Stochastic Approach for System Identification using Machine Learning

1st Shokhjakhon Abdufattokhov
Computer Engineering
Tashkent University of Information Technologies
named after Muhammad al-Khwarizmi
Amir Temur street, 108, 100200, Tashkent, Uzbekistan
shokhjakhon2010@gmail.com

2nd Behzod Muhiddinov
International Relations
Tashkent University of Information Technologies
named after Muhammad al-Khwarizmi
Amir Temur street, 108, 100200, Tashkent, Uzbekistan
behzodms@gmail.com

***Abstract*—This paper gives a clear understanding of how a system can be modeled in a non-parametric and probabilistic way using machine learning based on historical data with taking into account uncertain noise added to the output of the system. We explained key concepts step by step and end up with a unique algorithm. Moreover, we demonstrated an illustrative example of the proposed algorithm in case of nonlinear dynamic system.**

***Index Terms*—Bayesian analysis, system identification, dynamic system, regression, machine learning, stochastic process, deterministic modeling.**

I. Introduction

Rapid advancements in the ability to train flexible machine learning models, enabled by amassing data and breakthroughs in the understanding of the difficulties behind gradient-based training algorithms,have made the considerably younger field of machine learning explode with interest. However, most deterministic machine learning algorithms suffers from being data inefficient causing difficulties in training process. Moreover, after model is selected, when longterm predictions (or sampling trajectories from this model) leave the training data, the predictions of the function approximator are essentially arbitrary, but they are claimed with "full confidence". To overcome the problem, constructing a model based on a suitable machine learning algorithms which builds system's model using probabilistic function approximator that places a posterior distribution over the transition function and expresses the level of uncertainty about the model [1], can be an alternative and efficient solution. Hence, for learning from scratch, we first require a probabilistic model to express model uncertainty. We can employ non-parametric probabilistic Gaussian processes (GPs) for this purpose.

II. Gaussian Processes

Definition 2.1 A Gaussian processes is a collection of random variables, which have a joint Gaussian distribution.

Figure 1 shows some examples of joint Gaussian probability densities.

The Gaussian Process (GP) model can be regarded as a nonparametric method of nonlinear system identification where new predictions of system behaviour are computed through the use of Bayesian inference techniques applied to empirical data [2]. The GP model may also be considered as a black-box method as the identification process relies heavily upon experimental data. However, potentially useful attribute of the GP model is the possibility to include various kinds of prior knowledge into the model,, see e.g. [3] for the incorporation of local models and the static characteristic. GP models are closely related to approaches such as Support Vector Machines and specially Relevance Vector Machines [4], [5].

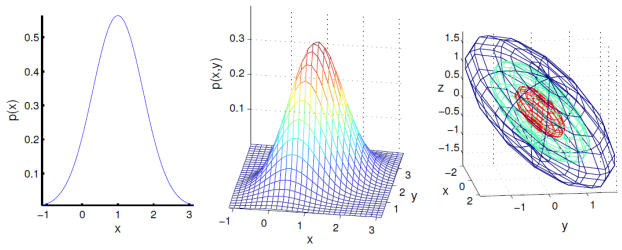

Fig. 1. A Gaussian distribution in 1D (left), 2D (middle), and 3D (right). The first two plots show the probability density function over the input space, the third plot shows contours of iso-probabilities, where red indicates a contour of high probability.

A GPs is completely specified by its mean function and covariance function. We define mean function $m_f(x)$ and the covariance function $C_f(x_i, x_j)$ of a real process $f(x)$ as

$$m_f(x_i) = E[f(x_i)] \tag{1}$$

$$C_f(x_i, x_j) = E[(f(x_i) - m_f(x_i))(f(x_j) - m_f(x_j))]. \tag{2}$$

Any finite set of values from random function $f(X) = [f(x_1), \ldots, f(x_n)]^T$ is jointly Gaussian distributed and can be written as

$$P(f(X)) = N(m_f, K_f). \tag{3}$$

The function that is GP is denoted as

$$f(X) \sim GP(m_f, K_f). \tag{4}$$

If we do not have prior information about the mean function of the GP model, we can specify as $m_f \equiv 0$.

A covariance matrix K_f is evaluated from the covariance function $C_f(x_i, x_j)$ applied to all the pairs i and j of measured

data. The elements K_f^{ij} of the covariance matrix K_f are covariances between the values of the functions $f(x_i)$ and $f(x_j)$ corresponding to the arguments x_i and x_j :

$$K_f^{ij} = cov(f(x_i), f(x_j)) = C_f(x_i, x_j). \tag{5}$$

Any function $C_f(x_i, x_j)$ can be a covariance function, that generates a positive, semi-definite, covariance matrix K_f. The covariance function $C_f(x_i, x_j)$ can be thought as a correlation between the function values $f(x_i)$ and $f(x_j)$.

III. A PROBABILISTIC PREDICTIVE MODELING

A. Prior GP model

Consider the system

$$y = f(x) + \epsilon \tag{6}$$

with the white Gaussian noise $\epsilon \sim N(0, \sigma_n^2)$, with the variance σ_n^2 and the vector of regressors x from the operating space R^D. We have $[y_1, \ldots, y_n]^T \sim N(0, K)$ with

$$K = K_f + \sigma_n^2 I \tag{7}$$

where K_f is the covariance matrix for the noise-free f of the system and I is the $n \times n$ identity matrix. The elements of K_f are defined as in eq. 5.
In general, we define our covariance function with noisy system as follows

$$C(x_i, x_j) = C_f(x_i, x_j) + C_n(x_i, x_j). \tag{8}$$

where C_f represents the functional part that describes the unknown system of our interest and C_n represents the noise part that tells about the measurement noise added to the model. For the noise part in eq. 8, it is most common to use a white Gaussian noise. Just to have an idea, we consider the composite covariance function composed of the squared exponential covariance function for the functional part and the constant covariance for the noise :

$$C(x_i, x_j) = \sigma_f^2 exp\left[-\frac{1}{2}\sum_{d=1}^{D}\theta_d(x_i^d - x_j^d)\right] + \sigma_n^2\delta_{ij}. \tag{9}$$

where θ_d, σ_f and σ_n are the hyperparameters of the covariance function, D is the input dimension, δ_{ij} is the Kronecker delta function and x_i^d stands for d^{th} feature of input x_i. The name hyperparameters originates from the kernel methods, because a covariance function is a kernel function. One can refer to [6] for various type of covariance function.

B. Posterior GP model

After the data for modelling is collected $D = \{(x_i, y_i)|i = 1, \ldots, n\} = \{(X, Y)\}$, and prior belief in the form of the mean function and the covariance function is selected, then the posterior GP model is inferred. The posterior distribution over f, we are intersted in, for the given data D and hyperparameters θ is

$$P(f|X, Y, \theta) = \frac{P(Y|f, X, \theta) \times P(f|\theta)}{P(Y|X, \theta)}. \tag{10}$$

where $P(Y|f, X, \theta)$ is the likelihood, $P(f|\theta)$ is the function f prior for the given hyperparameters θ, $P(Y|X, \theta)$ is the evidence and $P(f|X, Y, \theta)$ is the posterior distribution over.

The prior distribution over the function f is set as a GP model:

$$f \sim GP(m_f, K) \tag{11}$$

Then the posterior GP model looks as

$$\mu(x^\star) = E[f(x^\star)|X, Y, \theta]. \tag{12}$$

$$\sigma^2(x^\star) = var[f(x^\star)|X, Y, \theta]. \tag{13}$$

where $x^\star$ is a test input data.

C. Prediction

After the model is identified, the next task is to predict a new output estimate $y^\star$ of the GP model for a given $x^\star$. Prediction problem in the Bayesian framework means to predict probability distribution $P(y^\star|D, x^\star)$ for the posterior $P(f(x^\star))$. But it is also possible to get the predictive distribution of $y^\star$ from the application of conditional probability:

$$P(y^\star|D, K, x^\star) = \frac{P(Y_{n+1}^T|K, X, x^\star)}{P(Y|K, X)}. \tag{14}$$

where $Y_{n+1} = [Y^T, y^\star]$ and $Y = [y_1, \ldots, y_n]^T$ is an $n \times 1$ vector of training targets. Using Gaussian probability distribution function for random vector from [6](Appendix A1) the prediction probability density function is

$$P(Y_{n+1}^T|K, X, x^\star) = \frac{1}{(2\pi)^{\frac{n+1}{2}}\det(K_{n+1})^{\frac{1}{2}}} * exp\left[-\frac{1}{2}(Y_{n+1}K_{n+1}^{-1}Y_{n+1}^T)\right]. \tag{15}$$

The following step is to find how a new input is inserted to the covariance matrix K_{n+1}. For the collection of random variables $[y_1, \ldots, y_n, y^\star]$ we define :

$$Y_{n+1} \sim N(0, K_{n+1}). \tag{16}$$

with the covariance matrix

$$K_{n+1} = \begin{pmatrix} K & K_\star \\ K_\star^T & K_{\star\star} \end{pmatrix} \tag{17}$$

where

$K_\star = [C(x_1, x^\star), \ldots, C(x_n, x^\star)]^T$ is the $n \times 1$ vector of covariances between the training and the test input data,

$K_{\star\star} = C(x^\star, x^\star)$ is the autocovariance submatrix of the test input data.

Then, we write the prediction probability density as

$$P(y^\star|D, K, x^\star) = \frac{\det(K)^{\frac{1}{2}}}{(2\pi)^{\frac{1}{2}}\det(K_{n+1})^{\frac{1}{2}}} * exp\left[-\frac{1}{2}(Y_{n+1}K_{n+1}^{-1}Y_{n+1}^T - Y^TK^{-1}Y)\right] \tag{18}$$

Finally, the prediction probability distribution from eq. 18 is Gaussian with the following mean and variance :

$$E[y^\star] = \mu(x^\star) = K_\star^T K^{-1} Y. \tag{19}$$

$$var[y^\star] = \sigma^2(x^\star) = K_{\star\star} - K_\star^T K^{-1} K_\star \tag{20}$$

The vector $K_\star^T K^{-1}$ in eq. 19 is a smoothing term that weights Y to make a prediction at $x^\star$. Look at Figure 2 to see the demonstration of Gaussian prediction together with 95% confidence interval in grey color and how noise affects to the prediction.

Note: in the equations above, we were assuming that $m_f \equiv 0$ and will stay in this assumption in the following chapters too.

If we consider nonzero mean function, then eq. 19 and eq. 20 are written as

$$E[y^\star] = \mu(x^\star) = m_f(x^\star) + K_\star^T K^{-1}(Y - m_f(X)). \tag{21}$$

$$var[y^\star] = \sigma^2(x^\star) = K_{\star\star} - K_\star^T K^{-1} K_\star \tag{22}$$

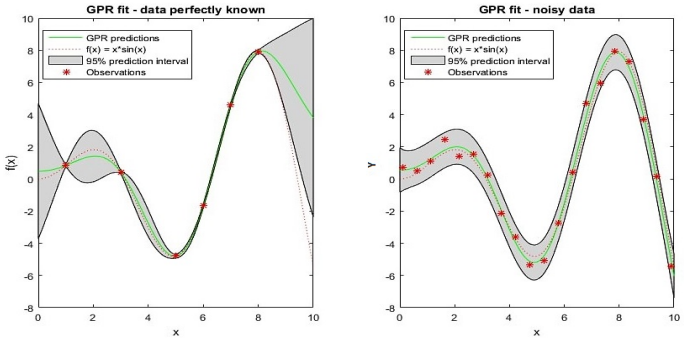

Fig. 2. Noise free data prediction(left) and Noisy data prediction(right)

IV. Model Selection

A. Optimisation: Marginal Likelihood (Evidence Maximization)

$$P(Y|X,\theta) = \frac{1}{(2\pi)^{\frac{n}{2}} \det(K)^{\frac{1}{2}}} exp\left[-\frac{1}{2}(Y^T K^{-1} Y)\right]. \tag{23}$$

with the $n \times n$ covariance matrix K of the training data.

Maximum likelihood method $P(Y|X,\theta)$ is based on the presumption that the most likely values of the hyperparameters $P(\theta|X,Y)$ are noticeably more likely than other values. This means that by looking for hyperparameters that maximise the probability of the training data, we are optimising the properties of the Gaussian process prior that is to be used to generate new predictive distributions.

The values of the hyperparameters depend on the collected data and it is difficult to select their prior distribution. If a uniform prior distribution is selected, then the hyperparameters' posterior distribution $P(\theta|X,Y)$ is proportional to the marginal likelihood

$$P(\theta|X,Y) \propto P(Y|X,\theta) \tag{24}$$

which states that the maximum a posteriori estimate of the hyperparameters θ equals the maximum marginal likelihood estimate.

Because of monotic property of the logarithm function for the numerical scaling purposes, the evidence logarithm is used as the objective function for the optimisation (minimization):

$$-l(\theta) = \frac{n}{2}\ln 2\pi + \frac{1}{2}\ln(detK) + \frac{1}{2}Y^T K^{-1} Y \tag{25}$$

where $l(\theta) = P(Y|X,\theta)$.

Minimizing $-l(\theta)$ using eq. 25 is equivalent to maximizing the evidence. One can use a conjugate gradient optimization method to solve following optimization problem

$$-\nabla l(\theta) = \frac{1}{2} trace\left(K^{-1}\frac{\partial K}{\partial \theta_i}\right) - \frac{1}{2} Y^T K^{-1} \frac{\partial K}{\partial \theta_i} K^{-1} Y \tag{26}$$

where $\nabla l(\theta)$ uses the Cholesky decomposition to invert $n \times n$ matrix K, see Appendix in [7]. However, there are exist alternative optimization methods given with full details in [6].

B. Model Validation

The model validation is one of the crucial part of the model selection, after the regressors, covariance function, mean function and hyperparameters are selected. Usually, at the beginning of the training process, available data is separated into identification and test sets. The former set is used for train the model and the latter set is used to evaluate the performance of the model. Below commonly used error metrics are given. For more choices one may refer to [7].

SMSE(Standardized Mean Square Error) - the measure that normalises the mean-squared error between the mean of the model's output and the measured output of the process by the variance of the output values of the training dataset

$$SMSE = \frac{1}{n}\frac{\sum_{i=1}^n (y_i - E[\hat{y}])^2}{\sigma_y^2} \tag{27}$$

NRMSE(Normalized Root Mean Square Error) - the measure that normalises the root mean-squared error between the mean of the model's output and the measured output of the process by the maximum difference of the output values of the training dataset

$$NRMSE = \sqrt{\frac{1}{n}\frac{\sum_{i=1}^n (y_i - E[\hat{y}])^2}{(y_{max} - y_{min})^2}} \tag{28}$$

If we sum up all information above, we end up with following pseudoalgorithm:

Algorithm for GP model selection

1) **input** training data D and test data $D^{\star}$
2) **repeat**
   - choose regressors from D
   - choose prior mean and covariance functions
   - **repeat**
     define:
     – prior hyperparameters θ
     compute:
     – covariance matrix K
     – find K^{-1}
     – $\nabla l(\theta)$
   - **until** $-l(\theta)$ is minimal.
   - **simulate(predict) the model for the test data**
     compute for each new input $x^{\star}$ from $X^{\star}$:
     – mean $\mu(x^{\star})$
     – variance $\sigma^2(x^{\star})$
3) **until** validatation error is minimal.

V. Demonstration with an example

The following example illustrates the use of described method in Matlab with toolbox functions created by authors in [8]. Consider the following discrete nonlinear system approximated by 4^{th} order Runga-Kutta method with $h = 0.5$:

$$y(k+1) = y(k) + h/6(t_1 + t_2 + t_3 + t_4) \quad (29)$$

$$t_1 = -tanh(y(k) + u(k)) \quad (30)$$

$$t_2 = -tanh(y(k) + u^3(k) + ht_1) \quad (31)$$

$$t_3 = -tanh(y(k) + u^3(k) + ht_2) \quad (32)$$

$$t_4 = -tanh(y(k) + u^3(k) + ht_3) \quad (33)$$

The output of the model is state y, disturbed with Gaussian white noise with variance 0.0045. Signal u was kept constant for 4 time instants and was generated by a random number generator with normal distribution in the magnitude between -1 and 1 with number of samples $N = 600$. We would like to obtain a Gaussian process model for the discrete-time system (29) following step by step the proposed algorithm in the previous section. Using 60% of the generated data set N(the rest is used for testing), the discrete-time system (29) is approximated with Gaussian Process with zero mean and the covariance function of the form (9). Models of various orders were fitted, but the optimization (25) found the secod order model as the most appropriate with metrics $SMSE = 0.0021553$ and $NRMSE = 0.011195$. The response of the Gaussian process model to the validation signal is shown in Fig.3 in the form of prediction means together with the 95% confidence band.

VI. Conclusion

This paper presented a stochastic approach for system identification using Gaussian Processes to build a model with probabilistic guarantee constructed directly from historical observations of a system. The possibility to include information about the probabilistic trust region in the model is one of

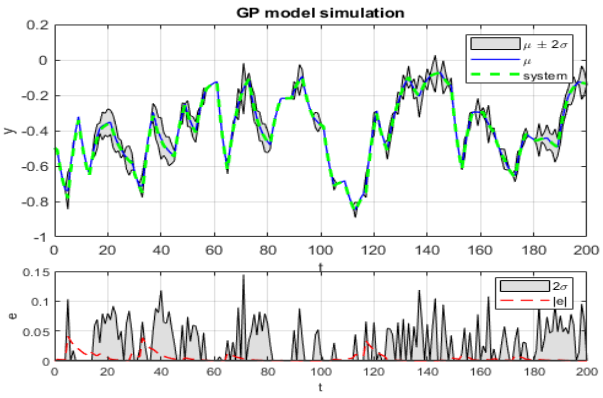

Fig. 3. Response of the Gaussian process model to the validation signal. Upper part plots the true values, the predicted mean and 95 % confidence intervals (2σ), below part shows the absolute residuals.

powerful feature of Gaussian Processes models that making them to be spread widely in many applications. However, trust region depends on data used for model identification and the computational complexity becomes hard as the size of data increases. Therefore, obtaining the best GP model with the least amount data that best explains the system behavior will be a new challenge in our future work.

References

[1] Galiana, F., Handschin, E., and Fiechter, A., "Identification of stochastic electric load models from physical data," Automatic Control, IEEE Transactions 19(6), 1974, pp. 887–893.

[2] Thompson, K.: "Implementation of Gaussian process models for non-linear system identification", PhD thesis, University of Glasgow, Glasgow, 2009.

[3] Solak, E., R. Murray-Smith, W.E. Leithead, D.J. Leith, and C.E. Rassmusen, "Derivative observations in Gaussian Process Models of Dynamic Systems". In: Becker, S., S. Thrun, K. Obermayer (Eds.), "Advances in Neural Information Processing Systems", MIT Press, 2003, pp. 529-536.

[4] Quinonero-Candela, J., C.E. Rasmussen. "Analysis of Some Methods for Reduced Rank Gaussian Process Regression". In: Murray-Smith, R. and R. Shorten (Eds.), "Switching and Learning in Feedback Systems", Lecture Notes in Computer Science, Vol.3355, 2005.

[5] Dong, B., Cao, C., and Lee, S. E., "Applying support vector machines to predict building energy consumption in tropical region," Energy and Buildings 37(5), 2005, pp. 545–553.

[6] Rasmussen, C. E. and Williams, C. K. I., "Gaussian processes for machine learning", Vol.1, MIT press, Cambridge, MA, 2006.

[7] Jus Kocijan., "Modelling and control of dynamic systems using Gaussian process models", Springer, 2016.

[8] Jus Kocijan, Kristjan Azman, Alexandra Grancharova, "The Concept for Gaussian Process Model Based System Identification Toolbox", International Conference on Computer Systems and Technologies - CompSysTech 07, 2006.