The Journal of Systems and Software 000 (2016) 1–7

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Recursive prediction algorithm for non-stationary Gaussian Process

Yulai Zhang[a,*], Guiming Luo[b]

[a] School of Information and Electronics Engineering, Zhejiang University of Science and Technology, Hangzhou, 310023, China
[b] School of Software, Tsinghua University, Beijing, 100084, China

ARTICLE INFO

Article history:
Received 29 September 2015
Revised 16 July 2016
Accepted 7 August 2016
Available online xxx

ABSTRACT

Gaussian Process is a theoretically rigorous model for prediction problems. One of the deficiencies of this model is that its original exact inference algorithm is computationally intractable. Therefore, its applications are limited in the field of real-time online predictions. In this paper, a recursive prediction algorithm based on the Gaussian Process model is proposed. In recursive algorithms, the computational time of the next step can be greatly reduced by utilizing the intermediate results of the current step. The proposed recursive algorithm accelerates the prediction and avoids the loss of accuracy at the same time. Experiments are done on an ultra-short term electric load data set and the results are demonstrated to show the accuracy and efficiency of the new algorithm.

1. Introduction

The Gaussian Process (GP) model has been widely investigated in the past decade in machine learning (Rasmussen and Williams, 2006; Rasmussen and Nickisch, 2010). It has also been adopted in many application studies to solve the problems of real-time data prediction for complex dynamic systems such as power load forecast (Zhang et al., 2011; Niu et al., 2010) and ozone concentration forecast (Petelin et al., 2013). Though it is a theoretically rigorous model, the applications in industry are limited due to the high computational complexity of its inference method, especially when the size of the training data set is increasingly large.

Many approximate inference algorithms for the Gaussian Process model are proposed in the recent years to solve this problem. These approximate methods reduce the running time at the cost of lower predict accuracy. Some of the approximate algorithms re-sample the data sets and eliminate those less informative data points (Cao et al., 2013). And some other algorithms use the sparse techniques to reduce the rank of the covariance matrix (Gittens and Mahoney, 2013). All of these approximate algorithms discard the less informative elements in the Gaussian Process model to simplify the computation. However, how to determine the less informative elements itself is time consuming. On the other hand, some of these methods require fixed data sets thus cannot perform well on the task of online prediction for dynamic systems.

Another way to speed up the computation for large data sets is to use the recursive methods. Recursive algorithms are widely used in real-time prediction problems. The most basic recursive algorithm is the well-known Recursive Least Square (RLS) method (Ljung, 1998). The current step will be calculated by using the intermediate results of the previous step or steps in the recursive algorithms. Similarly, in the proposed algorithm in this paper, most of the computational intensive operations on the increasingly large data matrices will be avoided by updating them from the results of the previous steps.

However, building the recursive inference algorithm for general Gaussian Process model is not an easy task. Because there are various choices for the covariance function which will decide the major properties of a specific GP model. Since non-stationary outputs are quite common in dynamic systems, we will focus on the GP model with the linear trend covariance function which is effective for the non-stationary problems in this paper. In addition, one of the initial motivations of this work is to solve the prediction problems for the ultra-short term power load series, which is also typically non-stationary.

The basics of the Gaussian Process model and its online prediction problem will be introduced and formulated in Section 2. The recursive GP inference algorithm will be developed in Section 3. Related works are discussed in Section 4. And the Experiments will be presented in Section 5. The RNSGP algorithm proposed in this paper can be taken as the recursive improvement of the FNSGP algorithm appears in our conference paper (Zhang and Luo, 2014a).

* Corresponding author.
E-mail addresses: zhangyulai@zust.edu.cn (Y. Zhang), gluo@tsinghua.edu.cn (G. Luo).

http://dx.doi.org/10.1016/j.jss.2016.08.036
0164-1212/

2. Preliminaries

2.1. Gaussian Process

The Gaussian Process (GP) method constructs a model:

$$y_t = f(x_t) + e_t \tag{1}$$

from the data set $\{(x_t, y_t)|t = 1, \ldots, n\}$. In time series problems, the feature vector x_t is composed of the history data, $x_t = [y_{t-1}, \ldots, y_{t-d}]$, where d is a manually selected constant. y_t is the corresponding scalar output. e_t is additive white noise with the Gaussian distribution $e(t) \sim N(0, \sigma_n^2)$.

For a new input vector x^*, the joint distribution of the new output y^* and the history data $Y = [y_1, y_2, \ldots, y_n]^T$ is

$$\begin{bmatrix} Y \\ y^* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(X,X) + \sigma_n^2 I & \mathbf{k}(x^*, X) \\ \mathbf{k}^T(x^*, X) & k(x^*, x^*) \end{bmatrix}\right) \tag{2}$$

where $k(\cdot, \cdot)$ is the covariance function. Matrix K is called the Gram matrix whose elements are $K_{ij} = k(x_i, x_j)$, and $X = [x_1, x_2, \ldots, x_N]^T$, $\mathbf{k}(x^*, X) = [k(x^*, x_1), k(x^*, x_2), \ldots, k(x^*, x_N)]^T$.

If the noises are not white, we can substitute the identity matrix I with the auto-covariance matrix of the noise e_t. Discussions of the Gaussian Process model with ARMA noises can be found in Murray-Smith and Girard (2001).

The expected value and the variance of y^* can be obtained by the joint Gaussian distribution as:

$$\hat{y}^* = \mathbf{k}^T(x^*, X)(K(X,X) + \sigma_n^2 I)^{-1} Y \tag{3}$$

$$Var(y^*) = k(x^*, x^*) - \mathbf{k}^T(x^*, X)(K(X,X) + \sigma_n^2 I)^{-1}\mathbf{k}(x^*, X) \tag{4}$$

Only the one-step-ahead prediction problems will be focused in this paper. In the s-steps-ahead prediction problems where $s > 1$, the feature vectors can be written as:

$$x_t = [y_{t-s}, \ldots, y_{t-d-s+1}]^T.$$

Alternatively, if the predicted values are used in the feature vector in multiple-steps-ahead prediction problems, the issue of uncertainty propagation should be concerned (Girard et al., 2003).

2.2. Non-stationary covariance function

One can chose a variety of covariance functions to fit the data generated from systems with diverse properties. Covariance function is the most important component in the GP model and the prediction results can be significantly affected by this choice.

The most frequently used covariance function is the Squared Exponential covariance function:

$$k_{se} = e^{-\|x_i - x_j\|^2}$$

It models the covariance value between two data points by their Euclidean distance in the feature space with the negative exponential function. It works well for stationary sequences but fails for the non-stationary data. Many real-time prediction problems have non-stationary outputs. The Linear Trend covariance functions (Brahim-Belhouari and Bermak, 2004) are always used in the Gaussian Process models for non-stationary time series:

$$k_{lt}(x_i, x_j) = {x_i}^T x_j \tag{5}$$

where $x_i^T x_j$ is the inner product of the feature vectors x_i and x_j. Similarly, the weighted version of the Linear Trend covariance function can be written as:

$$k_{wlt}(x_i, x_j) = {x_i}^T L x_j \tag{6}$$

where L is a $d \times d$ diagonal matrix whose diagonal entries are $l_1, \ldots, l_d$. Note that the corresponding features with larger weights will have bigger contributions to the final prediction results.

The diagonal elements of the matrix L are also called the hyper-parameters of the Gaussian Process model, and they can be learned from the training data by optimizing a most likelihood problem. This procedure has also been named as the Automatic Relevance Determination (ARD). The hyper-parameter vector can be written in together with the noise variance as:

$$\theta = [l_1, \ldots, l_d, \sigma_n^2]$$

If X_n, Y_n are the training data for the hyper-parameters estimation, the log likelihood function can be written as:

$$\log p(Y_n|X_n, \theta) = -\frac{1}{2} Y_n^T (K_n + \sigma_n^2 I) Y_n - \frac{1}{2}\log|K_n + \sigma_n^2 I| - \frac{n}{2}\log 2\pi \tag{7}$$

The optimal estimation can be obtained by:

$$\hat{\theta} = \arg\max_{\theta}(\log p(Y_n|X_n, \theta)) \tag{8}$$

In the above equations, the Gram matrix $K(X_n, X_n)$ are written as K_n for simplicity, and the variance of the additive noise σ_n^2 is also estimated in together with the hyper-parameters. Iterative methods, such as conjugate gradient, can be used to solve the most likelihood estimation (MLE) problem in (8).

Note that the Gaussian Process model is a non-parametric model, so the hyper-parameter learning step is equivalent to the model selection step in those parametric models. And the prediction inference step in the non-parametric GP model, which will be discussed in the next subsection, corresponds to another two steps in the parametric models, namely the parameters estimation step and the prediction inference step.

3. Recursive algorithm

Computational time complexity is an important issue in the real-time online forecast problems. The predictions should be calculated in a limited time which is significantly shorter than the sample interval. In addition, the number of training data n increases with time in most online problems. One solution is to abandon the old data points by using a moving window. But more often, we want to keep all the data points for future predictions. The time cost of the standard inference method for the Gaussian Process model (Rasmussen and Williams, 2006) increases rapidly with the number of data points ($o(n^3)$). Constructing a recursive algorithm is an effective way to solve this problem.

The new algorithm is presented in Table 1 at the end of this section. In order to give the readers a more clear perspective, we divided the algorithm construction into three parts. Part 1 in Section 3.1 reformulate the original calculation by matrix inversion lemma, which corresponds to line 7–9 and 11–12 in Table 1. Part 3 in Section 3.3 is the recursive update parts, which corresponds to line 1–5 in Table 1. Part 2 in Section 3.2 serves as a connection between part 1 and 2. And it corresponds to algorithm line 6 and 10. We also summarize the derivations in part 1 as Lemma 1 and Lemma 2, the update procedure in part 3 as Lemma 4 and 5, and the matrix decomposition in part 2 as Lemma 3. The above lemmas are finally gathered into Theorem 1 in Section 3.4.

In this paper, we focus on the prediction of y_{n+1}, the output at the $(n+1)$th step, from the results of the nth step. First, we update the feature vector of the new step, x_{n+1}, by:

$$x_{n+1} = [y_n, y_{n-1}, \ldots, y_{n-d+1}]^T$$

The data matrices X_n and Y_n can also be updated as:

$$Y_n = [y_n \; Y_{n-1}^T]^T \tag{9}$$

$$X_n = [x_n \; X_{n-1}^T]^T \tag{10}$$

Please cite this article as: Y. Zhang, G. Luo, Recursive prediction algorithm for non-stationary Gaussian Process, The Journal of Systems and Software (2016), http://dx.doi.org/10.1016/j.jss.2016.08.036