ATLANTIS PRESS

Advances in Computer Science Research, volume 80

International Conference on Computer Science, Electronics and Communication Engineering (CSECE 2018)

Two-step Gaussian Process Regression Improving Performance of Training and Prediction

Wei Wang*, Santong Zhang, Wei Yang and Xiangbin Liu

School of Electronic and Information Engineering, Beijing Jiaotong University, China

*Corresponding author

***Abstract*—Since Gaussian process regression (GPR) cannot feasibly be applied to big and growing data sets, this paper introduces an integration algorithm called Two-step Gaussian Process Regression (TGPR) which speeds up both training and prediction to solve the problem. First, analyze the basics behind regular GPR. Then, introduce TGPR by using the inducing inputs to optimize the regular GPR algorithm. Last, apply TGPR to a three-dimension model, the experimental results compared with regular GPR show that TGPR is faster and more accurate.**

Keywords—gaussian process; regression; inducing inputs; hyperparameter

I INTRODUCTION

Gaussian Process Regression is a newly developed method for machine learning based on Bayesian theory and statistical learning theory. It is first applied to machine learning by Rasmussen and Williams [1]. The prediction result of GPR has a clear probabilistic meaning compared with Artificial Neural Networks (ANN) [2] and Support Vector Machine (SVM) [3].

Although GPR has many advantages, its computational complexity increases sharply as the size of the measurements expands. Many sparse algorithms have been proposed to reduce computation. Williams presents the Nyström method for efficient GPR [4]. Snelson presents Sparse Pseudo-inputs Gaussian Process (SPGP) model whose covariance is parameterized by the locations of pseudo-input [5]. Tresp presents BCM (Bayesian Committee Machine) method to combine estimators that are trained on different data sets [6].

This paper is organized as follows. After providing the necessary background on GPR, the prediction and training of regular GPR is introduced in Section 2. In Section 3, a novel sparse algorithm TGPR which uses the inducing inputs is introduced. Section 4 presents the simulation on three-dimension model. The conclusions of TGPR algorithm is in Section 5.

II GAUSSIAN PROCESS REGRESSION

A. Prediction

A Gaussian Process (GP) is a collection of random variables, any finite numbers of which have joint Gaussian distributions. It can be expressed as follows:

$$\boldsymbol{f} \sim \boldsymbol{N}(\boldsymbol{f} \mid \boldsymbol{m}, K) \tag{1}$$

Where, $\boldsymbol{m}$ is mean function and $K=k(x,x)$ is covariance function, they are both called kernel function which describe the probabilistic meaning of GPR [7].

Regression is that, first get the function relation between measurement inputs X_m and measurement outputs $\boldsymbol{f_m}$, being $\boldsymbol{f_m} = f\left(X_m\right)$. Then predict the unknown test outputs $\boldsymbol{f}_*$ given test inputs X_*. The first step is called training while the second step is called prediction.

Considering the measurement input set $X_m = \{x_{m_1}, x_{m_2}, \ldots\ldots, x_{m_{n_m}}\}$ that consists of n_m points, and the test input set $X_* = \{x_{*_1}, x_{*_2}, \ldots\ldots, x_{*_{n_*}}\}$ that consists of n_* points. The prior distribution of $\boldsymbol{f_m}$ and $\boldsymbol{f}_*$ is given by:

$$\begin{bmatrix} \boldsymbol{f_m} \\ \boldsymbol{f}_* \end{bmatrix} \sim \boldsymbol{N}\left(\begin{bmatrix} \boldsymbol{m_m} \\ \boldsymbol{m}_* \end{bmatrix}, \begin{bmatrix} K_{mm} & K_{m*} \\ K_{*m} & K_{**} \end{bmatrix}\right). \tag{2}$$

The measurement outputs are generally affected by Gaussian white noise $v \sim \boldsymbol{N}\left(0, \hat{\Sigma}_{f_m}\right)$ where $\hat{\Sigma}_{f_m} = diag(\hat{\sigma}^2_{f_{m1}}, \hat{\sigma}^2_{f_{m2}}, \ldots, \hat{\sigma}^2_{f_{mn_m}})$. As a result, the measurement output becomes $\hat{\boldsymbol{f}}_{\boldsymbol{m}} = \boldsymbol{f_m} + \boldsymbol{v}$. The posterior distribution of $\boldsymbol{f}_*$ can be calculated from the conditional probability formula:

$$\boldsymbol{f}_* \sim \boldsymbol{N}\left(\boldsymbol{m}_* + K_{*m}K_{mm}^{-1}(\boldsymbol{f_m} - \boldsymbol{m_m}), K_{**} - K_{*m}(K_{mm} + \hat{\Sigma}_{f_m})^{-1}K_{m*}\right) \tag{3}$$

B. Training

In the training step, the mean function generally is initialized to zero mean function, and covariance function is chosen as the well-known Squared Exponential covariance function (SE):

$$k(\boldsymbol{x},\boldsymbol{x}')=\lambda_f^2\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^T\boldsymbol{\Lambda}_x^{-1}(\boldsymbol{x}-\boldsymbol{x}')\right). \quad (4)$$

Where, $\lambda_f^2=k(\boldsymbol{x},\boldsymbol{x})$, $\boldsymbol{\Lambda}_x=\boldsymbol{\lambda}_x^2=diag\left(\lambda_{x_{n_1}}^2,\lambda_{x_{n_2}}^2,\ldots\ldots,\lambda_{x_{n_m}}^2\right)$ while $\lambda_{x_{n_1}}^2,\lambda_{x_{n_2}}^2,\ldots\ldots,\lambda_{x_{n_m}}^2$ is length scale for each input.

Then there is a set $\boldsymbol{\theta}=\{\boldsymbol{\lambda}_x,\lambda_f,\hat{\sigma}_{f_m}\}$, in which all parameters are called hyperparameters [8]. Only with the truest set of hyperparameters called the most likely hyperparameters, the model can been explained exactly by the function relation between X_m and $\boldsymbol{f_m}$. The most likely hyperparameters can be chosen using Maximum Likelihood method. According to Bayesian theory, the posterior hyperparameter distribution is found as follows:

$$p\left(\boldsymbol{\theta}|\hat{\boldsymbol{f}}_{\boldsymbol{m}},X_m\right)=\frac{p\left(\hat{\boldsymbol{f}}_{\boldsymbol{m}}|\boldsymbol{\theta},X_m\right)p\left(\boldsymbol{\theta}|X_m\right)}{p\left(\hat{\boldsymbol{f}}_{\boldsymbol{m}}|X_m\right)}. \quad (5)$$

The key now is to find the maximum of (5). Remember that the covariance function is SE, now take the logarithm of (5), and then consider the derivative with respect to the hyperparameters:

$$\frac{\partial\log(p)}{\partial\boldsymbol{\theta}}=-\frac{1}{2}tr\left(P^{-1}\frac{\partial P}{\partial\boldsymbol{\theta}}\right)+\frac{1}{2}tr\left(\boldsymbol{\alpha}^T\frac{\partial P}{\partial\boldsymbol{\theta}}\boldsymbol{\alpha}\right)=\frac{1}{2}tr\left(\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^T-P^{-1}\right)\frac{\partial P}{\partial\boldsymbol{\theta}}\right). \quad (6)$$

Where $P=\left(K_{mm}+\hat{\Sigma}_{f_m}\right)$, $\boldsymbol{\alpha}=P^{-1}\left(\hat{\boldsymbol{f}}_{\boldsymbol{m}}-\boldsymbol{m}_{\boldsymbol{m}}\right)$. Note that $\boldsymbol{\theta}=\{\lambda_x,\lambda_f,\hat{\sigma}_{f_m}\}$, it means there will be three derivatives. Applying gradient ascent algorithm to all these derivatives, the most likely hyperparameters can be found [9].

C. *Downsides of GPR*

In the training step of GPR, the main computation focus on inverting the matrix $K_{mm}+\hat{\Sigma}_{f_m}$, resulting $\mathrm{O}\left(n_m^3\right)$ runtime requirement. In the prediction step of GPR, it takes $\mathrm{O}\left(n_m^2n_*\right)$ time to calculate $\boldsymbol{f}_*$. When the number of measurements n_m grows big, GPR runs into computational problems, requiring amounts of time.

III TWO-STEP GAUSSIAN PROCESS REGRESSION

A. *Speeding up Prediction: Using Inducing Inputs*

The inducing inputs set $X_u=\{x_{u_1},x_{u_2},\ldots\ldots,x_{u_{n_u}}\}$ that consists of n_u inducing inputs [10] is introduced to speed up prediction.

STEP 1. Calculating the posterior distribution of the inducing outputs $\boldsymbol{f_u}$ given measurements $(X_m,\boldsymbol{f_m})$.

Assuming that $\boldsymbol{f}_m$ and $\boldsymbol{f}_*$ are conditionally independent given $\boldsymbol{f_u}$. The prior distribution of $\boldsymbol{f_m}$ and $\boldsymbol{f_u}$ is given by:

$$\begin{bmatrix}\boldsymbol{f_m}\\ \boldsymbol{f_u}\end{bmatrix}\sim\boldsymbol{N}\left(\begin{bmatrix}\boldsymbol{m_m}\\ \boldsymbol{m_u}\end{bmatrix},\begin{bmatrix}K_{mm} & K_{mu}\\ K_{um} & K_{uu}\end{bmatrix}\right). \quad (7)$$

The outputs $\boldsymbol{f_m}$ are still affected by Gaussian white noise $\boldsymbol{v}\sim\boldsymbol{N}\left(0,\hat{\Sigma}_{f_m}\right)$ and can be written as:

$$\boldsymbol{f_m}=\hat{\boldsymbol{f}}_{\boldsymbol{m}}-\boldsymbol{v}\sim\boldsymbol{N}(\hat{\boldsymbol{f}}_{\boldsymbol{m}},\hat{\Sigma}_{f_m}). \quad (8)$$

There are two distributions for $\boldsymbol{f_m}$, (7) and (8). Here the idea is to combine these two distributions. The operation called "combine" is introduced as follows:

Assuming that, for n independent measurements, the random variable $\boldsymbol{x}$ can be explained by n distributions, being $\boldsymbol{N}(\boldsymbol{m_1},K_1),\ldots,\boldsymbol{N}(\boldsymbol{m_n},K_n)$. Then the posterior distribution for $\boldsymbol{x}$ can be calculated by combining these n distributions. Here use the operator $\oplus$ as "combine", the posterior distribution for $\boldsymbol{x}$ is now given by:

$$\boldsymbol{f}_x(\boldsymbol{x})\sim\boldsymbol{N}(\boldsymbol{\mu}_x,\Sigma_x)=\boldsymbol{N}\left(\boldsymbol{m}_1,K_1\right)\oplus\ldots\oplus\boldsymbol{N}\left(\boldsymbol{m}_n,K_n\right). \quad (9)$$

$$\boldsymbol{\mu}_x=\Sigma_x\left(K_1^{-1}\boldsymbol{m_1}+K_2^{-1}\boldsymbol{m_2}+\ldots+K_n^{-1}\boldsymbol{m_n}\right). \quad (10)$$

$$\Sigma_x=\left(K_1^{-1}+K_2^{-1}+\ldots+K_n^{-1}\right)^{-1}. \quad (11)$$

Then the posterior distribution of $\boldsymbol{f_m}$ and $\boldsymbol{f_u}$ can be calculated by combining (7) and (8):

$$\begin{bmatrix}\boldsymbol{f_m}\\ \boldsymbol{f_u}\end{bmatrix}\sim\boldsymbol{N}\left(\begin{bmatrix}\boldsymbol{\mu_m}\\ \boldsymbol{\mu_u}\end{bmatrix},\begin{bmatrix}\Sigma_{mm} & \Sigma_{mu}\\ \Sigma_{um} & \Sigma_{uu}\end{bmatrix}\right). \quad (12)$$

$$\begin{bmatrix}\boldsymbol{\mu_m}\\ \boldsymbol{\mu_u}\end{bmatrix}=\begin{bmatrix}\Sigma_{mm}\left(K_{mm}^{-1}\boldsymbol{m_m}+\hat{\Sigma}_{f_m}^{-1}\hat{\boldsymbol{f}}_{\boldsymbol{m}}\right)\\ \boldsymbol{m_u}+K_{um}\left(K_{mm}+\hat{\Sigma}_{f_m}\right)^{-1}\left(\hat{\boldsymbol{f}}_{\boldsymbol{m}}-\boldsymbol{m_m}\right)\end{bmatrix}. \quad (13)$$

$$\begin{bmatrix}\Sigma_{mm} & \Sigma_{mu}\\ \Sigma_{um} & \Sigma_{uu}\end{bmatrix}=\begin{bmatrix}K_{mm}\left(K_{mm}+\hat{\Sigma}_{f_m}\right)^{-1}\hat{\Sigma}_{f_m} & \hat{\Sigma}_{f_m}\left(K_{mm}+\hat{\Sigma}_{f_m}\right)^{-1}K_{mu}\\ K_{*m}\left(K_{mm}+\hat{\Sigma}_{f_m}\right)^{-1}\hat{\Sigma}_{f_m} & K_{uu}-K_{um}\left(K_{mm}+\hat{\Sigma}_{f_m}\right)^{-1}K_{mu}\end{bmatrix}. \quad (14)$$

From (12), not only the posterior distribution of inducing outputs $\boldsymbol{f_u}$ is found, but also that of the measurement outputs $\boldsymbol{f_m}$ which are noiseless. It is one of the advantages of TGPR algorithm.

STEP 2. Just using the posterior distribution of $\boldsymbol{f_u}$ to predict the test outputs $\boldsymbol{f_*}$.

The prior distribution of $\boldsymbol{f_u}$ and $\boldsymbol{f_*}$ is given by:

$$\begin{bmatrix} \boldsymbol{f_u} \\ \boldsymbol{f_*} \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{m_u} \\ \boldsymbol{m_*} \end{bmatrix}, \begin{bmatrix} K_{uu} & K_{u*} \\ K_{*u} & K_{**} \end{bmatrix} \right). \tag{15}$$

Another distribution of $\boldsymbol{f_u}$ is from (12), it is:

$$\boldsymbol{f_u} \sim \boldsymbol{N}\left(\boldsymbol{\mu_u}, \Sigma_{uu}\right). \tag{16}$$

The idea is still to combine (15) and (16). But note that, the prior knowledge of $\boldsymbol{f_u}$, being $\boldsymbol{f_u} \sim \boldsymbol{N}\left(\boldsymbol{m_u}, K_{uu}\right)$, has been used in both of these two distribution of $\boldsymbol{f_u}$. Combining (15) and (16) will use the prior knowledge twice, which is unacceptable, it is necessary to separate that prior for one time. Here use the operator $!$ as "separate", so the operation called "separate" is introduced as follows:

$$\boldsymbol{N}(\boldsymbol{\mu_x}, \Sigma_x) ! \ \boldsymbol{N}\left(\boldsymbol{m_u}, K_{uu}\right) = \boldsymbol{N}(\boldsymbol{\mu_x}, \Sigma_x) \oplus \boldsymbol{N}\left(\boldsymbol{m_u}, -K_{uu}\right). \tag{17}$$

Then the posterior distribution of $\boldsymbol{f_u}$ and $\boldsymbol{f_*}$ can be calculated by combining (15) and (16), and separating $\boldsymbol{N}\left(\boldsymbol{m_u}, K_{uu}\right)$ from the result for one time:

$$\begin{bmatrix} \boldsymbol{f_u} \\ \boldsymbol{f_*} \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{\mu_u} \\ \boldsymbol{\mu_*} \end{bmatrix}, \begin{bmatrix} \Sigma_{uu} & \Sigma_{u*} \\ \Sigma_{*u} & \Sigma_{**} \end{bmatrix} \right). \tag{18}$$

$$\begin{bmatrix} \boldsymbol{\mu_u} \\ \boldsymbol{\mu_*} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\mu_u} \\ \boldsymbol{m_*} + K_{*u} K_{uu}^{-1} \left(\boldsymbol{\mu_u} - \boldsymbol{m_u} \right) \end{bmatrix}. \tag{19}$$

$$\begin{bmatrix} \Sigma_{uu} & \Sigma_{u*} \\ \Sigma_{*u} & \Sigma_{**} \end{bmatrix} = \begin{bmatrix} \Sigma_{uu} & \Sigma_{uu} K_{uu}^{-1} K_{u*} \\ K_{*u} K_{uu}^{-1} \Sigma_{uu} & K_{**} - K_{*u} K_{uu}^{-1} \left(K_{uu} - \Sigma_{uu} \right) K_{uu}^{-1} K_{u*} \end{bmatrix}. \tag{20}$$

Performance of TGPR.

Assuming K_{uu}^{-1} has been calculated in STEP 1., calculating the posterior distribution of $\boldsymbol{f_*}$ only takes $\mathrm{O}\left(n_u n_*^2\right)$ time. Compared with $\mathrm{O}\left(n_m^2 n_*\right)$ time regular GPR takes, TGPR algorithm has speeded up prediction.

B. Speeding up Training: Train $\boldsymbol{n_m}$ Times Individually

STEP 1. Calculating $\boldsymbol{f_u}$ individually using each of n_m measurements.

Assuming that each test output $f_{m_1}, \ldots, f_{m_{n_m}}$ is fully independent given $\boldsymbol{f_u}$. Firstly use the first measurement $(\boldsymbol{x_{m_1}}, \hat{f}_{m_1})$ to calculate the distribution of $\boldsymbol{f_u}$, resulting in $\boldsymbol{f_u^1} \sim \boldsymbol{N}(\boldsymbol{\mu_u^1}, \Sigma_{uu}^1)$. Then use the second measurement $(\boldsymbol{x_{m_2}}, \hat{f}_{m_2})$ to find $\boldsymbol{f_u^2} \sim \boldsymbol{N}(\boldsymbol{\mu_u^2}, \Sigma_{uu}^2)$, and so on. At last n_m distributions of $\boldsymbol{f_u}$ are calculated.

STEP 2. Combining n_m distributions of $\boldsymbol{f_u}$.

The next thing to do is to combine all these n_m distributions together. Note that, each of these distributions contains the prior knowledge of $\boldsymbol{f_u}$. Combining n_m distributions will use the prior knowledge n_m times, so separating that prior distribution for n_m times is indispensable:

$$\boldsymbol{f_u} \sim \boldsymbol{N}(\boldsymbol{\mu_u}, \Sigma_{uu}) \propto \boldsymbol{N}\left(\boldsymbol{\mu_u^1}, \Sigma_{uu}^1\right) \oplus \boldsymbol{N}\left(\boldsymbol{\mu_u^2}, \Sigma_{uu}^2\right) \oplus \ldots \oplus \boldsymbol{N}\left(\boldsymbol{\mu_u^{n_m}}, \Sigma_{uu}^{n_m}\right) ! \underbrace{\boldsymbol{N}\left(\boldsymbol{m_u}, K_{uu}\right)}_{(n_m - 1)\text{times}}. \tag{21}$$

Then the posterior distribution of $\boldsymbol{f_m}$ and $\boldsymbol{f_u}$ becomes:

$$\begin{bmatrix} \boldsymbol{f_m} \\ \boldsymbol{f_u} \end{bmatrix} \sim N\left(\begin{bmatrix} \boldsymbol{\mu_m} \\ \boldsymbol{\mu_u} \end{bmatrix}, \begin{bmatrix} \Sigma_{mm} & \Sigma_{mu} \\ \Sigma_{um} & \Sigma_{uu} \end{bmatrix} \right). \tag{22}$$

$$\begin{bmatrix} \boldsymbol{\mu_m} \\ \boldsymbol{\mu_u} \end{bmatrix} = \begin{bmatrix} \boldsymbol{m_m} + \Sigma_{mm} \hat{\Sigma}_{f_m}^{-1} \left(\hat{\boldsymbol{f}}_{\boldsymbol{m}} - \boldsymbol{m_m} \right) \\ \boldsymbol{m_u} + \Sigma_{um} \hat{\Sigma}_{f_m}^{-1} \left(\hat{\boldsymbol{f}}_{\boldsymbol{m}} - \boldsymbol{m_m} \right) \end{bmatrix}. \tag{23}$$

$$\begin{bmatrix} \Sigma_{mm} & \Sigma_{mu} \\ \Sigma_{um} & \Sigma_{uu} \end{bmatrix} = \begin{bmatrix} \Sigma_{mm} & \hat{\Sigma}_{f_m} \left(\Lambda_{mm} + \hat{\Sigma}_{f_m} \right)^{-1} K_{mu} \Delta_{uu}^{-1} K_{uu} \\ K_{uu} \Delta_{uu}^{-1} K_{um} \left(\Lambda_{mm} + \hat{\Sigma}_{f_m} \right)^{-1} \hat{\Sigma}_{f_m} & K_{uu} \Delta_{uu}^{-1} K_{uu} \end{bmatrix}. \tag{24}$$

$$\Sigma_{mm} = \left(\Lambda_{mm}^{-1} + \hat{\Sigma}_{f_m}^{-1} \right)^{-1} + \hat{\Sigma}_{f_m} \left(\Lambda_{mm} + \hat{\Sigma}_{f_m} \right)^{-1} K_{mu} \Delta_{uu}^{-1} K_{um} \left(\Lambda_{mm} + \hat{\Sigma}_{f_m} \right)^{-1} \hat{\Sigma}_{f_m}. \tag{25}$$

Where $\Lambda_{mm} = diag\left(K_{mm} - K_{mu} K_{uu}^{-1} K_{um} \right)$, $\Delta_{uu} = K_{uu} + K_{um} \left(\Lambda_{mm} + \hat{\Sigma}_{f_m} \right)^{-1} K_{mu}$.

Performance of TGPR

The combined posterior distribution of $\boldsymbol{f_u}$ can be obtained from (22), by which the training only takes $O(n_m n_u^2)$ time. Compared with $O\left(n_m^3\right)$ time regular GPR takes, TGPR algorithm has speeded up training.

C. Choosing the Inducing Inputs

The assumptions of TGPR algorithm cause K_{mm} to become:

$$K_{mm}=K_{mu}K_{uu}^{-1}K_{um}+diag(K_{mm}-K_{mu}K_{uu}^{-1}K_{um})=K_{mu}K_{uu}^{-1}K_{um}+\Lambda_{mm}. \quad (26)$$

This means that K_{mm} now depends on K_{uu}. The idea to choose inducing inputs is to treat inducing inputs set X_u as another hyperparameter. That is, maximize the posterior hyperparameter distribution $p\left(\boldsymbol{\theta}|\hat{\boldsymbol{f}}_{\boldsymbol{m}}, X_m\right)$ with respect to K_{uu}. $p\left(\hat{\boldsymbol{f}}_{\boldsymbol{m}}|X_m, X_u\right)$ in (5) now becomes:

$$p\left(\hat{\boldsymbol{f}}_{\boldsymbol{m}}|X_m, X_u\right)=\boldsymbol{N}(\hat{\boldsymbol{f}}_{\boldsymbol{m}}|\boldsymbol{m}_{\boldsymbol{m}}, K_{mm}+\hat{\Sigma}_{f_m}). \quad (27)$$

Then by applying the Maximum Likelihood method in Section 2.2 to (26) and (27), the most likely inducing inputs will be found.

IV SIMULATION

The simulations are performed on a personal computer. And the Gaussian function used in the simulation is the semi-randomly chosen function as follows:

$$f(x_1,x_2)=\left(\frac{x_1}{4}\right)^2+\left(\frac{x_2}{4}\right)^2-\sin\left(2\pi\frac{x_1}{5}\right)\left(1-\frac{1}{2}\cos\left(2\pi\frac{x_2}{4}\right)\right)+\sin\left(2\pi\frac{x_2}{5}\right)-1. \quad (28)$$

The three-dimension model of (28), which lies on the interval $x_1, x_2 \in [-2,2]$, is shown in Figure I.

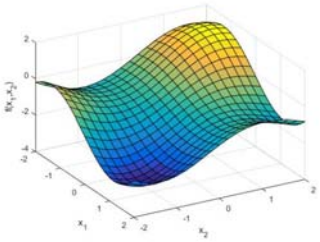
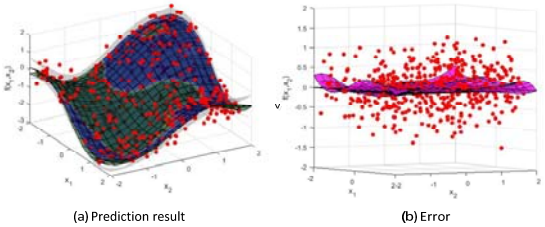


FIGURE I. THE THREE-DIMENSION MODEL

In simulations, the mean is zero mean function, while the covariance is SE with $\lambda_f=\lambda_{x_1}=\lambda_{x_2}=1$. The function is affected by Gaussian white noise with $\hat{\sigma}_{f_m}=0.5$.

One of the prediction results for above model using TGPR with $n_m=500$ is shown in Figure II(a), while the error is shown in Figure II(b). The red dots are measurements; the blue plane is the exact function while the green plane is the prediction result; the translucent gray planes indicate 95% confidence intervals (CIs); the purple plane indicates error, being the prediction result minus the exact function.



(a) Prediction result (b) Error

FIGURE II. THE PREDICTION RESULT AND ERROR

Five simulations for further analyses and comparisons are performed in the case of $n_u=25$ and $n_u=100$ respectively. The results are the averages of 100 simulation runs and are shown in Table I. The root mean square error (RMSE) is calculated over inputs to analyze the accuracy of the algorithms. A lower RMSE means a better accuracy.

TABLE I. SIMULATION RESULTS

Simulation	$n_u=25$			$n_u=100$		
	n_m	Runtime	RMSE	n_m	Run time	RMSE
1. Regular GPR with $n_m=5000$	5000	2.09s	0.048	5000	2.01 s	0.049
2. Regular GPR with $n_m=10000$	10000	10.11s	0.038	10000	9.95 s	0.038
3. TGPR with $n_m=5000$	5000	0.028s	0.056	5000	0.065s	0.050
4. TGPR with $n_m=10000$	10000	0.046s	0.047	10000	0.109s	0.038
5. TGPR with the same runtime as 1	1000000	2.09s	0.042	250000	2.01 s	0.0086

The analyses for Table I are as follows:

(1) No matter which GPR algorithm is applied, the more measurements used in training step, the more accurate the prediction becomes, and more time the prediction will take.

(2) In TGPR algorithm, the more inducing inputs used, the more accurate the prediction becomes, and more time the prediction will take.

(3) In regular GPR, as the number of measurements grows big, there is a significant increase of runtime (Simulation 1 and 2).

(4) Given the same number of measurements, compared with regular GPR, TGPR reduces the regression time with a slight decrease in accuracy (Simulation 1 and 3).

(5) As the number of measurements grows from 5000 to 10000, the improvement of regression performance that gains from TGPR algorithm is more significant. TGPR reduces the regression time without any decrease in accuracy (Simulation 2 and 4).

(6) Given the same amount of runtime, compared with regular GPR, TGPR algorithm noticeably improves the accuracy of prediction because it can incorporate more measurements (Simulation 1 and 5).

V CONCLUSION

For reducing the computational complexity in regular GPR, Two-step Gaussian Process Regression algorithm is proposed in this paper. By using the inducing inputs, TGPR reduces the prediction time from $\mathrm{O}\left(n_m^2 n_*\right)$ to $\mathrm{O}\left(n_u n_*^2\right)$. By training n_m times individually, TGPR reduces the training time from $\mathrm{O}\left(n_m^3\right)$ to $\mathrm{O}(n_m n_u^2)$. The performance of TGPR algorithm and regular GPR are compared based on several simulations. The results show that, when working on a large number of measurements, TGPR algorithm significantly speeds up training and prediction. And due to that more measurements can be incorporated in less time, TGPR algorithm also provides a more accurate prediction.

ACKNOWLEDGEMENTS

This research is supported by Key Projects in the National Science & Technology Support Program under No. 2013BAK06B03 and the Natural Science Foundation of China (61673050).

REFERENCE

[1] Rasmussen, Carl Edward, and Christopher KI Williams. Gaussian processes for machine learning. Vol. 1. Cambridge: MIT press, 2006.

[2] Zhang, Guoqiang, B. Eddy Patuwo, and Michael Y. Hu. Forecasting with artificial neural networks: The state of the art. International journal of forecasting 14.1 (1998): 35-62.

[3] Cristianini, Nello, and John Shawe-Taylor. An introduction to support vector machines and other kernel-based learning methods. Cambridge university press, 2000.

[4] Williams, Christopher KI, and Matthias Seeger. Using the Nyström method to speed up kernel machines. Advances in neural information processing systems. 2001.

[5] Snelson, Edward, and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. Advances in neural information processing systems. 2006.

[6] Tresp, Volker. A Bayesian committee machine. Neural computation 12.11 (2000): 2719-2741.

[7] Titsias, Michalis K., and Neil D. Lawrence. Bayesian Gaussian process latent variable model. International Conference on Artificial Intelligence and Statistics. 2010.

[8] Blum, Manuel, and Martin A. Riedmiller. Optimization of Gaussian process hyperparameters using Rprop. ESANN. 2013.

[9] Bottou, Léon. Large-scale machine learning with stochastic gradient descent. Proceedings of COMPSTAT'2010. Physica-Verlag HD, 2010. 177-186.

[10] Quiñonero-Candela, Joaquin, and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. Journal of Machine Learning Research 6.Dec (2005): 1939-1959.