Learning Gradients with Gaussian Processes[*]

Xinwei Jiang[1], Junbin Gao[2,**], Tianjiang Wang[1], and Paul W. Kwan[3]

[1] Intelligent and Distributed Computing Lab,
School of Computer Science and Technology,
Huazhong University of Science and Technology, Wuhan, 430074, China
ysjxw@hotmail.com

[2] School of Computing and Mathematics,
Charles Sturt University, Bathurst, NSW 2795, Australia
jbgao@csu.edu.au

[3] School of Science and Technology,
University of New England, Armidale, NSW 2351, Australia
kwan@turing.une.edu.au

Abstract. The problems of variable selection and inference of statistical dependence have been addressed by modeling in the gradients learning framework based on the representer theorem. In this paper, we propose a new gradients learning algorithm in the Bayesian framework, called Gaussian Processes Gradient Learning (GPGL) model, which can achieve higher accuracy while returning the credible intervals of the estimated gradients that existing methods cannot provide. The simulation examples are used to verify the proposed algorithm, and its advantages can be seen from the experimental results.

1 Introduction

Analyzing data sets associated with many variables or coordinates has become increasingly challenging in many circumstances, especially in biological and physical sciences [1]. A wide range of machine learning algorithms based on the regularization theory such as support vector machines (SVMs) [2] have been proposed to solve the predictive problems in the past two decades. Although these approaches demonstrate quite acceptable and robust performances in a lot of experiments and applications, sometimes one also wants to get an insight into the relationships between the coordinates and the influence of the coordinates/attributes/features on the outputs. For example, it is very interesting to investigate which covariant is most significant for prediction and how the variables vary with respect to each other in estimation.

The gradient of the target function provides a valuable measure to characterize the relationships [3,4,5,1] and it has been used in many approaches and

[*] This research was supported by the National High Technology Research and Development (863) Program of China (2007AA01Z161).

[**] The author to whom all the correspondence should be addressed.

applications. For example, the minimum average variance estimation (MAVE) method and the outer product of gradients (OPG) estimation approach proposed by [3] focus on finding the effective dimension reduction (e.d.r.) space in the data sets by using the gradients of the training data implicitly and explicitly, respectively. These models show better performance in the estimation of e.d.r. space than others, but learning gradient information would fail in the "large m (dimension), small n (size of the dataset)" paradigm [6]. Recently, [4] and [5] proposed a method to learn the gradient of a target function directly from a given data set based on the Tikhonov regularization method, which avoided the overfitting problem in the "large m, small n" settings. The most significant statistical measure we can get by those nonparametric kernel based models is the gradient outer product (GOP) matrix, which can interpret the importance of the coordinates for the prediction and the covariation with respect to each other. In addition, with the assistance of spectral decomposition of the gradient outer product matrix, the e.d.r. directions can be directly estimated [1]. Furthermore [7] extended gradient learning algorithm from the Euclidean space to the manifolds setting, and provided the convergence rate dependent on the intrinsic dimension of the manifold rather than the dimension of the ambient space. This is very important in the "large m, small n" settings. Except for the application examples proposed in the available literature, gradient learning from scattered data sets is particularly important for surfaces reconstruction in computer graphics where, when visually scaled geometric surfaces constructed from scattered data, analytical expression (or rules) of gradients was highly desirable in calculating the normals at any given point needed in most surface reconstruction algorithms (see [8]).

However, these direct gradient learning methods cannot offer any reasonable error bars for the estimated gradients because essentially the task of estimating gradients is the problem of point estimation. In many application scenarios, a confidence interval on the estimates is very important, such as found in computer graphics.

In this paper, we propose a new gradient learning approach under the Bayesian framework based on Gaussian Processes (GPs) [9]. Compared to the learning gradients method in [4], not only can our algorithm apply in the "large m, small n" cases and achieve higher accuracy, but it can also return the error bars of the estimated gradients, which provide us with an estimate of the uncertainty of stability. We will verify these features in Sections 5.

The rest of this paper is organized as follows. In Section 2, we introduce the statistical foundation for learning gradients. The gradients learning method with Gaussian Processes will be proposed in Section 3, which includes a brief introduction of the Gaussian Processes regression. The algorithm derivation is illustrated in Section 4. Then, simulated data are used to verify our algorithm in Section 5. Finally, closing remarks and comments will be given in Section 6.

2 The Statistical Foundation for Learning Gradients

2.1 Notations

Denote data $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$ where $\boldsymbol{x}_i$ is a vector in a m-dimensional compact metric subspace $\mathcal{X} \subset \mathbb{R}^m$ and $\boldsymbol{y}_i \in \mathbb{R}^p$ is a vector too. Without loss of generality, we will assume that $p = 1$. Our approach can be easily extended to the case of vectorial value outputs $\boldsymbol{y}$. Typically we assume that the data are drawn i.i.d. from a joint distribution, $(\boldsymbol{x}_i, \boldsymbol{y}_i) \sim p(X, Y)$. In the standard regression problem, we want to model the regression function F defined by the conditional mean of $Y|X$, i.e., $F = \mathbb{E}_Y[Y|X]$. The gradient of F is a vectorial value function with m components, if all the partial derivatives exist,

$$\boldsymbol{f}(\boldsymbol{x}) \triangleq \nabla F = (f_1(\boldsymbol{x}), ..., f_m(\boldsymbol{x}))^T = \left(\frac{\partial F(\boldsymbol{x})}{\partial x^1}, \cdots, \frac{\partial F(\boldsymbol{x})}{\partial x^m}\right)^T \tag{1}$$

where x^i are the components of the vector $\boldsymbol{x}$ and $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), ..., f_m(\boldsymbol{x}))$.

The gradient and the issues of variable selection and coordinate covariation are relevant because the gradient can provide following information [4]:

1. Variable selection: the norm of the partial derivative $\left\|\frac{\partial F}{\partial x^i}\right\|$ indicates the significance of the variables for the prediction because a small norm implies a slight change in the function F with respect to the i-th coordinate.
2. Coordinate covariation: the inner product of the partial derivatives with respect to different dimensions $\langle\frac{\partial F}{\partial x^i}, \frac{\partial F}{\partial x^j}\rangle$ indicates the covariance of the i-th and j-th coordinates.

A central concept in all gradients learning approaches, called the gradient outer product (GOP) matrix, is defined by

$$\Gamma_{ij} = \mathbb{E}\langle\frac{\partial F}{\partial x^i}, \frac{\partial F}{\partial x^j}\rangle \tag{2}$$

The GOP has a deep relation with the so-called effective dimension reduction (e.d.r.) space and the relationship was exploited in several gradient regression methods such as MAVE and OPG [3] and [4,5].

2.2 Learning Gradients

To propose our approach for gradients learning, let us focus on the introduction of those available algorithms of learning the gradients from the data. Recall that the MAVE and OPG suffer from the problem of the overfitting in the "large m, small n" paradigm, so the so-called regularized framework has been used to overcome the overfitting based on the kernel representer theorem. Actually the kernel representer theorem is also the motivation for our algorithm in the next section. The algorithms based on the kernel representer theorem show better performance than the MAVE and OPG [1].

Our goal is to design a model for the gradient estimate directly from data. The conventional methods usually take a two-steps procedure by first learning a regression function F and then calculating the gradients of the F. However a direct gradients learning algorithm may have more advantages than the conventional ways, as demonstrated in [4,1].

Essentially, all of those kinds of models are motivated by the Taylor expansion of the target function:

$$y_i \approx y_j + \boldsymbol{f}(\boldsymbol{x}_i)^T(\boldsymbol{x}_i - \boldsymbol{x}_j) \quad \text{for} \quad \boldsymbol{x}_i \approx \boldsymbol{x}_j \tag{3}$$

A model for gradients leaning from an observation dataset $\mathcal{D}$ is defined as

$$\boldsymbol{f} := \underset{\boldsymbol{f}}{\operatorname{argmin}} \left\{ \frac{1}{n^2} \sum_{i,j=1}^{n} w_{ij} [y_i - y_j + \boldsymbol{f}(\boldsymbol{x}_i)^T(\boldsymbol{x}_j - \boldsymbol{x}_i)]^2 + \lambda \|\boldsymbol{f}\|^2 \right\} \tag{4}$$

where w_{ij} is called weights and defined as $w_{ij} = \frac{1}{\sigma^{m+2}} \exp\left\{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right\}$ where σ^2 is set to the median of the input data. When $\boldsymbol{x}_j$ is far away from $\boldsymbol{x}_i$, the Taylor expansion of the function $F(\boldsymbol{x}_j)$ at $\boldsymbol{x}_i$ makes less contribution to the regression objective.

According to the representer theorem [10], the optimal solution to (4) is the linear combination of kernel function defined on the data points, thus the problem is actually transformed to solving a linear systems problem, see [4].

Due to regularization, this model can prevent overfitting in the "large m, small n" paradigm and obtain fairly remarkable performance. However, sometimes it is also important that we want to know the error bars of the point estimation for the gradient, which can not be provided by those kinds of models.

An alternative method is to define the model under the Bayesian learning and inference framework. We aim to use the Gaussian Processes (GPs) model which is also based on the kernel and can be viewed as the exhibition of the representer theorem. So motivated by the model in [4] and associated it with the GPs, we will show how to improve the accuracy and compute the error bars of the estimated gradient in the following section.

3 Gradients Learning with Gaussian Processes

3.1 Gaussian Processes Regression

Given the data set which consists of the i.i.d. samples from unknown distribution $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \cdots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\} \subset \mathbb{R}^m \times \mathbb{R}^p$ The standard Gaussian Process regression is concerned with the case when $p = 1$. The goal is to estimate the $p(\boldsymbol{y}|\boldsymbol{x}^*)$ for a test data $\boldsymbol{x}^*$. In the standard Gaussian processes (GPs) regression model, a latent variable f is introduced in the model defined by

$$y = f(x) + \epsilon$$

where ϵ is the additive noise, specified by the likelihood $p(y|f,x) = p(y|f)$. This model is nonparametric because the latent variable f is random function which follows the Gaussian Process with zero mean and covaiance function $k(\cdot,\cdot)$. Also the likelihood follows a Gaussian distribution with zero mean and covariance σ_t^2.

Denote by $X = \{\boldsymbol{x}_i\}_{i=1}^n$. Due to the independence of the samples $\mathcal{D}$, its likelihood under the model is the product of $p(y_i|f(\boldsymbol{x}_i))$ which is a Gaussian too. Given a test point $\boldsymbol{x}^*$, it is easy to check that the joint distribution of the latent function is given by, see [9],

$$\begin{bmatrix} f \\ f^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K_{XX} & K_{X\boldsymbol{x}^*} \\ K_{\boldsymbol{x}^*X} & K_{\boldsymbol{x}^*\boldsymbol{x}^*} \end{bmatrix}\right) \tag{5}$$

where K are matrix of the kernel function values at the corresponding points and $\mathcal{N}(\mu, \Sigma)$ denotes the Gaussian distribution with mean μ and covariance Σ.

Under the Gaussian likelihood assumption, we can simply add the covariance of the noise to the GP prior due to the independence assumption of the noise. So the predictive distribution on the observation is

$$f^*|\boldsymbol{x}^*, X, \boldsymbol{y} \sim \mathcal{N}(K_{\boldsymbol{x}^*X}(K_{XX} + \sigma_t^2\mathbf{I})^{-1}\boldsymbol{y},\ K_{\boldsymbol{x}^*\boldsymbol{x}^*} - K_{\boldsymbol{x}^*X}(K_{XX} + \sigma_t^2\mathbf{I})^{-1}K_{X\boldsymbol{x}^*}) \tag{6}$$

where the variance of the conditional distribution illustrates the uncertainty of the prediction and the mean can be written as $f(\boldsymbol{x}^*) = \sum_{i=1}^n \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}^*)$, where $\alpha = (K_{XX} + \sigma_t^2\mathbf{I})^{-1}\boldsymbol{y}$. This form of the prediction exhibits the fact that the GP can be represented in terms of a number of basis function is one feature of the representer theorem.

3.2 Gradients Estimation Model with Gaussian Processes

To apply the Gaussian Process model in the case of gradients learning, we have to overcome two hurdles. First, the regression model (4) shows that we are dealing with a multi-task regression problem as the gradient $\boldsymbol{f}$ is a vectorial function, so we have to generalize the standard Gaussian Process regression to multi-task case. This has been done in the recent works such as [11]. Second, the i.i.d. assumption for the data set can not be used to produce a joint likelihood which is the product of individual likelihood at each data point. In fact, when we transform (4) into probabilistic formulation, we see that the coupling between data makes learning and inference more complicated. However, we can still define a likelihood for the whole data set $\mathcal{D}$ rather than a likelihood for each data pair.

Under the above modification, we can formulate the gradients learning model in the Bayesian framework based on the GPs, named Gaussian Process Gradient Learning (GPGL) model, and we will show the advantages in Section 5.

Based on the analysis we have just given, a new likelihood formulation by extending the datum-based likelihood to dataset-based likelihood is defined as

$$p(Y|X,\mathbf{f}) \propto \exp\left\{-\frac{1}{2}\sum_{i,j=1}^n w_{ij}[y_i - y_j + \boldsymbol{f}(\boldsymbol{x}_i)^T(\boldsymbol{x}_j - \boldsymbol{x}_i)]^2\right\} \tag{7}$$

Let us introduce the following notation, the scalar $c = \sum_{i,j=1}^{n} w_{ij}(y_i - y_j)^2$, the $m \times m$ matrices $B_i = \sum_{j=1}^{n} w_{ij}(\boldsymbol{x}_i - \boldsymbol{x}_j)(\boldsymbol{x}_i - \boldsymbol{x}_j)^T$, and the m dimensional vectors $h_i = \sum_{j=1}^{n} w_{ij}(y_i - y_j)(\boldsymbol{x}_i - \boldsymbol{x}_j), i = 1, 2, ..., n$. We will use the same weights w_{ij} as [4] for comparison. Furthermore define $B = U^T \mathrm{diag}(B_1, B_2, \cdots, B_n)U$ and a column vector of dimension mn $\boldsymbol{h} = U^T[h_1^T, h_2^T, \cdots, h_n^T]^T$, where U is a permutation matrix. Similarly define the column vector (of dimension mn) $\mathbf{f} = [\boldsymbol{f}_1^T, \boldsymbol{f}_2^T, ..., \boldsymbol{f}_m^T]^T$ where $\boldsymbol{f}_i = [f_i(x_1), f_i(x_2), ..., f_i(x_n)]^T$.

Under the above notation, it is easy to validate that the likelihood (7) of the observation dataset $\mathcal{D}$ can be written as

$$p(Y|X, \mathbf{f}) = \frac{1}{M} \exp\left\{-\frac{1}{2}(\mathbf{f}^T B \mathbf{f} - 2\boldsymbol{h}^T \mathbf{f} + c)\right\} \tag{8}$$

where M is the normalized constant.

The variable $\mathbf{f}$ collects the information of m partial derivatives over the given input data X. In our model formulation, the variable $\mathbf{f}$ is assumed to be a Gaussian Processes while the covariance function is $\Sigma = K_{ff} \otimes K_{XX}$. So the Gaussian processes prior is

$$p(\mathbf{f}|X, \theta) = \frac{1}{|2\pi\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}\mathbf{f}^T \Sigma^{-1} \mathbf{f}\right\} \tag{9}$$

where $K_{ff} \in \mathbb{R}^{m \times m}$ is the coordinate-similarity matrix, $K_{XX} \in \mathbb{R}^{n \times n}$ is the covariance matrix of the samples X, and θ is the parameters of the covariance function.

By using the Bayesian formulation, the posterior of $\mathbf{f}$ given the dataset is

$$p(\mathbf{f}|X, Y, \theta) = \frac{p(Y|X, \mathbf{f})p(\mathbf{f}|X, \theta)}{p(Y|X, \theta)} \tag{10}$$

As all the densities in the above relation are Gaussian, it is easy to derive, see Appendix A of [9], the posterior of $\mathbf{f}$

$$p(\mathbf{f}|X, Y, \theta) = \frac{1}{|2\pi E|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{f} - E\boldsymbol{h})^T E^{-1}(\mathbf{f} - E\boldsymbol{h})\right\} \tag{11}$$

where $E = (B + \Sigma^{-1})^{-1}$.

For a new data $\boldsymbol{x}_*$, we want to estimate $\boldsymbol{f}_* = \boldsymbol{f}(\boldsymbol{x}_*)$ based on the observation data. According to the predictive distribution of Gaussian processes, we have

$$\boldsymbol{f}_*|\mathbf{f}, \boldsymbol{x}_*, X, \theta \sim \mathcal{N}((K^f \otimes K_{*X})^T \Sigma^{-1}\mathbf{f}, K^f \otimes K_{**} - (K^f \otimes K_{*X})^T \Sigma^{-1}(K^f \otimes K_{*X})) \tag{12}$$

where $K_{*X} = K(X, \boldsymbol{x}_*)$, $K_{**} = K(\boldsymbol{x}_*, \boldsymbol{x}_*)$. By integrating over the uncertainty $\boldsymbol{f}$ according to the posterior (11), we can get the gradients predictive distribution

$$\begin{aligned} p(\boldsymbol{f}_*|\boldsymbol{x}_*, X, Y) &= \int p(\boldsymbol{f}_*|\mathbf{f}, \boldsymbol{x}_*, X) p(\mathbf{f}|X, Y) d\mathbf{f} \\ &= \frac{1}{|2\pi Q|^{1/2}} \exp\left\{ -\frac{1}{2}(\boldsymbol{f}_* - P)^T Q^{-1} (\boldsymbol{f}_* - P) \right\} \end{aligned} \tag{13}$$

That is, the gradients predictive distribution is a Gaussian with the mean P and the covariance Q. Thus the gradient estimate is given by

$$P = (K_{ff} \otimes K_{*X})^T (B\Sigma + I)^{-1} \boldsymbol{h} \tag{14}$$

and the error bar is given by

$$Q = K_{ff} \otimes K_{**} - (K_{ff} \otimes K_{*X})^T (\Sigma + B^{-1})^{-1} (K_{ff} \otimes K_{*X}). \tag{15}$$

4 Learning Kernel Hyperparameters

To develop an approach for learning coordinate-similarity matrix K_{ff} and the kernel hyperparameters, we use gradient-based optimization of the marginal likelihood $p(Y|X, \theta)$. Without loss of generality, we just consider K_{ff} as unit matrix. Since K_{ff} controls the correlations between m dimensions of the gradients, the simplicity means that we are assuming the independence of different coordinates. Actually the optimization with respect to the parameters in K_{ff} can be dealt with in the same way as follows [11].

Then the log marginal likelihood $\log p(Y|X, \theta)$ is given by

$$L = -\frac{1}{2}\log|B^{-1} + \Sigma| - \frac{1}{2}\boldsymbol{h}^T (B^{-1} + \Sigma)^{-1} \boldsymbol{h} + C. \tag{16}$$

where C is a constant independent of the parameter θ, which can be ignored in optimizing L with respect to θ. To work out a formula for the derivatives of L with respect to θ, we refer to the matrix reference manual for the notation [12].

Denote by $F_1 = -\log|B^{-1} + \Sigma|$, then $dF_1 = -(B^{-1} + \Sigma)^{-1} :^T d(\Sigma) :$. Similarly, we have $dF_2 = \Big((B^{-1} + \Sigma)^{-1} \boldsymbol{h}\boldsymbol{h}^T (B +^{-1} + \Sigma)^{-1}\Big) :^T d(\Sigma) :$, where $F_2 = -\boldsymbol{h}^T (B^{-1} + \Sigma)^{-1} \boldsymbol{h}$,

According to the derivative formula for the Kronecker product of matrices, we have $d(\Sigma) = d(K_{ff} \otimes K_{XX}) = (I_{m,m} \otimes T_{n,m} \otimes I_{n,n})(K_{ff} : \otimes I_{n^2,n^2}) dK_{XX} :$, where $T_{m,n}$, called the vectorized transpose matrix, is the $mn \times mn$ permutation matrix whose (i,j)-th elements is 1 if $j = 1 + m(i-1) - (mn-1)\lfloor \frac{i-1}{n} \rfloor$ or 0 otherwise.

So the derivatives of L with respect to θ is

$$\begin{aligned} \frac{\partial L}{\partial \theta} =& \frac{1}{2} \left[-((B^{-1} + \Sigma)^{-1} :)^T + ((B^{-1} + \Sigma)^{-1} \boldsymbol{h}\boldsymbol{h}^T (B^{-1} + \Sigma)^{-1} :)^T \right] \\ & (I_{m,m} \otimes T_{n,m} \otimes I_{n,n})(K_{ff} : \otimes I_{n^2,n^2}) \frac{dK_{XX}}{d\theta}. \end{aligned} \tag{17}$$

In our experiments, we learn the parameters of the models so as to maximize the marginal likelihood using gradient-based search. The code is based on Neil D. Lawrence's MATLAB packages `Kern` and `Optimi`[1].

We have seen that $(B^{-1} + \Sigma)^{-1}$ needs to be inverted for both making predictions and learning the hyperparameters in time $\mathcal{O}(m^3 n^3)$. This can lead to computational problems if mn is large. Although we only use cholesky decomposition and singular value decomposition to accelerate computation, the efficient approximation method in [11] can be directly used in our GPGL algorithm to reduce the computational complexity.

5 Experiments

In this section we will verify our GPGL algorithm in two simulated data sets to show the higher accuracy of the estimation and the credible intervals that the gradient learning methods in [4], named Mukherjee's algorithm in the following, can not gain. In the first data set, we generate some samples from four simple functions which can compute the real gradients for comparison. Another high-dimensional data set is used to test that our algorithm can be applied to show the variable selection and coordinate covariance like Mukherjee's algorithm.

5.1 Error Bar Estimation

We illustrate how GPGL can be used to estimate the credible intervals of the estimated gradient and compare Mukherjee's algorithm with GPGL to show higher accuracy that GPGL demonstrates.

Given four representative elementary regression models $y = \exp(x); y = \ln(x); y = x^2; y = \sin(x)$, where $\{(x_i, y_i)\}_{i=1}^n \in \mathbb{R} \times \mathbb{R}$ and $x_i \sim N(1, 0.1)$. In our experiment, we sampled 100 points from the Gaussian distribution. The true derivatives are given by $y' = \exp(x); y' = 1/x; y' = 2 * x; y' = \cos(x)$, respectively. The comparison of the results between proposed GPGL algorithm and Mukherjee's algorithm is shown in Figures 1 to 4. We use the mean squared error between the true derivative and learned derivative to measure the quality of learning algorithm. The smaller MSE means that a better performance of the algorithm. All the MSEs for those four functions with different algorithms are collected in Table 1. It can be seen that the proposed GPGL algorithm gives better performance in terms of lower MSEs for three out of the four functions.

Although for the functions $y = \exp(x)$ and $y = x^2$, Mukherjee's algorithm gives slightly better results, the proposed GPGL algorithm outperforms Mukherjee's Algorithm in other cases. However, in the experiment, we find that Mukherjee's algorithm is sensitive to the value of regularization parameter and the percentage of the eigenvalues parameters (see the code in [4]) that need to be chosen manually, especially the regularization parameter. Sometimes, it is hard to choose them optimally, although a standard cross validation can be

[1] `http://www.cs.manchester.ac.uk/~neil/software.html`

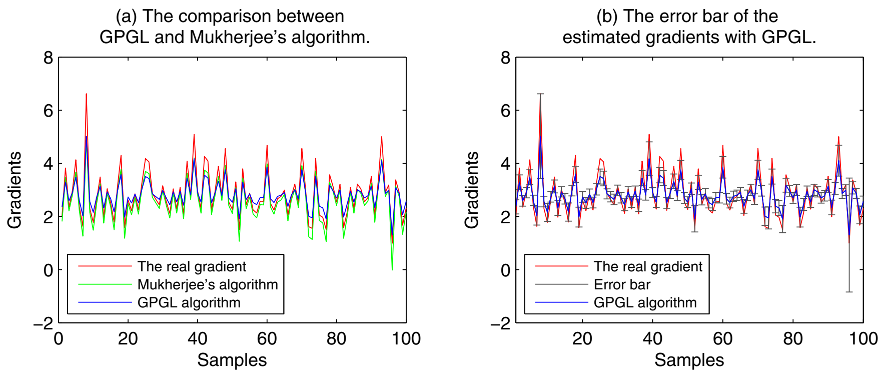

Fig. 1. The Result for function $y = \exp(x)$

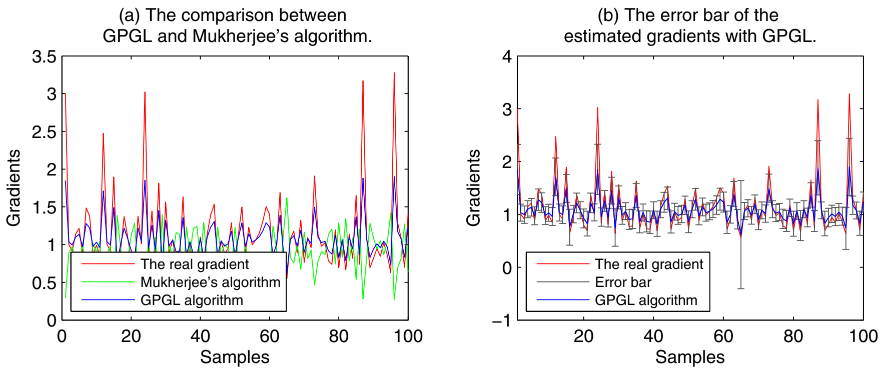

Fig. 2. The Result for function $y = \ln(x)$

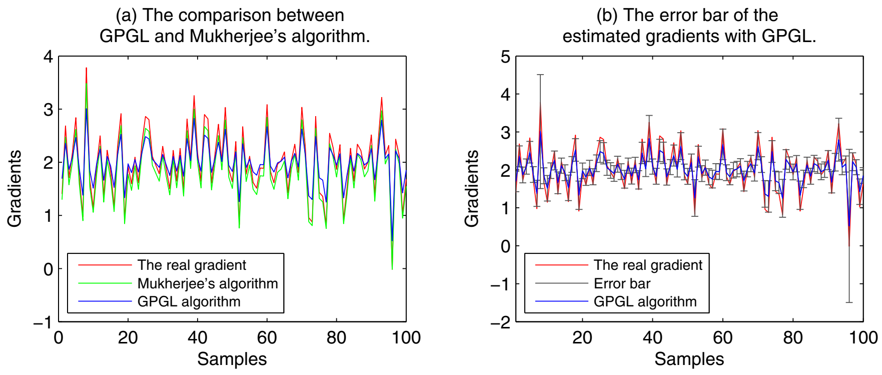

Fig. 3. The Result for function $y = x^2$

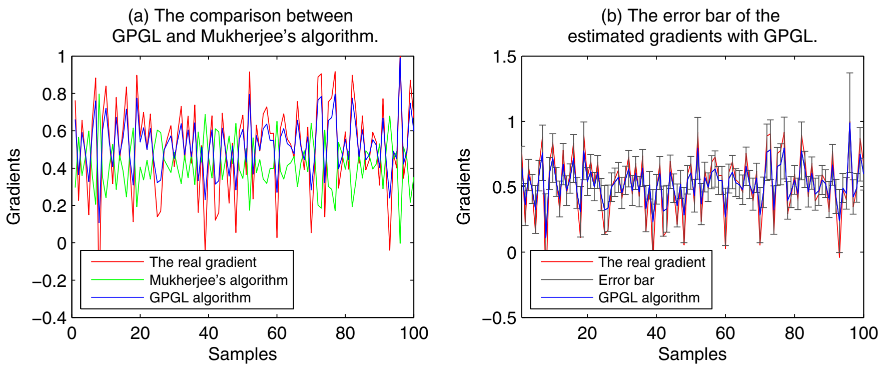

Fig. 4. The Result for function $y = \sin(x)$

Table 1. The Mean Squared Error

Algorithm	$y = \exp(x)$	$y = \ln(x)$	$y = x^2$	$y = \sin(x)$
GPGL	19.6275	12.8840	7.9557	1.3999
Mukherjee's	12.0330	80.8199	2.7621	15.9319

applied. However, the proposed GPGL method does not suffer from this problem and is more stable with ability to automatically adapt parameters. In addition, the error bars can be obtained from our algorithm along with gradient estimation.

5.2 High-Dimensional Data Set

Definition 1. The empirical gradient matrix (EGM), F_z, is the $m \times n$ matrix whose columns are $\boldsymbol{f}(\boldsymbol{x}_j)$ with $j = 1, \cdots, n$. The empirical covariance matrix (ECM), is the $m \times m$ matrix of inner products of the directional derivative of two coordinates, which can be denoted as $\text{Cov}(\boldsymbol{f}) := [\langle(\boldsymbol{f})_p, (\boldsymbol{f})_q\rangle_K]_{p,q=1}^m$.

The ECM gives us the covariance between the coordinates while the EGM provides us information about how the variables differ over different sections of the space.

For a fair comparison, we construct the same artificial data as those used in [4]. By creating a function in an $m = 80$ dimensional space which consists of three linear functions over different partitions of the space, we generate $n = 30$ samples as follows:

1. For samples $\{\boldsymbol{x}_i\}_{i=1}^{10}$,

$$\boldsymbol{x}^j \sim \mathcal{N}(1, \sigma_x), \text{ for } j = 1, \cdots, 10;$$
$$\boldsymbol{x}^j \sim \mathcal{N}(0, \sigma_x), \text{ for } j = 11, \cdots, 80;$$

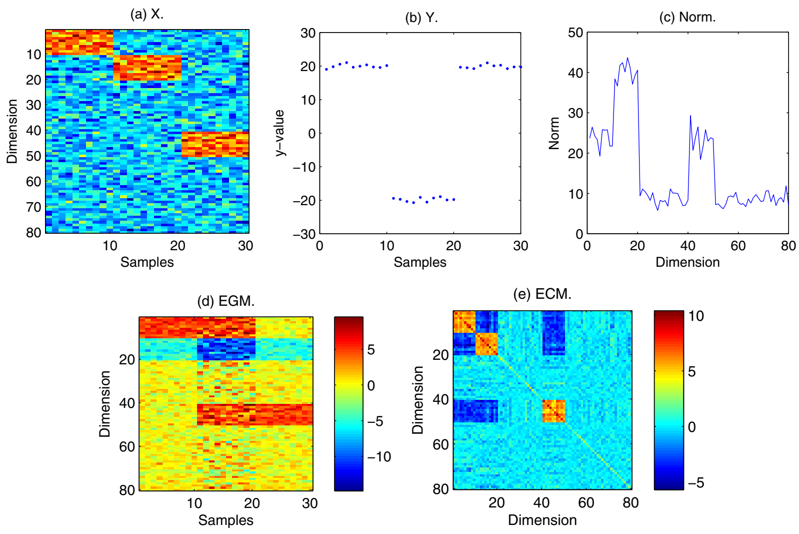

Fig. 5. a). The data matrix x; b). The vector of y values; c). The RKHS norm for each dimension; d). An estimate of the gradient at each sample; e). The empirical covariance matrix

2. For samples $\{\boldsymbol{x}_i\}_{i=11}^{20}$,

$$\boldsymbol{x}^j \sim \mathcal{N}(1, \sigma_x), \ \text{for } j = 11, \cdots, 20;$$
$$\boldsymbol{x}^j \sim \mathcal{N}(0, \sigma_x), \ \text{for } j = 1, \cdots, 10, 21, \cdots, 80;$$

3. For samples $\{\boldsymbol{x}_i\}_{i=21}^{30}$

$$\boldsymbol{x}^j \sim \mathcal{N}(1, \sigma_x), \ \text{for } j = 41, \cdots, 50;$$
$$\boldsymbol{x}^j \sim \mathcal{N}(0, \sigma_x), \ \text{for } j = 1, \cdots, 40, 51, \cdots, 80;$$

A representation of this X matrix is shown in Figure 5(a). Three vectors with support over different dimensions were constructed as follows:

$$w_1 = 2 + 0.5\sin(2\pi i/10) \ \text{ for } i = 1, \cdots, 10 \text{ and } 0 \ \text{ otherwise},$$
$$w_2 = -2 + 0.5\sin(2\pi i/10) \ \text{ for } i = 11, \cdots, 20 \text{ and } 0 \ \text{ otherwise},$$
$$w_3 = 2 - 0.5\sin(2\pi i/10) \ \text{ for } i = 41, \cdots, 50 \text{ and } 0 \ \text{ otherwise},$$

Then the function is defined by

1. For samples $\{y_i\}_{i=1}^{10}$ $\quad y_i = \boldsymbol{x}_i \cdot w_1 + \mathcal{N}(0, \sigma_y)$,
2. For samples $\{y_i\}_{i=11}^{20}$ $\quad y_i = \boldsymbol{x}_i \cdot w_2 + \mathcal{N}(0, \sigma_y)$,
3. For samples $\{y_i\}_{i=21}^{30}$ $\quad y_i = \boldsymbol{x}_i \cdot w_3 + \mathcal{N}(0, \sigma_y)$.

A draw of the y values is shown in Figure 5(b). In Figure 5(c), we plot the norm of each component of the estimate of the gradient using the GPGL algorithm.

The norm of each component gives an indication of the importance of a variable and variables with small norms can be eliminated. Note that the coordinates with nonzero norm are the ones we expect, $l = 1, \cdots, 20, 41, \cdots, 50$. In Figure 5(d) we plot the EGM, while the ECM is displayed in Figure 5(e). The blocking structure of the ECM indicates the coordinates that covary. The similar result can be found in [4].

6 Conclusions

In this paper we have proposed a direct gradient learning algorithm from sample dataset in the Bayesian framework. The Gaussian Processes Gradient Learning (GPGL) model we propose can be seen as the manifestation of the representer theorem which is the basis of Mukherjee's algorithm. However, only the GPGL model can provide the error bars of the estimated gradients which characterize the uncertainty of the estimation. Besides, the GPGL model is stable and shows higher accuracy than Mukherjee's algorithm in terms of MSE in some circumstances. Another advantage is that GPGL model is more stable with automatical parameter adapting while the result from Mukherjee's algorithm heavily depends on the better tuning of the regularization parameters. In future work we plan to extend GPGL to sparse model to improve the generalization capability that is especially useful in the "large m, small n" setting.

References

1. Wu, Q., Guinney, J., Maggioni, M., Mukherjee, S.: Learning gradients: predictive models that infer geometry and dependence. Technical report, Duke University (2007)
2. Vapnik, V.: Statistical Learning Theory. Wiley, Chichester (1998)
3. Xia, Y., Tong, H., Li, W.K., Zhu, L.X.: An adaptive estimation of dimension reduction space. Journal of Royal Statistical Society 64(3), 363–410 (2002)
4. Mukherjee, S., Zhou, D.X.: Learning coordinate covariances via gradients. Journal of Machine Learning Research 7, 519–549 (2006)
5. Mukherjee, S., Wu, Q.: Estimation of gradients and coordinate covariation in classification. Journal of Machine Learning Research 7, 2481–2514 (2006)
6. West, M.: Bayesian factor regression models in the "large p, small n" paradigm. In: Bayesian Statistics, vol. 7, pp. 723–732. Oxford University Press, Oxford (2003)
7. Mukherjee, S., Wu, Q., Zhou, D.X.: Learning gradients and feature selection on manifolds. Technical report, Duke University (2007)
8. Dollar, P., Rabaud, V., Belongie, S.: Non-isometric manifold learning: Analysis and an algorithm. In: International Conference on Machine Learning, pp. 241–248 (2007)
9. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press, Cambridge (2006)
10. Schoelkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. The MIT Press, Cambridge (2001)
11. Bonilla, E.V., Chai, K.M.A., Williams, C.K.I.: Multi-task gaussian process prediction. In: Advances in Neural Information Processing Systems, vol. 20, pp. 153–160 (2008)
12. Brookes, M.: The Matrix Reference Manual (2005)