Fast Algorithm for Non-Stationary Gaussian Process Prediction *

Yulai Zhang and **Guiming Luo**
School of Software, Tsinghua University
Beijing, 100084, China P.R.
{zhangyl08@mails, gluo@mail}.tsinghua.edu.cn

Abstract

The FNSGP algorithm for Gaussian process model is proposed in this paper. It reduces the time cost to accelerate the task of non-stationary time series prediction without loss of accuracy. Some experiments are verified on the real world power load data.

Introduction

Gaussian process for machine learning (Rasmussen and Williams 2006) is a practical method for the time series prediction problems. One of the deficiencies of the original exact inference method of GP is the high time complexity $o(n^3)$. In order to solve this problem, many approximate inference methods have been developed (Rasmussen and Nickisch 2010). In this paper, a new exact inference method for GP models with non-stationary covariance function is proposed. Real world experiments showed that this algorithm greatly reduces the computational time for online prediction of non-stationary time series.

Process Models

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

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

$$\begin{bmatrix} Y \\ y^* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(X,X) + \sigma_n^2 I & \bar{k}(x^*, X) \\ \bar{k}^T(x^*, X) & k(x^*, x^*) \end{bmatrix}\right) \quad (1)$$

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

*This work was supported by the Funds NSFC61171121 and the Science Foundation of Chinese Ministry of Education - China Mobile 2012.

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

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

$$Var(y^*) = k(x^*, x^*) - \bar{k}^T(x^*, X)(K(X,X) + \sigma_n^2 I)^{-1} \bar{k}(x^*, X) \quad (3)$$

We only consider one step ahead prediction in this paper. In multi-step problems, if the predicted values in the previous steps are used in the feature vectors, the uncertainty will propagate (Girard et al. 2003).

The weighted linear trend covariance function (Brahim-Belhouari and Bermak 2004) is fit for the non-stationary time series prediction tasks:

$$k_{ns}(x_i, x_j) = {x_i}^T T x_j \quad (4)$$

where T is a $d \times d$ diagonal matrix whose diagonal elements are called hyper parameters and can be determined by solving an optimization problem in the GP learning phase before the inference step.

Fast algorithm

The time complexity of the standard Gaussian Process inference algorithm mainly lies in the matrix inverse operation in equation (2) and (3). In addition, the inverse matrix may be numerically singular when the value of n is large. These are the two major problems encountered in constructing the fast algorithm.

In the non-stationary time series problems , when the covariance function (4) is used, the the inverse operation in Equation (2) and (3) can be written as:

$$(K(X,X) + \sigma_n^2 I)^{-1} = (XTX^T + \sigma_n^2 I)^{-1} = \frac{1}{\sigma_n^2} I - \frac{1}{\sigma_n^2} X(\sigma_n^2 T^{-1} + X^T X)^{-1} X \quad (5)$$

by using the matrix inversion lemma:

$$(P + QRS)^{-1} = P^{-1} - P^{-1}Q(SP^{-1}Q + R^{-1})^{-1} SP^{-1}$$

In (5), T is a diagonal matrix, so T^{-1} can be easily calculated with $o(d)$ time cost. The other nontrivial matrix inverse operation is on the $d \times d$ matrix $\sigma_n^2 T^{-1} + X^T X$. d is a constant and is equivalent to the size of the hyper parameter

vector. In most regular statistical learning problems, the size of the parameter vector is much smaller than the number of data samples. Thus we have $d \ll n$, and the matrix inversion will be done on a $d \times d$ matrix, rather than an $n \times n$ one. The time cost of (2) and (3) can be reduced significantly.

Next, we will solve the singular matrix problem. Because the elements of the matrix $(\sigma_n^2 T^{-1} + X^T X)^{-1}$ will be very small when the size of the matrix X increases with n, the inversion matrix may be singular and run into serious round off problems when n is quite large. We calculate the intermediate variables K_y and K_v instead to avoid this problem. Let

$$K_y = (\sigma_n^2 T^{-1} + X^T X)^{-1} X^T Y \quad (6)$$

$$K_v = (\sigma_n^2 T^{-1} + X^T X)^{-1} X^T \bar{k} \quad (7)$$

where K_y is for the calculation of the expected value of the predicted output and K_v is for the corresponding variance value.

The Cholesky decomposition can be used to solve the matrix inversion in (6) and (7). Let $U = \sigma_n^2 T^{-1} + X^T X$, $b = X^T Y$, and $b' = X^T \bar{k}$. Let V be the cholesky decomposition matrix of U, denoted as $V = cholesky(U)$. V is an upper triangle matrix, and $U = VV^T$. Then $K_y = U^{-1} b$ and $K_v = U^{-1} b'$ can be calculated by:

$$U^{-1} b = V^{-T} V^{-1} b, \;\; U^{-1} b' = V^{-T} V^{-1} b' \quad (8)$$

The time costs of the cholesky decomposition and (8) are $o(d^3/6)$ and $o(d^2)$ respectively. Note that the computational complexity of the matrix inversion operation for an upper triangle matrix is much lower than that of the regular one. Let a_y and a_v be:

$$a_y = \frac{1}{\sigma_n^2}\{Y - XK_y\}, \;\; a_v = \frac{1}{\sigma_n^2}\{\bar{k} - XK_v\}. \quad (9)$$

The predicted value and the variance can be finally obtained by $y^* = \bar{k} a_y$ and $\mathrm{var}(y^*) = x^* T x^{*T} - \bar{k} a_v$.

The algorithm is summarized as FNSGP (Fast Non-Stationary Gaussian Process Inference) in the Figure 1.

The time cost of FNSGP is $o(n^2 + d^3)$. Since $d \ll n$ in regular problems, the total time cost of one inference step is reduced from $o(n^3)$ to $o(n^2)$.

Experiments

Our experiment is done on real world ultra short time electric power load data. Realtime online power load prediction have great economic importance in the power industry (Blum and Riedmiller 2013). We choose $n = 1000$, $d = 30$, and the results on predict error (normalized mean square error) and running time are averaged over 5000 steps. For non-stationary time series shown in fig.2, the new method obtains the same predicted values as the standard GP, whereas the time cost is greatly reduced.

References

Blum, M., and Riedmiller, M. 2013. Electricity demand forecasting using gaussian processes. *Power* 10:104.

INPUT: X, Y, T, σ_n^2, x^*
OUTPUT: $y^*, \mathrm{var}(y^*)$

$U = \sigma_n^2 T^{-1} + X^T X$
$b = X^T Y$
$\bar{k} = XTx^{*T}$
$b' = X^T \bar{k}$
$V = cholesky(U)$
$K_y = V^{-T} V^{-1} b$
$K_v = V^{-T} V^{-1} b'$
$a_y = (Y - XK_y)/\sigma_n^2$
$a_v = (\bar{k} - XK_v)/\sigma_n^2$
$y^* = \bar{k} a_y$
$\mathrm{var}(y^*) = x^* T x^{*T} - \bar{k} a_v$

Figure 1: FNSGP (Fast Non-Stationary Gaussian Process Inference)

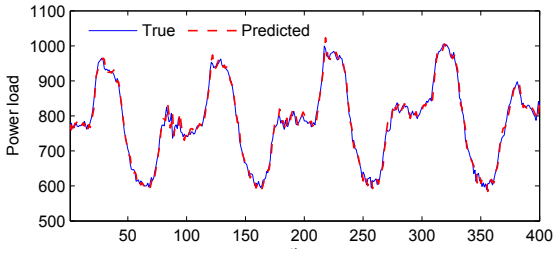

Figure 2: Predicted power load series

Table 1: Comparison of the prediction errors and time cost of FNSGP and Standard GP.

Method	Predict Error (NMSE)	Average Running time
FNSGP	0.005	0.008s
Standard GP	0.005	0.651s

Brahim-Belhouari, S., and Bermak, A. 2004. Gaussian process for nonstationary time series prediction. *Computational Statistics & Data Analysis* 47(4):705–712.

Girard, A.; Candela, J. Q.; Murray-smith, R.; and Rasmussen, C. E. 2003. Gaussian process priors with uncertain inputs–application to multiple-step ahead time series forecasting. In *Advances in Neural Information Processing Systems*, 529–536.

Rasmussen, C. E., and Nickisch, H. 2010. Gaussian processes for machine learning (gpml) toolbox. *The Journal of Machine Learning Research* 9999:3011–3015.

Rasmussen, C. E., and Williams, C. K. I. 2006. Gaussian processes for machine learning. *Adaptive computation and machine learning*.