases.

Before explaining the details, we remark that the idea explained in this subsection can be generalized to the cases where we try to change more than one components of z at one time. For example, if we change all components of z simultaneously, the Metropolis-Hastings ratio r is expressed using $f(z_i^{(t)})$ and $f(z_i^*)$ for $i = 1, \cdots, n$. Even in this case, we can construct an algorithm similar to that we discuss in the following sections, where the vector $\mathbf{f}$ in (12) has $2n$ components.

3.2 More on Sampling of f

In this subsection, we explain details of the procedure 2' as well as how to integrate the sampling of $\{f(\tilde{x}_i)\}$ into the MCMC algorithm. We refer to the appendix for another look at the derivation in this subsection, which will also clarify the relation to the representer theorem.

To give a concise and unified description, we define $(n + m + 1)$-dimensional vectors $\tilde{z}$ and $\tilde{\mathbf{f}}$ by

$$\tilde{z} = (z_1^{(t)}, \cdots, z_n^{(t)}, z_k^*, \tilde{x}_1, \cdots, \tilde{x}_m), \tag{11}$$

$$\tilde{\mathbf{f}} = (f(z_1^{(t)}), \cdots, f(z_n^{(t)}), f(z_k^*), f(\tilde{x}_1), \cdots, f(\tilde{x}_m)). \tag{12}$$

The vector $\tilde{\mathbf{f}}$ contains everything that we need at step t, that is, the values of f at the inferred input points $\{z_i^{(t)}\}$, candidate z_k^*, and observation points $\{\tilde{x}_i\}$. Using this notation, the procedure 2' in the previous subsection is replaced with

2". Sample $\mathbf{f}$ from $p(\mathbf{f}|x, y, z^{(t)})$;

this includes the sampling of $\{f(\tilde{x}_i)\}$ at step t.

Now, the problem is specification of the conditional density $p(\mathbf{f}|x, y, z^{(t)})$. Using the expression (8), the conditional density of the function f is written as

$$p(f|x, y, z^{(t)}) \propto \exp\left(\sum_i \left\{-\frac{(y_i - f(z_i^{(t)}))^2}{2\sigma_y^2}\right\} - \frac{1}{2}\|f\|_{\mathcal{H}}^2\right). \tag{13}$$

Then, by using (5), the density of the vector $\tilde{\mathbf{f}}$ is given by

$$p(\tilde{\mathbf{f}}|x, y, z^{(t)}) \propto \exp\left(-\frac{1}{2\sigma_y^2}(y_+ - \tilde{\mathbf{f}})^T J(y_+ - \tilde{\mathbf{f}}) - \frac{1}{2}\tilde{\mathbf{f}}^T K^{-1}\tilde{\mathbf{f}}\right), \tag{14}$$

where

$$y_+ = (y_1, \cdots, y_n, 0, \cdots, 0)^T, \quad J = \begin{pmatrix} I_n & O \\ O & O \end{pmatrix}. \tag{15}$$

The matrix K in (14) denotes the Gram matrix associated with $\tilde{z}$, that is, the matrix whose (i, j)-component is given by $k(\tilde{z}_i, \tilde{z}_j)$, where $\tilde{z}_i$ is the i-th component of the vector $\tilde{z}$ defined by (11). The block structure of the matrix K is represented as

$$K = \begin{pmatrix} K_1 & K_2 & K_3 \\ K_2^T & k_* & K_4^T \\ K_3^T & K_4 & K_5 \end{pmatrix}, \tag{16}$$

where K_1 and K_5 are $n \times n$ and $m \times m$ symmetric matrix, respectively; K_2, K_3, K_4 are $n \times 1$, $n \times m$, and $m \times 1$ matrices; k_* is a scalar.

A routine calculation shows that the density $p(\tilde{\mathbf{f}}|x, y, z^{(t)})$ defined by (14) is a Gaussian density with the mean

$$\mu(z^{(t)}) = \begin{pmatrix} K_1 \\ K_2^T \\ K_3^T \end{pmatrix}(K_1 + \sigma_y^2 I_n)^{-1} y, \tag{17}$$

and the covariance matrix

$$V(z^{(t)}) = \left(K^{-1} + \frac{1}{\sigma_y^2} J\right)^{-1}. \tag{18}$$

Here, the dependence on $z^{(t)}$ arises through the definition of K. Note that $\mu(z^{(t)})$ of (17) coincides with the estimate by usual kernel regression with the given $z^{(t)}$.

Basically, (17) and (18) are enough for generating a sample from $p(\tilde{\mathbf{f}}|x, y, z^{(t)})$. However, if we use (18) directly, we often encounter a numerical difficulty, because the matrix K_1 usually has many near-zero eigenvalues; even with a mathematical proof that K_1 has full-rank, it can be highly ill-conditioned. To avoid this difficulty, the covariance matrix V is expressed as

$$V = L^T\left(I_{n+m+1} + \frac{1}{\sigma_y^2} LJL^T\right)^{-1} L, \tag{19}$$

where L is the Cholesky decomposition of K, which satisfies $K = L^T L$. Then, by substituting an incomplete Cholesky decomposition for the Cholesky decomposition, we have a stable and numerically efficient algorithm; this technique is commonly used in the field of kernel multivariate analysis [18] and provides controllable approximations with arbitrary accuracy.

3.3 Rao-Blackwellization

Until now, we discuss sampling of $\{f(\tilde{x}_i)\}$, the values of f on the observation points. If we are interested only in the posterior averages of $\{f(\tilde{x}_i)\}$, an efficient way is to calculate the averages of the $(n + 2)$-th $\sim (n + m + 1)$-th components of $\mu(z^{(t)})$ defined by (17); they are the posterior averages of $\{f(\tilde{x}_i)\}$ conditioned with $z^{(t)}$. Denoting these components as $\tilde{\mu}(z^{(t)})$,

$$\begin{aligned} \mathbb{E}(f(\tilde{x}_i)) &= \int\int f(\tilde{x}_i) p(f, z|x, y) df dz \\ &= \int \left\{\int f(\tilde{x}_i) p(f|x, y, z) df\right\} p(z|x, y) dz \\ &= \int \tilde{\mu}(z) p(z|x, y) dz \simeq \frac{1}{t_{\max}} \sum_{t=1}^{t_{\max}} \tilde{\mu}(z^{(t)}), \end{aligned} \tag{20}$$

holds, which justifies this method. Such an approach is called "Rao-Blackwellization" in statistical science [19], although it has been used in physics.

We can also calculate the variance of $f(\tilde{x}_i)$ in a similar way; it is represented as a sum of the average of conditional variance and the variance of conditional average, both of which can be computed using the proposed algorithm.

3.4 Summary of the Proposed Algorithm

Putting these pieces together, a step of the proposed algorithm is summarized as follows.

- Choose k randomly and generate the candidate z_k^* of the next value of z_k from $q(\cdot|z_k^t)$.
- Generate the Gram matrix K associated with the vector $\tilde{z}$ defined by (11). Note that only part of the elements of K need to be refreshed at one time.
- Calculate μ and V using (17) and (19). An incomplete Cholesky decomposition L of K is used in (19).
- Generate the vector $\mathbf{f}$ as a sample from the Gaussian distribution with the mean μ and covariance matrix V.
- Store the current value of $\{f(\tilde{x}_i)\}$ and/or μ.
- Calculate r using (10). Draw a uniform random number $rnd \in (0, 1]$. If $rnd < r$ then $z^{(t+1)} = z^*$ else $z^{(t+1)} = z^{(t)}$.
- $t = t + 1$.

Starting from an initial value of $z = z^{(0)}$ and $t = 1$, the above procedures are iterated prescribed times. Then, the obtained samples of $\{f(\tilde{x}_i)\}$ and/or μ are used to estimate posterior averages and their errors.

4. Experiments

We perform numerical experiments by using synthetic data of one-dimensional and two-dimensional input space in order to examine the performance of the proposed algorithm.

In all experiments, we use the Gaussian kernel

$$k(z, z') = \lambda \exp(-\beta \|z - z'\|^2) \tag{21}$$

with two hyperparameters λ and β.

4.1 One-Dimensional Case

4.1.1 Setting

Here we perform experiments for one-dimensional case, in which Gaussian noise is added to the input variable. $n = 50$ true values $\{z_i^{true}\}$ of the input variable are drawn independently from standard Gaussian distribution $\mathcal{N}[0, 1]$. Then the observed values $\{x_i\}$ of input are generated by adding independent Gaussian noise with variance $\sigma_x^2 = (0.3)^2$ to z_i^{true}.

As a target function, we choose the same function as in Berry et al.'s paper [6],

$$f(z) = \frac{\sin(\pi z/2)}{1 + 2z^2(\mathrm{sign}(z) + 1)} \tag{22}$$

and the observation y_i of the output is generated by adding Gaussian noise with the variance $\sigma_y^2 = (0.1)^2$ to the function value $f(z_i^{true})$ at z_i^{true}.

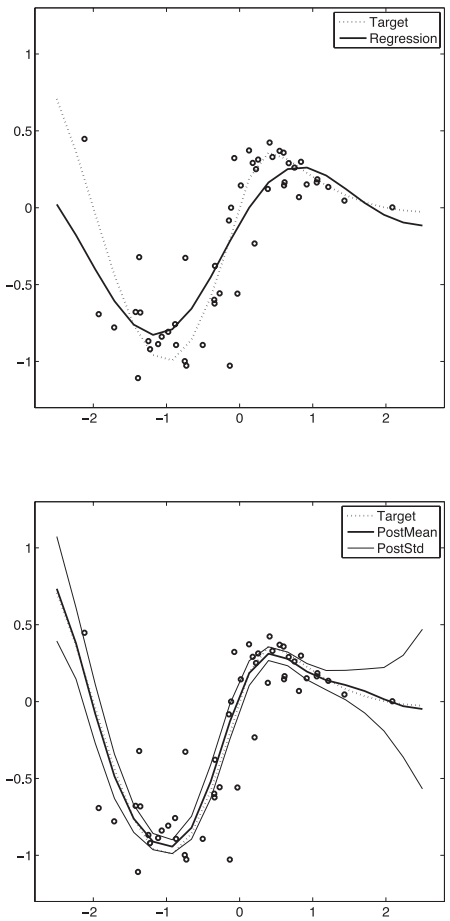


Fig. 1 Fitting example (one-dimensional case). Upper: the usual kernel regression optimized by CV. Lower: the proposed method (posterior mean with errorbar (=mean±std. dev.)). Dotted line represents the target function.

As a proposal distribution $q(\cdot\,|z_i^{(t)})$ for the candidate z_i^* of the i-th latent variable, we use a sampler that generates a sample independent of $z^{(t)}$ but dependent on x_i,

$$q(z_i|z_i^{(t)}) = \frac{1}{\sqrt{2\pi\sigma_x^2}} \exp\left(-\frac{\|z_i - x_i\|^2}{2\sigma_x^2}\right). \tag{23}$$

Initial value $z_i^{(0)}$ is chosen randomly with respect to $\mathcal{N}[x_i, (0.1)^2]$ which fluctuates around x_i.

Observation points are placed at regular intervals between -2.5 and 2.5 to evaluate the function values. Total number of the observation points are $m = 20$. Mean square loss measured at observation points is given by

$$\mathrm{LOSS} = \frac{1}{m} \sum_i (\hat{f}(\tilde{x}_i) - f(\tilde{x}_i))^2, \tag{24}$$

where $\hat{f}(\tilde{x}_i)$ is the estimated value of f at the observation point $\tilde{x}_i$.

4.1.2 Comparison to Usual Regression

Figure 1 shows an example of fitting. In this example, the

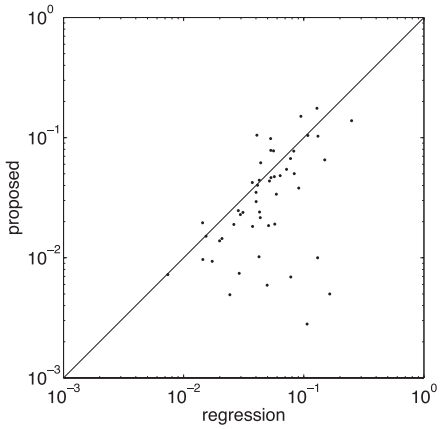


Fig. 2 Comparison of LOSS for 50 different datasets (log-log scale). Horizontal axis: usual kernel regression optimized by cross-validation; Vertical axis: the proposed method ($\lambda = \beta = 1$).

proposed algorithm estimates a solution closer to the target curve than the usual kernel regression without errors in the input variable.

The number of iterations are set to 480 MCMC cycles after 20 burn-in cycles, where we define a cycle by the updates of the whole samples, (thus one cycle includes n iteration of the steps defined in Sec. 3.4).

The hyperparameters are fixed to $\lambda = \beta = 1$ for the proposed algorithm, while they are optimized for the usual kernel regression using the cross-validation (CV). It is easy to calculate leave-one-out cross-validation error for usual kernel regression,

$$CV(\lambda,\beta) = \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - H_{ii}} \right)^2, \tag{25}$$

where the $n \times n$ matrix $H = (H_{ij})_{i,j=1,\ldots n}$ is defined by $H = K_1(K_1 + \sigma_y^2 I_n)^{-1}$ and $\hat{y} = (\hat{y}_i)_{i=1,\ldots,n}$ is an estimated value of $f(x_i)$ that can be calculated by $\hat{y} = Hy$. We calculate $CV(\lambda,\beta)$ for the grid points $(\lambda,\beta) \in \{0.1, 0.2, \ldots, 3.0\} \times \{0.1, 0.2, \ldots, 3.0\}$ and choose the hyperparameters that give the smallest CV value.

We compare the proposed algorithm with the kernel regression optimized by cross-validation; 50 datasets are generated with the same function (22) using different sets of Gaussian random numbers added to both inputs and outputs. The result is shown in Fig. 2. It is observed that the proposed algorithm outperforms the kernel regression on average.

4.1.3 Sensitivity against Parameters

Since we used a fixed setting of parameters λ and β for the proposed algorithm, we investigate the sensitivity of performance against the choice of parameters here. First, we examine the performance for 50 different datasets used in Fig. 2, changing the values of parameters (Fig. 3). Averaged value of LOSS over all the datasets for each parameter setting is summarized in Table 1. It can be seen that the performance is robust against the change of the parameter setting.

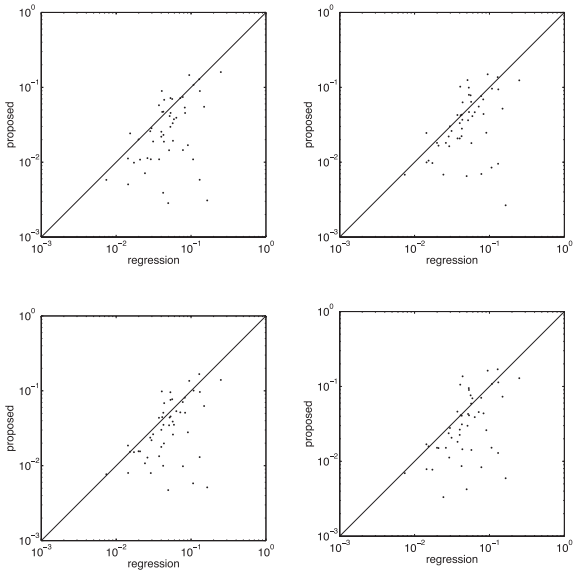


Fig. 3 Comparison of LOSS for 50 different datasets. (Upper left: $\lambda = 1, \beta = 0.5$, Upper right: $\lambda = 1, \beta = 1.5$), Lower left: $\lambda = 0.5, \beta = 1$, Upper right: $\lambda = 1.5, \beta = 1$). The number of iterations are 480 MCMC cycles after 20 burn-in cycles.

Table 1 Average and standard deviation of LOSS for 50 datasets.

			LOSS average	(std.)
kernel regression (CV opt.)			0.06167	(0.04576)
MCMC	($\lambda = 1$,	$\beta = 1$)	0.04321	(0.03949)
MCMC	($\lambda = 1$,	$\beta = 0.5$)	0.03978	(0.03732)
MCMC	($\lambda = 1$,	$\beta = 1.5$)	0.04519	(0.03769)
MCMC	($\lambda = 0.5$,	$\beta = 1$)	0.04366	(0.03833)
MCMC	($\lambda = 1.5$,	$\beta = 1$)	0.04535	(0.04318)

It is more preferable if we can apply cross-validation to the proposed method. We show some preliminary results here.

We examine the cross-validation of the proposed algorithm for a dataset with 50 samples. We perform randomized 5-fold cross-validation, in which each sample is chosen as a member of the validation set C_v randomly with probability 1/5. The validation samples are removed from the training set, and input variables x_i of the validation samples are added to the observation points, and the function value $\hat{f}(x_i)$ at x_i is estimated by MCMC. The validation error is calculated by

$$\frac{1}{|C_v|} \sum_{i \in C_v} (y_i - \hat{f}(x_i))^2 \tag{26}$$

where $|C_v|$ is the size of validation set. We obtain cross-validation error by averaging the validation errors for 10 different validation sets. Note that the above error should be $(y_i - \hat{f}(z_i))^2$ from the measurement error model standpoint, but it is impossible to estimate z_i without using y_i and it is not clear how much the validation error is biased if we use y_i to estimate z_i. The theoretical analysis of this issue remains as a future work.

The resulting cross-validation errors are shown in

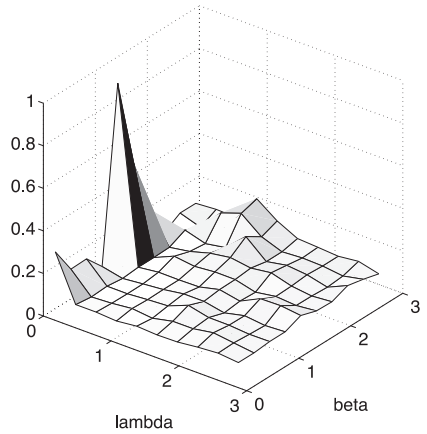


Fig. 4 Cross-validation error of the proposed method for various values of λ, β.

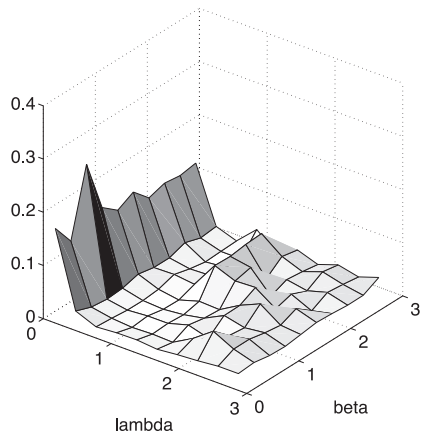


Fig. 5 LOSS of the proposed method for various values of λ, β.

Fig. 4. Although the CV error is minimized at $\lambda = 1, \beta = 0.4$ in this experiment, the error surface is very flat around the optimum, which is consistent with the fact that the performance is robust against the change of the parameter values.

To relate the cross-validation error with generalization error, we apply the proposed algorithm for the same set of parameters in which the whole dataset is used as training samples, and calculate LOSS (Fig. 5). In this figure, LOSS is minimized at $\lambda = 1.4, \beta = 1$, the error surface is also flat around the optimum.

The shapes of error surface of cross-validation error and LOSS are relatively similar, which suggests that we can use the cross-validation error as a parameter selection procedure. However, the random nature of the cross-validation error is tend to be further deteriorated by the stochastic nature of MCMC, and it is necessary to use many validation sets to obtain reliable cross-validation error. Hence it is computationally extensive and the further systematic studies also remain as a future work.

4.1.4 Convergence of MCMC

Here we examine the convergence of the proposed algorithm. For each dataset among 50 datasets, we perform two MCMC experiments with different initial conditions using different random number sequence. Figure 6 shows plots of the pair of LOSS values corresponding to the two experiments at the 50, 100, 200, and 400 MCMC cycles, where each axis represents LOSS of each experiment. Correlation coefficients between LOSS values obtained by the two experiments are 0.815, 0.874, 0.976, and 0.981 respectively. It is observed that most of the pairs converge to the same LOSS values at 200 MCMC cycles.

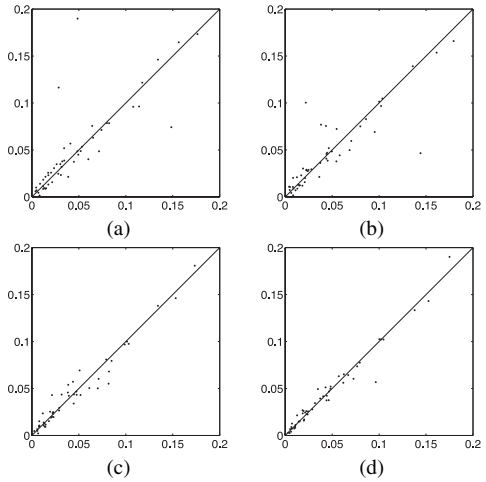


Fig. 6 Relation between two LOSS values of different initial conditions for 50 different datasets (20 burn-in and (a) 50 (b) 100 (c) 200 (d) 400 MCMC cycles).

4.2 Two-Dimensional Case

In this subsection, we show some preliminary results for two-dimensional case, mainly to confirm the validity of implementation of the algorithm.

Analogous to the one-dimensional case, $n = 50$ values $\{z_i^{true}\}$ are drawn independently from the standard Gaussian distribution $\mathcal{N}[0, I_2]$, where I_2 is the 2×2 identity matrix. The observed values $\{x_i\}$ of the input variable are generated by adding Gaussian noise $\mathcal{N}[0, \sigma_x^2 I_2]$, $\sigma_x^2 = (0.3)^2$ to $\{z_i^{true}\}$.

Here we try to estimate the hidden variable z_i^{true}; such inference potentially has many applications. In this experiment, we choose

$$f(z) = \|z\|^2 \tag{27}$$

as a target function, and y is generated by adding Gaussian noise $\mathcal{N}[0, (0.1)^2]$ in the same way as it is generated in one-dimensional case.

An example of the estimation of hidden variable is shown in Fig. 7. The upper panel shows the original configuration of input variable, where true values $\{z_i^{true}\}$ and samples $\{x_i\}$ are connected by segments. The lower panel shows the result of estimation, where the estimated values are closer to the true values than sample points in average.

To measure the performance of estimation, the average square distance of original configuration $\frac{1}{n}\sum_i \|x_i - z_i^{true}\|^2$

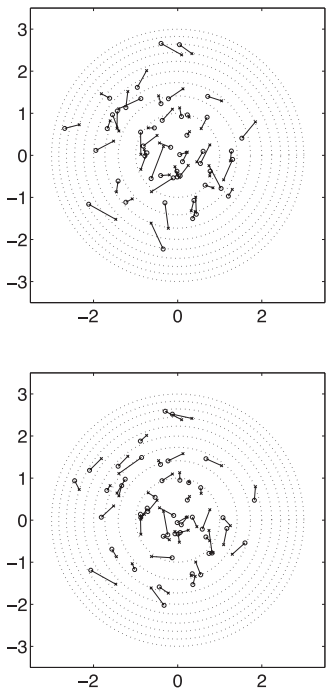


Fig. 7 Example of hidden variable estimation. ×: true value of z_i^{true}, ∘: given x_i (upper panel), or estimated value $z_i^{estimate}$ (lower panel). Dotted lines: contour lines of $f(x) = 1, 2, \ldots, 9$. The average square distance from the true value is reduced by 27% in this figure.

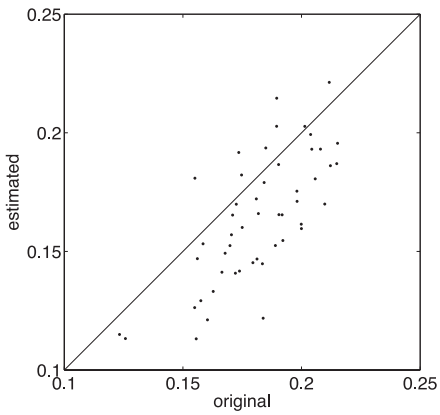


Fig. 8 The average square distance $\frac{1}{n}\sum_i \|x_i - z_i^{true}\|^2$ (original) vs $\frac{1}{n}\sum_i \|z_i^{estimated} - z_i^{true}\|^2$ (estimated).

and that of the estimated value $\frac{1}{n}\sum_i \|z_i^{estimated} - z_i^{true}\|^2$ are plotted in Fig. 8 for 50 different random configurations, and clear improvement of the accuracy is observed.

Note that the function takes the same value along the contour line in two-dimensional space, and there remains continuous freedom of z_i if we ignore the constraint by the prior $p(z_i)$ of z_i, even when the true function is known and $\sigma_y = 0$,

This effect is observed in Fig. 7 as well; each point approaches the contours, on which the corresponding true point is located.

Probably from this reason, and also by sparseness caused by high dimensionality, we didn't observe significant improvement in LOSS of estimates of $f(z)$ on observation points in this two-dimensional case. Further studies on the estimation of the function should be done as a future work.

As described above, in two-dimensional cases, when the function f is given, the probability density $p(x_i|f)$ is flat on a contour; it is even worse in higher dimension, where the contour is replaced by a (hyper)surface. In these cases, it will be interesting to consider measurements of several different quantities at each point x_i; it is formally expressed as a vector variable y_i. With this assumption, it is possible to identify the location of inputs as a cross point of contours or surfaces. It seems possible and interesting to deal such problems with our Bayesian framework.

5. Summary and Discussion

In this paper, we propose a method of Gaussian process regression for measurement error problems; it is regarded as an extension of usual kernel regression to incorporate the errors in input variables. We implement and test the proposed algorithm in cases of one- and two-dimensional input spaces. In the one-dimensional case, the proposed algorithm outperforms usual kernel regression for a set of synthetic data, even with cross-validated choice of hyperparameters in the usual method. The result is robust against the choice of hyperparameters in the proposed method.

Also, we report preliminary results on (1) cross-validated choice of hyperparameters in the proposed method; (2) a two-dimensional case. For (1), some promising results are obtained, but further intensive computation is required for the systematic study; implementation on parallel hardware will be necessary. For (2), partial success is obtained for the estimation of hidden input variables, while the estimate of the function f is not significantly improved; such observations are, however, based on limited examples and further experiments are necessary to understand high-dimensional cases.

An important issue in this study is convergence of MCMC. In Sec. 4, especially in one-dimensional cases, convergence seems to attain within moderate number of steps. It can be, however, very slow, in more difficult cases mentioned below. In these cases, implementation of advanced MCMC methods such as replica exchange Monte Carlo (REM) [20]–[22] will be useful; an example of the use of REM in a measurement error model is found in [1].

In this paper, we restrict ourselves to the cases that the Gaussian noise is added to the input variables. Other interesting examples arise from the analysis of "binned" or "coarse grained" data, where part of information on inputs is erased intentionally or by space-saving attempts. Such problems can be formulated by introducing a tessellation $\{S_\gamma\}$ of the input space, which satisfies $S_\gamma \cap S_{\gamma'} = \phi$ $(\gamma \neq \gamma')$ and $\cup_\gamma S_\gamma = \mathbb{R}^d$. The "bins" or "tiles" $\{S_\gamma\}$ represent coarse-graining of the input space. We assume the position z_i is described by telling which of $\{S_\gamma\}$ contains z_i. Denoting S_γ that contains z_i as $S_{\gamma(z_i)}$, this assumption is expressed as

$$p(x_i|z_i) = \begin{cases} \text{const.} & x_i \in S_{\gamma(z_i)}, \\ 0 & \text{else}. \end{cases} \tag{28}$$

Problems with these densities $p(x_i|z_i)$ can be treated by the proposed method, where any candidate z_i^* of $x_i \notin S_{\gamma(z_i^*)}$ is rejected in the Metropolis-Hastings steps.

In two-dimensional cases, when the function f is given, the probability density $p(x_i|f)$ is flat on a contour (see Sec.xxx); it is even worse in higher dimension, where the contour is replaced by a (hyper)surface. In these cases, it will be interesting to consider measurements of several different quantities at each point x_i; it is formally expressed as a vector variable y_i. With this assumption, it is possible to identify the location of inputs as a cross point of contours or surfaces. It seems possible and interesting to deal such problems with our Bayesian framework.

Finally, we note that there is an alternative approach to the problem. That is, we first integrate out f in $p(f, z|x, y)$; then we get a complicated formula of $p(z|x, y)$ that contains determinants, but it is still treated by a Metropolis-Hastings algorithm. Then we can obtain samples of $\{f(\tilde{x}_i)\}$ using $p(\{f(\tilde{x}_i)\}|x, y, z)$; we can also use Rao-Blackwellization in this step. Implementation of this alternative approach is also an issue left for future studies, as well as comparison with the method proposed in this paper.

Acknowledgements

We would be grateful to Prof. Arnaud Doucet, who suggested us an alternative approach discussed in Sec. 5. This study is supported by KAKENHI (Grant-in-Aid for Scientific Research(C)) 18500184 .

References

[1] K. Nakae, Y. Iba, Y. Tsubo, T. Fukai, and T. Aoyagi, "Bayesian estimation of phase response curves," Neural Netw., vol.23, pp.752–763, 2010.
[2] C.L. Cheng and J.W.V. Ness, Statistical Regression with Measurement Error (Kendall's Library of Statistics, 6), Wiley, 1999.
[3] W.A. Fuller, Measurement error models, Wiley-Blackwell, 2006.
[4] D. Ruppert, L.A. Stefanski, and C.M. Crainiceanu, Measurement Error in Nonlinear Models: A Modern Perspective (Monographs on Statistics and Applied Probability), Chapman and Hall, 2006.
[5] S. Amari and M. Kawanabe, "Information geometry of estimating functions in semi-parametric statistical models," Bernoulli, vol.3, no.1, pp.29–54, 1997.
[6] S.M. Berry, R.J. Carroll, and D. Ruppert, "Bayesian smoothing and regression splines for measurement error problems," J. American Statistical Association, vol.97, no.457, pp.160–169, 2002.
[7] P.J. Bickel, C.A.J. Klaassen, Y. Ritov, and J.A. Wellner, Efficient and Adaptive Estimation for Semiparametric Models, Springer-Verlag, 1993.
[8] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, eds., Markov chain Monte Carlo in practice, Chapman & Hall, 1995.
[9] J. Fan and Y. Truong, "Nonparametric regression with errors in variables," Annals of Statistics, vol.21, no.4, pp.1900–1925, 1993.
[10] A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin, eds., Bayesian data analysis, 2nd ed., Chapman & Hall, 2003.
[11] D.J.C. MacKay, Information theory, inference and learning algorithms, Cambridge University Press, 2003.
[12] C.E. Rasmussen and C.K.I. Williams, Gaussian processes for machine learning, The MIT Press, 2005.
[13] J. Neyman and E. Scott, "Consistent estimates based on partially consistent observations," Econometrica: J. Econometric Society, vol.16, no.1, pp.1–32, 1948.
[14] B. Schölkopf and A.J. Smola, Learning with kernels, MIT Press, 2002.
[15] C.P. Robert and G. Casella, Monte Carlo statistical methods, 2nd ed., Springer, 2004.
[16] D. MacKay, "Bayesian interpolation," Neural Comput., vol.4, no.3, pp.415–447, 1992.
[17] A.J. Smola, S.V.N. Vishwanathan, and T. Hofmann, "Kernel methods for missing variables," Proc. 10th Int. Workshop on AI& Statistics 2005, pp.325–334, 2005.
[18] J. Shawe-Taylor and N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, New York, NY, USA, 2004.
[19] G. Casella and C.P. Robert, "Rao-Blackwellisation of sampling schemes," Biometrika, vol.83, no.1, pp.81–94, 1996.
[20] C.J. Geyer, "Markov chain Monte Carlo maximum likelihood," Computing science and statistics: Proc. 23rd Symposium on the Interface Interface Foundation, ed. E. Keramidas, pp.156–163, Fairfax Station, 1991.
[21] K. Hukushima and K. Nemoto, "Exchange Monte Carlo method and application to spin glass simulations," J. Physical Society of Japan, vol.65, no.6, pp.1604–1608, 1996.
[22] Y. Iba, "Extended ensemble Monte Carlo," Int. J. Modern Physics C, vol.12, no.5, pp.623–656, 2001.

Appendix

In this appendix, we will discuss the results in Sec. 3.2 from a different viewpoint. Here we consider the case of changing many z_i's at one time, because there is no additional complexity for treating the generic case.

Remember that $\{\tilde{x}_i\}$ denotes observation points and $\{z_i^{(t)}\}$ are values of the input variables at t; hereafter we omit the superscript t and write $\{z_i\}$ in place of $\{z_i^{(t)}\}$. Also z_i^* indicates the candidate for the next value of z_i. Then, we define a set $\mathcal{S}$ as

$$\mathcal{S} = \{z_i\} \cup \{z_i^*\} \cup \{\tilde{x}_i\}. \tag{A·1}$$

By using this, $\mathcal{H}_\mathcal{S}$ is defined as the linear space spanned by the functions $\{k(\cdot, s_i)\},\ s_i \in \mathcal{S}$, i.e.,

$$f \in \mathcal{H}_\mathcal{S} \Leftrightarrow f = \textstyle\sum_i a_i k(\cdot, s_i),\ \exists\{a_i\}, a_i \in \mathbb{R}, \tag{A·2}$$

while $\mathcal{H}_\mathcal{S}^\perp$ denotes the orthogonal subspace of $\mathcal{H}_\mathcal{S}$ with respect to the inner product $(\ ,\ )_\mathcal{H}$.

Now, let us start from the decomposition

$$\mathcal{H} = \mathcal{H}_\mathcal{S} \oplus \mathcal{H}_\mathcal{S}^\perp. \tag{A·3}$$

The expression (A·3) indicates that any function $f \in \mathcal{H}$ is written as $f = f^\parallel + f^\perp$ using functions $f^\parallel \in \mathcal{H}_\mathcal{S}$ and $f^\perp \in \mathcal{H}_\mathcal{S}^\perp$. Then, the relations

$$f^\perp(s) = 0,\ \forall s \in \mathcal{S} \tag{A·4}$$

and

$$\|f\|_\mathcal{H}^2 = \|f^\parallel\|_\mathcal{H}^2 + \|f^\perp\|_\mathcal{H}^2, \tag{A·5}$$

hold. The decomposition (A· 3) is similar to the one used in the derivation of the representation theorem, which gives a connection with usual kernel regression.

By using (A· 4) and (A· 5), we can show that the conditional posterior density $p(f|z, x, y)$ derived from (8) satisfies

$$p(f|z, x, y) = p(f^{\|}|z, x, y)\, p(f^{\perp}|z, x, y), \tag{A· 6}$$

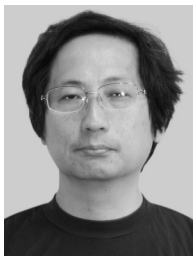
which enables independent sampling of $f^{\|}$ and $f^{\perp}$ from the posterior. Since the value of $f^{\perp}$ is zero in $\mathcal{S} = \{z_i\} \cup \{z_i^*\} \cup \{\tilde{x}_i\}$, we need not sample $f^{\perp}$. On the other hand, $f^{\|}$ is a sample from a finite-dimensional Gaussian distribution.

To specify this distribution of $f^{\|}$, we rewrite the posterior (13) with $a = \{a_i\}$ in (A· 2). Using the relation $\|f\|_{\mathcal{H}}^2 = \sum_{i,j} a_i a_j k(s_i, s_j)$, which is derived from (6), it is written as a Gaussian density

$$p(a|x, y, z) \propto \exp\left(-\frac{\sum_i (y_i - \sum_j a_j k(z_i, s_j))^2}{2\sigma_y^2} - \frac{1}{2} a^T K a\right), \tag{A· 7}$$

where K is the Gram matrix appeared in (14). The mean $\hat{a}$ of a maximizes the density (A· 7) and coincides with regression coefficients of usual kernel regression for the given value of z. On the other hand, the covariance matrix[†] is $\left(K + \frac{K^2}{\sigma_y^2} J\right)^{-1}$. Using (A· 2), i.e., $f^{\|} = Ka$, we can translate these results to the mean and covariance matrix of the distribution of $f^{\|}$; it reproduces (17) and (18).

[†]Here we assume the Gram matrix K has full rank.



Yukito Iba is an Associate Professor of the Institute of Statistical Mathematics (ISM). He was Born in Tokyo in 1959 and majored in statistical physics at the University of Tokyo. Since 1988, he is a staff member of ISM. He obtained Ph. D degree in physics from Osaka University in 2000. He is interested in statistical science, statistical physics, Monte Carlo algorithms, and cross-fertilization of different fields of sciences.

Shotaro Akaho received a Bachelor, Master and Ph.D. degrees in Mathematical Engineering at the University of Tokyo in 1988, 1990 and 2001 respectively. He joined Electrotechnical Laboratory (ETL) in 1990. Since 2001, he has been the Group Leader of Mathematical Neuroinformatics Group of Human Technology (previously Neuroscience) Research Institute, The National Institute of Advanced Industrial Science and Technology (AIST). He has also been an invited professor at Osaka university in 2003-2007. His research interests are theories and algorithms of statistical machine learning.