An introduction of Gaussian processes and deep Gaussian processes and their applications to speech processing

Tomoki Koriyama*

Graduate School of Information Science and Technology, The University of Tokyo, 7–3–1 Hongo, Bunkyo-ku, Tokyo, 113–8656 Japan

Abstract: Gaussian process (GP) is a distribution of functions, which can be used for a machine learning framework. GP regression has characteristics of Bayesian model, which can predict uncertainty of outputs, and kernel methods, which enables nonlinear function with a small number of parameters. In this paper, we first describe the basic of GP regression, and introduce recent notable advances of GP. Specifically, we focus on stochastic variational GP that is an approximation method available for a huge amount of training data, and explain a GP-based deep architecture model called deep Gaussian process. Since GP regression is a general-purpose machine learning framework, there are many applications. In this paper, we introduce GP-based applications to speech information processing including speech synthesis.

Keywords: Gaussian process regression, Kernel methods, Deep learning, Bayesian model, Speech processing

PACS number: 43.10.Ln [doi:10.1250/ast.41.457]

1. INTRODUCTION

Recently, deep learning, which uses a function defined by deep neural networks (DNNs) becomes popular and it caused artificial intelligence (AI) boom. Gaussian process (Gaussian process, GP) is used as another machine learning framework that predicts the function [1]. In the analysis of the behavior of DNNs, GP is attracted because is is related to the DNN with an infinite number of hidden units [2,3]. In this paper, we introduce the basics of machine learning using Gaussian process and its application to speech information processing. In general speech information processing, a large amount of training data is used and the relationship between input and output features is complicated. Therefore, we explain an approximation method of GP available for an arbitrary amount of training data, and describe deep Gaussian process for expressing complicated functions.

2. MACHINE LEARNING AND BAYESIAN TRAINING

We briefly introduce a general machine learning framework in this section. In typical machine learning frameworks, we predict a function \( f(x) \) that expresses the relationship among variables using training data. For example, in the regression task in which we predict a scalar value \( y \in \mathbb{R} \) from a \( D \)-dimensional vector \( x \in \mathbb{R}^D \), it is assumed that the relationship is represented using \( f : \mathbb{R}^D \rightarrow \mathbb{R} \) by

\[
y = f(x) + \epsilon
\]  

(1)

where \( \epsilon \) is an observation noise. In the case of binary classification where an output variable is yes (\( y = 1 \)) or no (\( y = 0 \)), \( y \) can be predicted as follows:

\[
y = \begin{cases} 
1 & (f(x) \geq 0) \\
0 & (f(x) < 0).
\end{cases}
\]  

(2)

Since the use of a neural network (NN) as the function has become popular in recent years, we take NN-based framework as an example to explain machine learning. When we use a neural network with one hidden layer and \( D_H \) hidden units, the function is expressed by the following equation

\[
f_\theta(x) = b^{(2)} + w^{(2)T} \psi(b^{(1)} + W^{(1)T} x)
\]  

(3)

where \( w^{(2)} \in \mathbb{R}^{D_H} \), \( W^{(1)} \in \mathbb{R}^{D \times D_H} \) are weight parameters and \( b^{(2)} \in \mathbb{R} \), \( b^{(1)} \in \mathbb{R}^{D_H} \) are bias ones. We represent the parameter set as \( \theta = (w^{(2)}, W^{(1)}, b^{(2)}, b^{(1)}) \), \( \psi \) is an element-wise activation function, which we typically use a hyperbolic tangent (tanh) or rectified linear unit (ReLU) function.

Let \( X = [x_1, \ldots, x_N]^T \in \mathbb{R}^{N \times D} \) and \( y = [y_1, \ldots, y_N]^T \in \mathbb{R}^N \) be input vectors and output variables, respectively. Here, we consider predicting a function \( f(\cdot) \) from
the training data $\mathcal{D} = (X, y)$. A widely-used method is a point estimate of model parameters $\theta$ using the frameworks of maximum likelihood or maximum a posteriori. After training parameters, we use the function $f_\theta$ obtained from the optimal parameters $\hat{\theta}$ to predict the output $f_\theta(x^*)$ given an unknown input vector $x^*$.

Bayesian learning is another approach to predicting the output from an unknown input, in which the posterior distributions of parameters are inferred. Based on Bayes’ theorem, a posterior $p(\theta | \mathcal{D})$ is calculated as follows:

$$ p(\theta | \mathcal{D}) = \frac{p(\mathcal{D} | \theta)p(\theta)}{p(\mathcal{D})} $$

where $p(\theta)$ is a prior which assumes how parameters are distributed, and $p(\mathcal{D} | \theta)$ is a likelihood that represents the goodness of fit of parameters to data. $p(\mathcal{D})$ is a marginal distribution, which is independent of parameters.

The important point of Bayesian learning is that we have the distributions of parameters. Since the function is uniquely determined from the parameter, we can obtain the prior and posterior distributions of the function from those parameters. We show an example of the distributions of functions sampled from the parameter distribution of a 1-hidden-layer neural network in Fig. 1(a). The method that infers the parameter distribution of a neural network is referred to as Bayesian neural network.

The purpose of Bayesian inference is to obtain the following predictive distribution by integrating out the posterior distribution of the function:

$$ p(y_* | x_*) = \int p(f_\theta | \mathcal{D})p(y_* | f_\theta(x_*))df_\theta $$

(5)

Note that the predictive distribution does not depend on specific parameters but on the model itself.

Bayesian model selection is another property of Bayesian learning. In Bayesian model selection, the performance of a model, which is independent of its parameters, can be evaluated using marginal likelihood (also called model evidence) given by

$$ p(\mathcal{D}) = \int p(\mathcal{D} | \theta)p(\theta)\,d\theta. $$

(6)

We can choose an appropriate model for given data by comparing the marginal likelihoods.

The issue of Bayesian learning is that the calculations of predictive distributions and marginal likelihoods tend to be hard. Moreover, to derive closed forms of predictive distributions, the definitions of priors are often restricted.

### 3. GAUSSIAN PROCESS REGRESSION

#### 3.1. Gaussian Process

In machine learning based on Gaussian process, we assume the distribution over functions by considering the statistic of function values, instead of the distributions over parameters. Let $X = [x_1, \ldots, x_N]^T$ and $f = [f(x_1), \ldots, f(x_N)]^T$ denote an input variable sequence and a function variable sequence, respectively, and we assume that the joint distribution of $f$ becomes a multivariate Gaussian distribution. Here, if the joint distribution of $f$ becomes a multivariate Gaussian distribution for arbitrary input sequence, the distribution over the function is referred to as Gaussian process. Gaussian process is expressed as follows:

$$ f \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot)). $$

(7)

Gaussian process is formulated by a mean function $m(\cdot)$ and a kernel (covariance) function $k(\cdot, \cdot)$. The mean function represents that the mean of $f_1(x), f_2(x), \ldots$ becomes $m(x)$ for any input $x$ when the sampled functions are $f_1, f_2, \ldots$. Similarly, the variance of $(f_1(x), f_2(x), \ldots)$ corresponds to $k(x, x)$ and the covariance between $(f_1(x), f_2(x), \ldots)$ and $(f_1(x'), f_2(x'), \ldots)$ is defined by $k(x, x')$. In practice, a constant function $m(x) = 0$ is generally used as the mean function, which is called zero-mean function. Figure 1(b) shows an example of functions sampled from a Gaussian process using an exponential quadratic (EQ) kernel (See Sect. 3.3.) as a kernel function. With the EQ kernel, the covariance decreases with the distance between inputs, which derives the nature “If the inputs are similar, the outputs become similar.” in the sampled functions.

The joint distribution of $f$ using the zero-mean is expressed as follows:

$$ p(f) = \mathcal{N}(f; 0, K_N) $$

(8)

$$ K_N = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_N, x_1) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{bmatrix} $$

(9)

where $0$ denotes a zero vector whose elements are all zero,
We derive the predictive distribution given unknown input from a training data \( D = (X, y) \). Let \( X_T = \{x_1, \ldots, x_T\}^T \in \mathbb{R}^{T \times D} \) and \( f_T = [f(x_1), \ldots, f(x_T)]^T \in \mathbb{R}^T \) be sequences of unknown inputs and latent function variables, respectively. From the definition of Gaussian process, the joint distribution of \( f_T \) and \( f \) is expressed by the following Gaussian distribution:

\[
p(f, f_T | X_T, D) = \mathcal{N}
\begin{pmatrix}
f \\
f_T
\end{pmatrix}; \begin{bmatrix} K_N & K_{NT} \\
K_{TN} & K_T
\end{bmatrix} \tag{10}
\]

where \( K_{TN} = K_{NT}^T \) is a Gram matrix between \( X_T \) and \( X \), \( K_T \) is a Gram matrix obtained from \( X_T \). We adopt the property of Gaussian distribution: when the joint distribution is a Gaussian, the conditional distribution is also Gaussian. Then, we obtain the predictive distribution as follows:

\[
p(f_T | X_T, D) = \mathcal{N}(f_T; \mu_T, \Sigma_T) \tag{11}
\]

\[
\mu_T = Ay \tag{12}
\]

\[
\Sigma_T = K_T - AK_{NT} \tag{13}
\]

\[
A = K_{TN}(K_N + \sigma^2_I I_N)^{-1}. \tag{14}
\]

We also obtain the predictive distribution of output variable \( y_T \) as follows:

\[
p(y_T | X_T, D) = \mathcal{N}(y_T; \mu_T, \Sigma_T + \sigma^2_I I_T) \tag{15}
\]

For the derivation of Gaussian process regression, [1] provides the detail.

### 3.3. Kernel Function Selection

To perform Gaussian processes regression represented by (6), we have to define the kernel function that determines the values of Gram matrices. A widely used kernel is exponential quadratic (EQ) kernel defined by \( k(x, x') = \exp(-\|x - x'\|^2/2) \). EQ kernel is also referred to as radial basis function (RBF) kernel, squared potential (SE) kernel, or Gaussian kernel. The EQ kernel determines the value using the distance of two inputs of \( x \) and \( x' \). Besides the EQ kernel, whatever kernel functions can be used if the kernel function is positive definite. For example, the sum of two kernel functions is also a positive definite kernel. We can design a new kernel function by multiplying two kernel functions.

Using the correspondence between Gaussian process and neural network with an infinite number of hidden units, we can consider the relationship between kernel functions and activation functions as shown in Table 1. The EQ kernel is related to a cosine function. Random Fourier features are approximation methods for kernel-based frameworks using this relationship [5]. The error function in the Table is a function that has a similar contour to a sigmoid and hyperbolic tangent (tanh) function. Lee et al. recently showed that it is possible to define kernel function for arbitrary activation functions [3].

The selection of an appropriate kernel function is not an easy task as the selection of an activation function is not. A basic principle is to choose the kernel that represents the relationship between the training data points. To overcome the kernel selection issue, deep Gaussian processes (described in Sect. 5) are proposed, where the input features are warped by a hierarchical architecture.

### 3.4. Training of Gaussian Process Regression

In the Gaussian process regression, hyperparameters such as kernel function parameters and noise variances are trained on the basis of Bayesian model selection described in Sect. 2, namely, the marginal likelihood is maximized to optimize the hyperparameters. The log marginal likelihood is given by

\[
\log p(y|X) = -\frac{1}{2} y^\top (K_N + \sigma^2_I I)^{-1} y
\]

\[
-\frac{1}{2} \log |K_N + \sigma^2_I I| + \text{const.} \tag{16}
\]

The first and second terms in (16) represent the goodness of fit of the kernel function to the data and the complexity of the model, respectively. The complexity term works as the penalty term that prevents an overfitting problem.

### 3.5. Characteristics of Gaussian Process Regression

Gaussian process regression has several characteristics such as nonparametric model, kernel methods, Bayesian model, and Gaussian.

Nonparametric model means that we utilize all samples of training data for prediction instead of parameterizing the training data into a fixed amount of parameters. This is

| Activation                  | Kernel     |
|-----------------------------|------------|
| cos                         | EQ [5]     |
| ReLU                        | ArcCos [6] |
| error function              | ArcSin [7] |
| arbitrary function          | NNGP [3]   |

\( K_N \) is referred to as a Gram matrix.

It is known that Gaussian processes are closely related to neural networks [3,4]. Specifically, if a neural network has an infinite number of hidden units and its weight parameters are distributed independently and identically on Gaussian distributions, the function represented by the neural network becomes a sample of Gaussian process.
confirmed by showing (12), which represents the predictive mean is directly calculated using the weighted sum of training data points $y$.

The advantage of kernel methods is the complicated functions can be predicted with a small number of parameters. Moreover, kernel methods enable us to use a diverse type of input features without the restriction of the use of continuous variable features. For example, we can use structured input features such as trees and probability distribution functions [8].

Gaussian process regression also has the characteristics of Bayesian modeling. Specifically, the training is performed based on Bayesian model selection that considers the model complexity. Also, the uncertainty of output feature can be obtained by Bayesian inference.

Furthermore, since Gaussian process is the assumption of Gaussian distribution over function variables, we can utilize the properties of Gaussian distributions. For example, the derivation of conditional distribution in (11) is based on the property of Gaussian distribution. By utilizing these properties, we can obtain close forms of marginal and predictive distribution.

4. SCALABLE GAUSSIAN PROCESS COMPUTATION

4.1. Stochastic Variational Gaussian Process (SVGP)

One issue of Gaussian process regression is computational complexity. Let $N$ be the number of samples in training data. We require $O(N^3)$ storage for $(N \times N)$ Gram matrix and $O(N^3)$ computational complexity for inversion and determinant in (16) and (11). This restricts $N$ at most 10,000. However, speech processing requires much more training data points. For example, 60-minute speech data with a 5 ms frame shift exceeds 700,000 training data points.

One noteworthy technique to overcome the problem is an approximation method based on stochastic variational inference (SVI) [9,10], which is referred to as SVGP. The SVGP framework utilizes inducing point methods (also called pseudo data methods), which approximate the pairs of input $X$ and function output $f$ by a small number of representative pairs. We represent the pairs by inducing inputs $Z = (z_1, \ldots, z_M)$ and inducing outputs $u = (u_1, \ldots, u_M)$. $M (< N)$ is the number of inducing points, which is generally hundreds or approximately 1,000.

From the definition of Gaussian process, the prior distribution of $u$ is given by

$$p(u) = \mathcal{N}(u; 0, K_M) \quad (17)$$

where $K_M$ is a Gram matrix of input $Z$. Also, the joint distribution is given by

$$p(f, u) = \mathcal{N}\left( \left[ f \right]_M; 0, \begin{bmatrix} K_N & K_{NM} \\ K_{MN} & K_M \end{bmatrix} \right) \quad (18)$$

where $K_{MN} = K_{NM}^T$ is a Gram matrix between $Z$ and $X$.

In SVGP, we train the variational distribution $q(u) = \mathcal{N}(u; m, S)$ that approximates the posterior $p(u|y)$. Using the variational distribution $q(u)$, the predictive distribution is approximated by

$$p(f_T|y) = \int p(f_T|u, y)p(u|y)du$$

$$\approx \int p(f_T|u)p(u|y)du$$

$$\approx \int p(f_T|u)q(u)du$$

$$= \int \frac{p(f_T, u)}{p(u)}q(u)du$$

$$= \mathcal{N}(f_T; \mu_T, \Sigma_T) \triangleq q(f_T) \quad (19)$$

$$\mu_T = K_T M^{-1} m$$

$$\Sigma_T = K_T - K_T M^{-1} K_M + K_T M^{-1} S K_M^{-1} M K_T \quad (20)$$

These equations means that we just require $O(M^2)$ storage and $O(M^3)$ computational complexity. We here define the operation of calculating $q(f_T)$ as SVGP($f_T; X_T, Z, q(u)$).

The variational distribution $q(u)$ can be trained by a variational Bayesian method. Using Jensen’s inequality for log marginal likelihood (16), we obtain the following evidence lower bound (ELBO):

$$\mathcal{L} = \sum_{i=1}^{N} \mathbb{E}_{q(f(x_i))} [\log p(y_i|f(x_i))]$$

$$- \text{KL}(q(u)||p(u)) \quad (22)$$

where $\mathbb{E}_{q(f)}$ is expectation over the distribution $p(\cdot)$, and KL represents Kullback–Leibler divergence. By maximizing the ELBO, we can optimize the variational distribution parameters ($m, S$).

The advantage of SVGP is that the parameters can be optimized by the stochastic gradient method because ELBO is decomposed to the sum of respective training data points. Hence, it is possible to use a minibatch training available for a large amount of training data and employ adaptive optimization techniques such as AdaGrad and Adam.

Since SVGP does not restrict the kernel function, it can be used for various kinds of input features. Moreover, it is easy to apply SVGP to classification task because we only have to calculate the expectation $\mathbb{E}_{q(f(x_i))} [\log p(y_i|f(x_i))]$ in (22).
4.2. Other Approximation Methods of Gaussian Process Regression

Besides SVGP, diverse approximation methods have been proposed to overcome the problem of computational complexity [11]. In local GP, training data is partitioned into several clusters, and Gaussian process regression is executed for each cluster [12]. This enables us to use all training data points in prediction while reducing the computational complexity to the cube of the number of frames of each cluster. Product of experts [13] and Bayesian committee machine [14] combines the results of local GPs to enhance the prediction accuracy.

In the inducing point methods including SVGP, the number of inducing points is the limit of performance. KISS-GP [15] focuses on the structure of Gram matrices, which are approximated by Kronecker product and interpolation. Although KISS-GP can utilize much larger inducing points than SVGP, the dimension of input feature vectors has to be small (less than 4) due to the assumption of Kronecker structure. TT-GP [16] and SKIP [17] attempt to overcome the problem by approximating the Kronecker structure.

Other approaches to the reduction of computational complexity are the approximation of kernel function by the inner product of finite-dimensional vectors. Random Fourier features [5] and sparse spectrum GP (SSGP) [18] approximate EQ kernel with finite-dimensional orthogonal bases. SSGP has weight vectors as parameters instead of inducing points, the model structure is very close to Bayesian neural networks.

5. DEEP GAUSSIAN PROCESSES

One of the problems of Gaussian process regression is that the performance depends on the design of kernel functions. For example, EQ kernel assumes that the kernel function depends only on the distance between two input vectors. Thereafter, to focus the other measures such as norms and angles of vectors, we have to choose other kernel functions. However, it is a laborious work to choose the best kernel function.

Deep Gaussian process (DGP) [19] is proposed to overcome the problem by transforming the feature space of input variables using a deep architecture. In DGP, we assume that the function is \( f : \mathbb{R}^{D_h} \rightarrow \mathbb{R}^{D_h} \) expressed by a composite function of several functions

\[
f = f^L \circ \cdots \circ f^1
\]

\[
f^\ell = (f^{\ell,1}, \ldots, f^{\ell,D_\ell})
\]

and respective functions that output the value of dimension \( d \) and layer \( \ell \) are sampled from Gaussian processes. When the hidden layer variables are defined by

\[
H^{\ell-1} = f^{\ell-1}(\ldots (f^1(X))
\]

we can infer the predictive distribution of upper layer value \( h^{\ell,d} \) from \( H^{\ell-1} \), using Gaussian process regression. By repeating the inference until the final layer \( L \), we obtain the predictive distribution of output variable.

5.1. Stochastic Variational Inference for DGP

To achieve scalable training of DGP for a large amount of training data, Salimbeni et al. proposed a doubly stochastic variational inference (DSVI)-based DGP [20]. In DSVI-DGP, the inference of \( \ell \)-th layer from \((\ell-1)\)-th layer is calculated using the following equations in the same manner as (21):

\[
q(h^{\ell,d}_i) = \text{SVGP}(h^{\ell,d}_i; H^{\ell-1}_T, Z^\ell, q(u^{\ell,d}_i))
\]

\[
q(u^{\ell,d}_i) = \mathcal{N}(u^{\ell,d}_i; m^{\ell,d}_i, s^{\ell,d}_i)
\]

By repeating SVGP, we obtain the predictive distribution of output variable give by

\[
p(Y_T) = \int \cdots \int q(Y_T|H^2_T)q(H^1_T|H^{L-1}_T)
\]

\[
\ldots q(H_1|X_T)dH^L_T \ldots dH^1_T
\]

\[
q(H^\ell_T|H^{\ell-1}_T) = \prod_{d=1}^{D_\ell} q(h^{\ell,d}_i).
\]

Since the calculation of this integral is intractable, predictive means are practically used for 1 to \((L-1)\)-th layers.

The training of DSVI-based DGP is executed by maximizing the following ELBO in the same manner as SVGP:

\[
\mathcal{L} \approx \sum_{s=1}^{S} \sum_{i=1}^{N} \sum_{d=1}^{D_\ell} \mathbb{E}_{q(\theta^s)} \log p(y^d_i|H^\ell_T)\
- \sum_{\ell=1}^{L} \sum_{d=1}^{D_\ell} \text{KL}(q(u^{\ell,d}_i)||p(u^{\ell,d}_i))
\]

where \( S \) is the number of points for Monte Carlo sampling which is generally set to unit. \( q(\theta^s) \) is the predictive distribution of \( L \)-th layer. The Monte Carlo sampling is carried out using the predictive distribution for each layer, and the sampled values are used for the inference of the next layer.

5.2. The Relationship between Neural Networks and Deep Gaussian Processes

Deep Gaussian processes are closely related to (Bayesian) neural networks. Figure 2 shows the transition from a neural network to a Gaussian process. The left network in the figure is a 3-hidden-layer neural network. The hidden layer values \( g^\ell \) are obtained using a weight matrix \( W^\ell \) and an activation function. Here, we decompose
the weight matrix $W^t$ into two matrices $W^{t,a}$, $W^{t,b}$, and define the new hidden layer as $h^t$. If we increase the number of hidden units of $g^t$ to the infinite number, the relationship between $h^{t-1}$ and $h^t$ is represented by a Gaussian process. Therefore, a 3-hidden-layer neural network can be transformed into a 3-layer deep Gaussian process. The difference is that whereas Bayesian neural networks infer the posterior of weight matrices, deep Gaussian processes infer the posterior that of inducing points.

6. APPLICATION OF GAUSSIAN PROCESSES TO SPEECH PROCESSING

In the previous sections, the frameworks of Gaussian process regression and deep Gaussian process have been described. Since the GP-based frameworks are general machine learning ones, they can be applied to speech information processing. For example, in the voice conversion task, Pilkington et al. used the acoustic feature vectors of source and target speakers as input and output features [21]. To reduce computational complexity, they performed the clustering of acoustic features and employed a local GP approximation. Park et al. applied Gaussian process regression to voice activity detection [22]. In this study, the sample of a waveform is predicted from adjacent samples. Since the optimal kernel parameter of voiced regions is different from that of noisy segment, this difference is used to detect voiced regions.

In [23], Gaussian process regression is used for statistical parametric speech synthesis. This technique utilizes the advantage of kernel methods that structured input can be used. As an input of speech synthesis, they use structural frame-level context that consists of phoneme- and phrase-level information combined with frame position information. To alleviate the computational complexity problem, an approximation method based on local GP is used, and the partition of training data is performed using the decision tree used in hidden Markov model (HMM)-based speech synthesis. However, this framework depends on the design of kernel functions for complicated context, and the performance is greatly affected by the construction of decision trees.

DGP-based speech synthesis [24] incorporated a deep architecture that encodes the complicated context represented by hundreds dimensional context vector. It is reported that the DGP-based speech synthesis can generate more natural sounding speech than DNN-based speech synthesis.

Moungsri et al. proposed the duration model for speech synthesis, which utilizes the uncertainties of multiple Gaussian process models [25]. By multiplying the predictive distributions of syllable- and phone-level durations, this method achieved more robust inference than using a single Gaussian process regression.

For application to speech recognition, Lam proposed to insert Gaussian process layer for DNN-based speech recognition [26]. This model uses an approximation method based on random Fourier features.

In this paper, we have explained the Gaussian process regression which predicts the output variables from the input variables. However, as another model of Gaussian process, Gaussian process latent variable model (GPLVM) is proposed, in which the distribution of latent input variables is inferred from observed variables. GPLVM can encode observed vectors into latent features with a small dimensions. Gaussian process dynamic model (GPDM) is an extension of GPLVM which can represent time-series information. GPLVM and GPDM are used to various applications such as music genre recognition [27], phone classification [28], dynamical latent representation of acoustic feature sequence [29], syllable-level stress detection [30], and semi-supervised prosody modeling [31].

7. CONCLUSIONS

In this paper, we introduced the basics of Gaussian process, and approximation techniques and the extension to deep architecture models. Since Gaussian process regression is a general-purpose machine learning technique, it can be applied not only to speech information processing but also in a wide variety of applications. The recent developments for scalable Gaussian process enable us to use a huge amount of data for training. In future work, it is expected to construct deep Gaussian process models easily in the same way as neural network models.

REFERENCES

[1] C. E. Rasmussen and C. K. I. Williams, Gaussian Processes for Machine Learning (The MIT Press, Cambridge, Mass., 2006).

[2] A. G. de G. Matthews, M. Rowland, J. Hron, R. E. Turner and Z. Ghahramani, “Gaussian process behaviour in wide deep
neural networks,” *Proc. ICLR* (2018).

[3] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington and J. Sohl-Dickstein, “Deep neural networks as Gaussian processes,” *Proc. ICLR* (2018).

[4] R. M. Neal, “Priors for infinite networks,” in *Bayesian Learning for Neural Networks*, Lecture Notes in Statistics, Vol. 118 (Springer, New York, 1996).

[5] A. Rahimi and B. Recht, “Random features for large-scale kernel machines,” *Proc. NIPS*, pp. 1177–1184 (2008).

[6] Y. Cho and L. K. Saul, “Kernel methods for deep learning,” *Proc. NIPS*, pp. 342–350 (2009).

[7] C. K. I. Williams, “Computing with infinite networks,” *Neural Comput.*, 10, 1203–1216 (1998).

[8] E. Snelson and Z. Ghahramani, “Local and global sparse Gaussian process approximations,” *Proc. AISTATS*, pp. 524–531 (2007).

[9] J. Hensman, N. Fusi and N. D. Lawrence, “Gaussian processes for big data,” *Proc. UAI*, pp. 282–290 (2013).

[10] J. Hensman, A. Matthews and Z. Ghahramani, “Scalable variational Gaussian process classification,” *Proc. AISTATS*, pp. 1648–1656 (2015).

[11] H. Liu, Y.-S. Ong, X. Shen and J. Cai, “When Gaussian process meets big data: A review of scalable GPs,” *arXiv preprint*, 1807.01065 (2018).

[12] E. Snelson and Z. Ghahramani, “Local and global sparse Gaussian process approximations,” *Proc. AISTATS*, pp. 524–531 (2007).

[13] Y. Cao and D. J. Fleet, “Generalized product of experts for automatic and principled fusion of Gaussian process predictions,” *Proc. Modern Nonparametrics 3: Automating the Learning Pipeline Workshop at NIPS* (2014).

[14] H. Liu, J. Cai, Y. Wang and Y.-S. Ong, “Generalized robust Bayesian committee machine for large-scale Gaussian process regression,” *Proc. ICML*, pp. 3131–3140 (2018).

[15] A. G. Wilson and H. Nickisch, “Kernel interpolation for scalable structured Gaussian processes (KISS-GP),” *Proc. ICML*, pp. 1775–1784 (2015).

[16] P. Izmailov, A. Novikov and D. Kropotov, “Scalable Gaussian processes with billions of inducing inputs via tensor train decomposition,” *Proc. AISTATS*, pp. 726–735 (2018).

[17] J. R. Gardner, G. Pleiss, R. Wu, K. Q. Weinberger and A. G. Wilson, “Product kernel interpolation for scalable Gaussian processes,” *Proc. AISTATS*, pp. 1407–1416 (2018).

[18] M. Lázaro-Gredilla, J. Quironero-Candela, C. E. Rasmussen and A. R. Figueiras-Vidal, “Sparse spectrum Gaussian process regression,” *J. Mach. Learn. Res.*, 11, 1865–1881 (2010).

[19] A. C. Damianou and N. D. Lawrence, “Deep Gaussian processes,” *Proc. AISTATS*, pp. 207–215 (2013).

[20] H. Salimbeni and M. Deisenroth, “ Doubly stochastic variational inference for deep Gaussian processes,” *Proc. NIPS*, pp. 4591–4602 (2017).

[21] N. C. V. Pilkington, H. Zen and M. J. F. Gales, “Gaussian process experts for voice conversion,” *Proc. Interspeech*, pp. 2761–2764 (2011).

[22] S. Park and S. Choi, “Gaussian process regression for voice activity detection and speech enhancement,” *Proc. IJCNN*, pp. 2879–2882 (2008).

[23] T. Koriyama, T. Nose and T. Kobayashi, “Statistical parametric speech synthesis based on Gaussian process regression,” *IEEE J. Sel. Top. Signal Process.*, 8, 173–183 (2014).

[24] T. Koriyama and T. Kobayashi, “Statistical parametric speech synthesis using deep Gaussian processes,” *IEEE/ACM Trans. Audio Speech Lang. Process.*, 27, 948–959 (2019).

[25] D. Moungsri, T. Koriyama and T. Kobayashi, “Duration prediction using multiple Gaussian process experts for GPR-based speech synthesis,” *Proc. ICASSP*, pp. 5495–5498 (2017).

[26] M. W. Y. Lam, S. Hu, X. Xie, S. Liu, J. Yu, R. Su, X. Liu and H. Meng, “Gaussian process neural networks for speech recognition,” *Proc. Interspeech*, pp. 1778–1782 (2018).

[27] K. Markov and T. Matsui, “Music genre and emotion recognition using Gaussian processes,” *IEEE Access*, 2, 688–697 (2014).

[28] H. Park, S. Yun, S. Park, J. Kim and C. D. Yoo, “Phoneme classification using constrained variational Gaussian process dynamical system,” *Proc. NIPS*, pp. 2006–2014 (2012).

[29] G. E. Henter, M. R. Frean and W. B. Kleijn, “Gaussian process dynamical models for nonparametric speech representation and synthesis,” *Proc. ICASSP*, pp. 4505–4508 (2012).

[30] D. Moungsri, T. Koriyama and T. Kobayashi, “Unsupervised stress information labelling using Gaussian process latent variable model for statistical speech synthesis,” *Proc. Interspeech*, pp. 1517–1521 (2016).

[31] T. Koriyama and T. Kobayashi, “Semi-supervised prosody modeling using deep Gaussian process latent variable model,” *Proc. Interspeech*, pp. 4450–4454 (2019).

**APPENDIX: FORMULAS ABOUT GAUSSIAN DISTRIBUTION**

Calculation of Gaussian process regression is attributed by the properties of Gaussian distribution. In this section, we introduce representative formulas about Gaussian distribution. When $p(x)$ is a Gaussian distribution with mean $\mu$ and covariance matrix $\Sigma$, we represent $p(x) = \mathcal{N}(x; \mu, \Sigma)$.

### A.1. Joint and Conditional Distributions

When the joint distribution of $x$ and $y$ is the following multivariate Gaussian distribution,

$$ p\left(\begin{bmatrix} x \\ y \end{bmatrix} \right) = \mathcal{N}\left( \begin{bmatrix} \mu_x \\ \mu_y \\ \Sigma_{xx} \Sigma_{xy} \end{bmatrix} \right) \tag{A-1} $$

the conditional distribution of $y$ given $x$ also becomes a Gaussian distribution expressed by

$$ p(y|x) = \mathcal{N}(y; \mu_{yx} \Sigma_{yx}) \tag{A-2} $$

$$ \mu_{yx} = \mu_y + \Sigma_{yx} \Sigma_{xx}^{-1} \mu_x \tag{A-3} $$

$$ \Sigma_{yx} = \Sigma_{yy} - \Sigma_{yx} \Sigma_{xx}^{-1} \Sigma_{xy} \tag{A-4} $$

### A.2. Linear Transformation on Gaussian Distribution

When the conditional mean of $y$ depends on $x$ as follows:

$$ p(y|x) = \mathcal{N}(y; Ax + b; L) \tag{A-5} $$

$$ p(x) = \mathcal{N}(x; m, S) \tag{A-6} $$

the marginal distribution of $p(y) = \int p(y|x)p(x)dx$ and the conditional distribution of $p(x|y) = p(y|x)p(x)/p(y)$ become Gaussian distributions given by
\begin{align}
p(y) &= \mathcal{N}(Am + b; L + ASA^\top) \tag{A-7} \\
p(x|y) &= \mathcal{N}(x; \mu_{x|y}, \Sigma_{x|y}) \tag{A-8} \\
\mu_{x|y} &= \Sigma_{x|y}(A^\top L(y - b) + S^{-1}m) \tag{A-9} \\
\Sigma_{x|y} &= (S^{-1} + A^\top LA)^{-1}. \tag{A-10}
\end{align}

**A.3. Sum of Variables of Gaussian Distribution**

When the distributions of \(x_1\) and \(x_2\) are Gaussian distributions represented by \(\mathcal{N}(x_1; \mu_1, \Sigma_1)\) and \(\mathcal{N}(x_2; \mu_2, \Sigma_2)\), respectively, the distribution of the sum of \(x_1\) and \(x_2\) is the following Gaussian:

\[
p(x_1 + x_2) = \mathcal{N}(x_1 + x_2; \mu_1 + \mu_2, \Sigma_1 + \Sigma_2). \tag{A-11}
\]