Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Power Load Forecasting based on Multi-task Gaussian Process

Yulai Zhang * **Guiming Luo** * **Fuan Pu** *

* *Tsinghua National Laboratory for Information Science and Technology, School of Software, Tsinghua University, Beijing, 100084, China, (email:{zhangyl08@mails, gluo@mail, pfa12@mails}.tsinghua.edu.cn).*

Abstract: The power load data from nearby cities are significantly correlated because they share the same hidden variables as well as the underlying noises. A multi-task Gaussian process method for non-stationary time series prediction is introduced and applied to the power load forecasting problem in this paper. The prediction accuracies are effectively improved due to the additional information provided by the related data sets. A novel algorithm for prediction is developed to reduce the computational complexity of the multi-task Gaussian Process method. The algorithm's prediction precision and efficiency are validated by a real world short-term power load data sets.

1. INTRODUCTION

Gaussian process is an important tool for regression and classification problems (Rasmussen and Williams [2006]) in many fields. One of the most basic applications for regression is predicting the unknown output for the given new input. In time series prediction problems, history outputs are always used as the input vectors, which is different from that of the regular problems. In the past decades, a lot of progresses have been made on the topic of time series prediction using Gaussian process. Gaussian Process based algorithms for multiple step prediction (Girard et al. [2003]), non-stationary time series prediction (Brahim-Belhouari and Bermak [2004]), and non-linear time series prediction (Wang et al. [2005]) have been proposed in succession.

Multi-task learning, or transfer learning (Pan and Yang [2010]), is a hot research topic in the recent years. Gaussian process model is a competitive candidate for multi-task learning because the concept of covariance function in Gaussian Process, which is originally adopted to describe the correlations between the data points in single time series, can be easily used to define the correlations between data points from different tasks (Bonilla et al. [2008]).

On the other hand, the Gaussian process methods have also been adopted by the research communities in electric power industry (Chen et al. [2013]). Accurate power load forecast is important to reduce the operational cost in the management of the power system (Ranaweera et al. [1997]). Therefore, a lot of main stream models and methods such as Artificial Neural Networks (Bashir and El-Hawary [2009]) and Support Vector Machine (Zhang et al. [2011]) niu2010powerhave been used to solve this problem in the previous researches. Power load are related to many hidden variables such as wind, sunlight and holidays, etc. Some of these variables are difficult to obtain or quantify. It is a reasonable assumption that the values of these hidden variables are similar for the cities located in the same region. Therefore, the power load data sets of these nearby cities should be highly correlated, and the multi-task learning methods will increase the prediction accuracies.

In this paper, Gaussian process models will be used to perform predictions for multiple time series. Since there are more data sets in the multi-task problems, computational complexity becomes an important issue. Furthermore, in short term power load forecast, the calculation time of the prediction value is limited by the sample interval. In fact it should be much shorter than the sample interval, otherwise the forecasting will be meaningless. In the previous multi-task Gaussian process models in Bonilla et al. [2008], the parameters of the covariance function of single tasks are connected by a task similarity matrix. This leads to a complex structure for the multi-task covariance function. In this work we treat the data in the multiple inputs and multiple outputs manner and minimized the number of hyper-parameters by flattening the task similarity matrix in the multi-task covariance function. Based on this model, a novel algorithm for Gaussian Process inference with non-stationary covariance function is also proposed.

In the rest of this paper, preliminaries including introductions of the Gaussian process and its covariance functions are given in section 2. In section 3, the multi-task Gaussian process model for time series prediction and the inference algorithm are presented. Section 4 shows the numerical results of the methods on power load data sets from neighboring cities in the same geographical region.

2. PRELIMINARIES

2.1 Gaussian Process

In this section, a brief introduction of the standard Gaussian Process model is given.

Consider a regression problem with the data set $\{(x_t, y_t)|t = 1, ..., N\}$. We are going to construct a predictor that satisfies

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

where x_t is the feature vector at time t , and y_t is the corresponding scalar output. e_t is the additive white noise with

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

Gaussian distribution $e_t \sim N(0, \sigma_n^2)$. Assume that the mean of the output process is zero, the distribution of the time series $Y = [y_1, y_2, ..., y_N]^T$ can be written as:

$$Y \sim N(0, K(X, X) + \sigma_n^2 I)$$

where $X = [x_1, x_2, ..., x_N]^T$ and K is called the Gram matrix. The elements of K is $K_{ij} = k(x_i, x_j)$ where $k(\cdot, \cdot)$ is the covariance function which satisfies $k(a, b) = k(b, a)$. The explicit expressions of the covariance functions will be discussed in next subsection. I is the identity matrix. The discussion of Gaussian process with non-white noises can be found in Murray-Smith and Girard [2001]. The identity matrix will be substituted by a matrix whose entries are the parameters of the ARMA process.

For a new input x_*, the joint distribution of the output without disturbed noises, and the history training data is

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

where

$$\bar{k}(x_*, X) = [k(x_*, x_1), k(x_*, x_2), ..., k(x_*, x_N)]^T.$$

We write $\hat{y}_*$ instead of f_* in prediction task. According to the joint Gaussian distribution, the expectation value and the variance of the target variable can be written as:

$$\bar{y}_* = E(\hat{y}_*) = \bar{k}^T(x_*, X)(K(X, X) + \sigma_n^2 I)^{-1} Y \quad (3)$$

$$\begin{aligned} Var(\hat{y}_*) = & k(x_*, x_*) \\ & - \bar{k}^T(x_*, X)(K(X, X) + \sigma_n^2 I)^{-1} \bar{k}(x_*, X) \end{aligned} \quad (4)$$

Note that for the prediction problems of time series , the feature vector is

$$x_t = [y_{t-1}, ..., y_{t-D}].$$

D is the feature length selected by the users.

In this paper, we only consider one step ahead prediction. For the multiple steps ahead time series prediction problems, if the predicted values in the previous steps are used as the elements of the feature vector, the input vector will be considered as random variables. See Girard et al. [2003] for more details.

2.2 Covariance Functions

The covariance function is a core concept in the Gaussian process model. It calculates the relatedness between two data points in the feature space. For non-stationary time series, the linear trend covariance function (Brahim-Belhouari and Bermak [2004]) is:

$$k_{ns}(x_i, x_j) = {x_i}^T x_j \quad (5)$$

where x_i is the feature vector of the model at time i ,and $x_i^T x_j$ is the inner product of the feature vector x_i and x_j.

The weighted version of the linear trend covariance function can be written as:

$$k_{ns\_ard}(x_i, x_j) = {x_i}^T L x_j \quad (6)$$

where L is a D by D diagonal matrix whose diagonal elements are $l_1, ..., l_D$. These are called hyper parameters in Gaussian Process model.

The hyper parameters in the matrix L can be learned from the data by solving an optimization problem. This is called the Automatic Relevance Determination (ARD). The hyper parameter vector can be written as:

$$\theta = [l_1, ..., l_D]$$

Note that the features with small weight l_i will make a small contribution to the prediction results.

In order to learn the optimal values of the hyper parameters in the covariance function, the most likelihood estimation (MLE) based iterative methods, such as conjugate gradient, can be used here. The likelihood can be written as:

$$\begin{aligned} \log p(y|X, Y, \theta) = & -\frac{1}{2} Y^T (K + \sigma_n^2 I) Y \\ & - \frac{1}{2} \log |K + \sigma_n^2 I| - \frac{n}{2} \log 2\pi \end{aligned} \quad (7)$$

$$\hat{\theta} = \arg\max(\log p(y|X, Y, \theta)) \quad (8)$$

Note that for noisy models, the variance of the additive noise σ_n^2 is also estimated by this optimization programming procedure.

In addition, in multi-task circumstances, the length of the parameter vector will be multiplied and seriously affects the efficiency of the ARD calculation.

3. MULTI-TASK GP MODEL AND ALGORITHM

In multi-task learning problems, the observed input and output data sequences from different tasks are connected by the hidden variables as depicted in Figure 1. Therefore additional information can be obtained from data sets of the neighboring tasks. Furthermore, these tasks may also share the same underlying noises. From this point of view, larger number of the observations from different tasks can also decrease the influences of the disturbed noises.

The model of multi-task Gaussian Process is given in Section 3.1 and the inference algorithm is in Section 3.2.

3.1 Multi-task GP model

Given a M time series of length N, we focus on the prediction at time t, the training data can be represented as:

$$Y_t = \begin{bmatrix} y_{t-1}^{(1)} & y_{t-1}^{(2)} & \cdots & y_{t-1}^{(M)} \\ y_{t-2}^{(1)} & y_{t-2}^{(2)} & \cdots & y_{t-2}^{(M)} \\ \vdots & \vdots & \ddots & \vdots \\ y_{t-n}^{(1)} & y_{t-n}^{(2)} & \cdots & y_{t-n}^{(M)} \end{bmatrix}$$

Y_t is a n-by-M matrix. Note that we write Y instead of Y_t whenever this causes no confusion. Let y_{t-i} be the ith row of Y. And let $Y^{(j)}$ be the jth column of Y, which is the data from the single task j.

Similarly, the input vector can be written as:

$$X_t = \begin{bmatrix} x_{t-1}^{(1)} & x_{t-1}^{(2)} & \cdots & x_{t-1}^{(M)} \\ x_{t-2}^{(1)} & x_{t-2}^{(2)} & \cdots & x_{t-2}^{(M)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{t-n}^{(1)} & x_{t-n}^{(2)} & \cdots & x_{t-n}^{(M)} \end{bmatrix}$$

where

$$x_i^{(m)} = [y_{i-1}^{(m)}, y_{i-2}^{(m)}, ..., y_{i-D}^{(m)}]$$

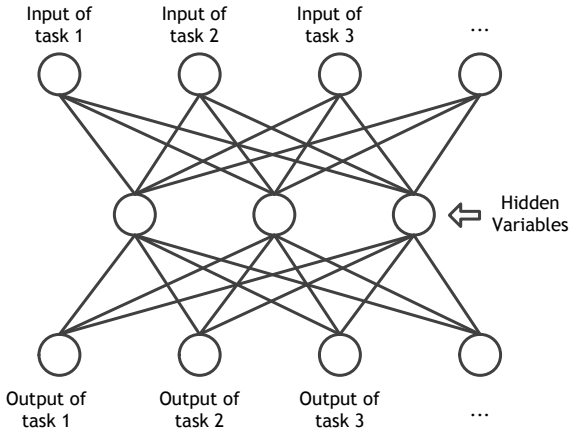

Fig. 1. Input variables, output variables, and hidden variables in the multi-task learning problem.

X_t is a n-by-$D \times M$ matrix, D is the feature length selected by users. We also write X instead of X_t whenever this causes no confusion. Let x_{t-i} be the be the ith row of X.

Note that the length of training data for prediction at time t is n and $n \ll N$, N is the length of the whole time series and is continuous to increase in the precess of the prediction. n can be taken as the length of the training data's window on the whole history data sequence.

Next, for jth task, the covariance function can be written as the sum of the single task covariance functions,

$$k(x_p, x_q, \theta_j) = \sum_{m=1}^{M} x_p^{(m)} L_m(\theta_j) x_q^{(m)} \tag{9}$$

Since L_m, $m = 1, .., M$ are diagonal matrices. The hyper parameter can be written as:

$$\theta_j = [\theta_{j1}^T, \theta_{j2}^T, ..., \theta_{jM}^T]^T$$

where θ_{ji} is a D-by-1 vector, thus $L_m(\theta_j)$ is a D-by-D diagonal matrix for task j and θ_j is a $M \times D$-by-1 vector.

The values of the hyper-parameters can be obtained by Equation (7) and (8). Note that Gaussian Process is non-parametric model, so the calculation of hyper parameters is equivalent to the model structure selection step in the parametric models. The optimation calculation has to be done just once for each task before the prediction steps.

The predictor of the expectations and the variances for the task j at a new step can be obtained in the similar way as in (3) and (4). They are written in Equations (10) and (11) where some parameters are omitted for simplicity.

$$\bar{y}_t^{(j)} = E(\hat{y}_t^{(j)}) = \bar{k}^T(x_t, X)(K(X,X) + \sigma_n^2 I)^{-1} Y^{(j)} \tag{10}$$

$$\begin{aligned} Var(\hat{y_t}^{(j)}) &= k(x_t, x_t) \\ &- \bar{k}^T(x_t, X)(K(X,X) + \sigma_n^2 I)^{-1} \bar{k}(x_t, X) \end{aligned} \tag{11}$$

3.2 Inference Algorithm

The computational complexity is an important issue in the multi task problems. The major time cost in Gaussian Process inference comes from the matrix inversion operation in (10) and (11). Furthermore, when the data sets is large, the inverse matrix will be singular.

By using the matrix inversion lemma:

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(DA^{-1}B + C^{-1})^{-1}DA^{-1}$$

the the inverse operation in Equation (10) and (11) can be converted to be operated on a smaller matrix. The inverse matrix in Equation (10) and (11) can be written as:

$$\begin{aligned} &(K(X,X) + \sigma_n^2 I)^{-1} = (XLX^T + \sigma_n^2 I)^{-1} \\ &= \frac{1}{\sigma_n^2} I - \frac{1}{\sigma_n^2} X(\sigma_n^2 L^{-1} + X^T X)^{-1} X \end{aligned} \tag{12}$$

where L is a $M \times D$-by-$M \times D$ diagonal matrix whose diagonal elements are the corresponding diagonal elements of $L_1, L_2, ..., L_M$ in (9).

Following the equation (12), the computational complexity is reduced from $o(n^3)$ to $o((D \times M)^3)$. Note that $M \times D$ is equivalent to the number of hyper- parameters. It is a valid assumption that numbers of data should be much larger than the numbers of parameters in most statistical learning problems. Since we have $D \times M \ll n$, the computational complexity is effective reduce by implementing Equation (12).

When n is large, the direct representation of the inversion matrix of $K(X,X)+\sigma_n^2 I$ may have serious round off problems. The elements of the matrix K will increase with n, therefore the element of the inverse matrix will be very small. Instead, we always calculate the intermediate variable:

$$\alpha = \frac{1}{\sigma_n^2}\{Y^{(j)} - X(\sigma_n^2 L^{-1} + X^T X)^{-1} X^T Y^{(j)}\} \tag{13}$$

for the calculation of the new output and

$$\beta = \frac{1}{\sigma_n^2}\{\bar{k} - X(\sigma_n^2 L^{-1} + X^T X)^{-1} X^T \bar{k}\} \tag{14}$$

for the calculation of the variance of the new output.

The matrix inversion in (13) and (14) can be solved by the Cholesky decomposition. Let U be the upper triangle cholesky decomposition matrix of an positive definite matrix A,

$$U = chol(A) \tag{15}$$

where

$$A = UU^T$$

The computational complexity of (15) is $n^3/6$, where n is the size of matrix A. $A^{-1}b$ can be calculated by solving $AX = b$:

$$A^{-1}b = U^{-T}U^{-1}b \tag{16}$$

The complexity of (16) is $n^2/2$.

So in order to solve the inverse operation in Equation (13) and (14), let

$$\begin{aligned} A &= \sigma_n^2 L^{-1} + X^T X \\ b &= X^T Y^{(j)} \\ b' &= X^T \bar{k} \end{aligned}$$

Note that L is a diagonal matrix whose inverse matrix can be easily obtain with $o(n)$ complexity by calculating $1/l_{ii}$, l_{ii} is the diagonal elements of L.

α and β can be written as:

$$\alpha = \frac{1}{\sigma_n^2}\{Y^{(j)} - XK_\alpha\} \tag{17}$$

$$\beta = \frac{1}{\sigma_n^2}\{\bar{k} - XK_\beta\} \tag{18}$$

INPUT: $X, Y, \theta, \sigma_n^2, x_*$
OUTPUT: $\bar{y}_*, \mathrm{var}(\hat{y}_*)$

1: for j = 1:M
2: $L = diag(\theta^{(j)})$
3: $A = \sigma_n^2 L^{-1} + X^T X$
4: $b = X^T Y^{(j)}$
5: $U = chol(A)$
6: $K_\alpha = U^{-T} U^{-1} b$
7: $\alpha = \frac{1}{\sigma_n^2}(Y^{(j)} - X K_\alpha)$
8: $\bar{k} = X L x_*^T$
9: $y^{(j)} = \bar{k}\alpha$
10: $b' = X^T \bar{k}$
11: $K_\beta = U^{-T} U^{-1} b'$
12: $\beta = \frac{1}{\sigma_n^2}(\bar{k} - X K_\beta)$
13: $\mathrm{var}(y^{(j)}) = x_* L x_*^T - \bar{k}\beta$
14: end for
15: $\bar{y}_* = [y^{(1)}, y^{(2)}, ..., y^{(M)}]$
16: $\mathrm{var}(\hat{y}_*) = [\mathrm{var}(y^{(1)}), \mathrm{var}(y^{(2)}), ..., \mathrm{var}(y^{(M)})]$

Fig. 2. Algorithm1: Multi-task time series Prediction Algorithm for Multi-task Gaussian Process with non-stationary linear trend covariance function

where $K_\alpha = A^{-1}b$ and $K_\beta = A^{-1}b'$ are calculated by (15) and (16).

The algorithm is summarized as Algorithm 1 in the Figure 2. The time complexity of line 5 in Algorithm 1 is $(MD)^3/6$, line 6 and line 11 is $(MD)^2/2$.

4. POWER LOAD DATA ILLUSTRATION

4.1 Data set and accuracy evaluation

The data sets in this work are chosen from major cities of Jiangxi province in China. Electricity power load data are sampled every 15 minutes for each city. Our target is to predict the value of the electric power load in the next 15 minutes. In Figure 3 and Figure 4, data sets from 3 cities are plotted in order to demonstrate the relatedness of these data sets.

Figure 3 shows the short term power loads of these cities in five days. And in Figure 4, we sampled the short term data sequences once a day at the same clock. It shows that the long term trends of these power load data sets are also highly correlated. In addition the non-stationary properties of the power load data can also be observed from Figure 4.

As shown in Figures 3 and 4, since the absolute values of power load for the selected cities are not in the same range. The normalize criteria should be used to evaluate the prediction errors. Normalized mean square error is used in this paper.

$$NMSE = \left\| \frac{\hat{x}(t) - x(t)}{\hat{x}(t) - mean(x(t))} \right\|^2 \qquad (19)$$

where $\hat{x}(t)$ is the estimated value of $x(t)$.

We did not adopt any of the pre-process technics such as standardizing the data sequence to have zero means and unit variances or eliminating the non-stationary properties. On the contrary, these are essential information for the Gaussian Process model prediction.

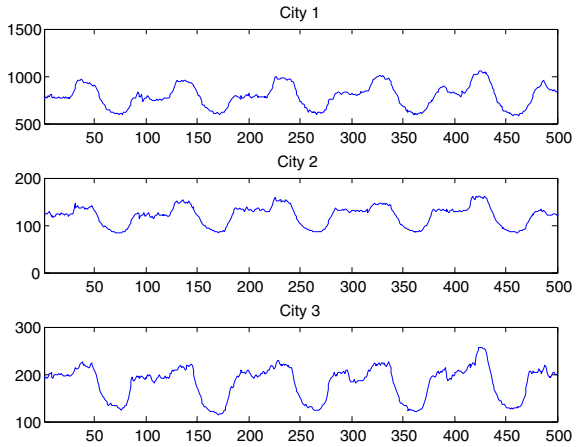

Fig. 3. Comparison of the power load data from 3 nearby cities (sample interval: 15minutes)

Table 1. Prediction errors of the multi-task GP and single-task GP

Method	NMSE for city 1	NMSE for city 2	NMSE for city3
Multi-task GP	0.0054	0.0052	0.0043
Single-task GP	0.0072	0.0057	0.0050

Three groups of experiments have been done and demonstrated in the following subsections. In section 4.2, we compare the prediction errors of the multi-task method and the single task method (Rasmussen and Nickisch [2010]). We show that the data sequences provide additional information for the other tasks in the prediction. Next in section 4.3, a group of methods are compared with the proposed method on accuracy and running time. Time cost, which is important for short-term online prediction, is reduced. Finally in section 4.4, a simulation with data sequences from 9 cities is represented.

The experiments are done by MATLAB on Intel Core i7-4770K CPU @3.50GHz.

4.2 Comparison between single task Gaussian Process and multi-task Gaussian Process

In this experiment we use the data sets of three cities as shown in Figure 3 and 4. There are 6000 data points for each sequence. And 1000 points are used for hyper-parameter estimating and 5000 steps are used for prediction method validation. Other parameters are $D = 10$, $M = 3$, $n = 1000$.

The errors are listed in Table 1. The prediction errors of the multi-task cases are significantly smaller than their corresponding single-task cases.

The multi-task cases also have smaller variances as shown in Figure 5. Variance of the predictor can be taken as the confidence of the prediction, which is calculated by the joint Gaussian distribution. It is different from the prediction error,

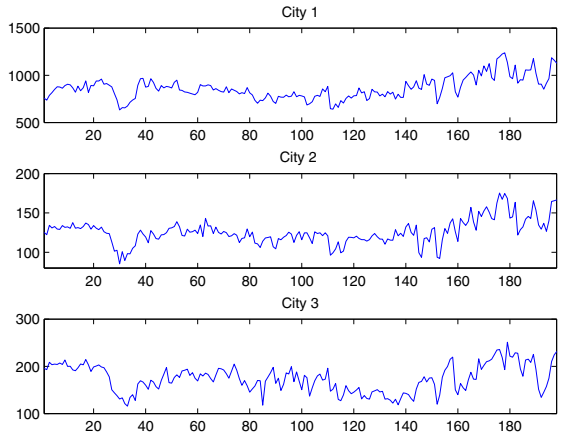

Fig. 4. Comparison of the power load data from 3 nearby cities (sample interval: 24hours)

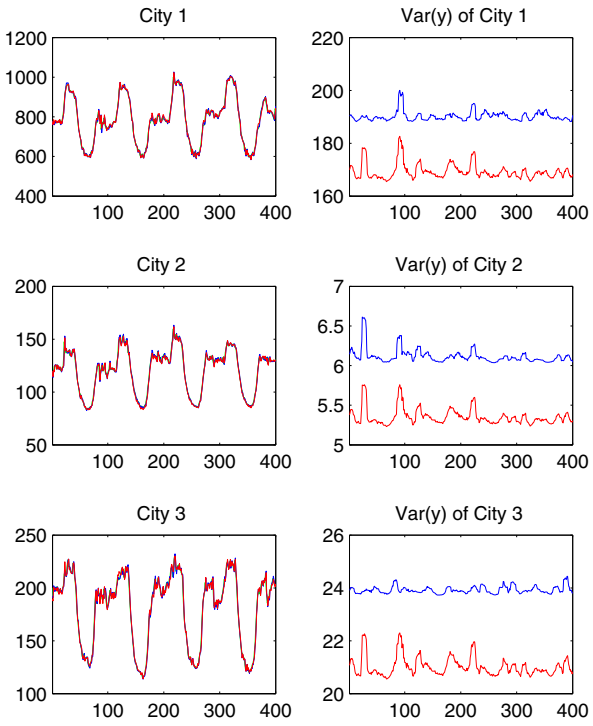

Fig. 5. First 400 data points of the prediction are plotted. Left column: comparison of the prediction value of multi-task GP and single-task GP (Green line: true value, Blue line: prediction of single task GP, Red line: prediction of multi-task GP); Right column: comparison of the variance of the prediction of multi-task GP and single task GP (Blue line: prediction variances obtained by the single task GP; Red line: prediction variances obtained by the multi-task GP).

Table 2. Comparison of the prediction errors and time cost of Multi-task GP (Algorithm1), Standard Single-task GP, Multi-task GP (Bonilla), Multi-task AR, Single-task AR

Method	Average NMSE	Running time for prediction (second per step)
Multi-task GP (Algorithm1)	0.00496	7.968e-3
Standard Single-task GP	0.00596	5.801e-1
Multi-task GP (Bonilla)	0.00496	6.507e-1
Multi-task AR	0.01452	2.325e-3
Single-task AR	0.01797	2.970e-3

Table 3. Comparison of the hyper parameter learning time of Multi-task GP model in section 3.1, Multi-task GP model(Bonilla)

	Multi-task GP model in section 3.1	Multi-task GP model in (Bonilla)
Hyper-parameter learning time	18.17s	210.70s

which is calculated by the true value and the predicted value using (19).

4.3 Methods comparison

In this section, we compare the average NMSE and time cost of 5 different methods. The data sets and parameters are the same with that of section 4.2. Note that the running time for single task methods are the sum of all tasks.

The algorithm for the auto-regression (AR) model in this experiment is the recursive least square (RLS) algorithm in Ljung [2007]. The orders of the AR models equals to the parameter n in Gaussian process model. The Multi-task GP (Bonilla) method is in Bonilla et al. [2008].

The Algorithm 1 in section 3 greatly reduces the computational complexity for the multi-task Gaussian Process with non-stationary covariance function meanwhile the accuracy is as good as the complex model. In addition, smaller number of hyper-parameters reduces the time cost on hyper parameter estimation greatly. The comparison of the learning time cost of the hyper-parameters is represented in Table 3. Note that the comparison uses the same optimization algorithm. The multi-task model with task similarity matrix in Bonilla et al. [2008] has much complex structures, whereas the flattened model in section 3.1 runs much faster without losing transferred information from other tasks.

4.4 Experiment with larger number of data sets

In this section, we use more data sets from 9 neighbouring cities and the result is show in Table.4. The improvement percentages are listed.

Table 4. Comparison of the NMSE of the single task GP and multi-task GP on larger number of data sets

City	NMSE (single-task)	NMSE (Multi-task)	Improvement on prediction errors
Nanchang	0.0072	0.0055	23.93%
Yichun	0.0091	0.0075	17.41%
Jingdezhen	0.0057	0.0053	6.74%
Jiujiang	0.0091	0.0075	17.41%
Shangrao	0.0193	0.0184	4.54%
Fuzhou	0.0050	0.0042	15.28%
Ganxi	0.0703	0.0611	13.05%
Ji'an	0.0212	0.0183	13.83%
Ganzhou	0.0162	0.0143	11.99%

Note that part of the data sets are already used in the experiments in the above subsections. Some of the prediction errors are not significantly improved comparing with the previous multi-task experiments which contain smaller number of data sets. This indicates that not all of these tasks get more information from the newly added data sets, therefore the data sets can be further clustered in to smaller groups. The conditions of a successful multi-task learning is an essential issue. This will be investigated as one of the future works.

5. CONCLUSIONS AND FUTURE WORK

In this work, we investigate the multi-task Gaussian Process model and apply it to the problem of short-term electric power load forecast. The improvement on the prediction accuracy is due to the fact that the data sequences share the same hidden variables, as depicted in Figure 1. A novel algorithm for prediction of non-stationary time series using Gaussian Process regression with linear trend covariance function is proposed and greatly reduces the time cost of the multi-task regression problem.

One of the future works is how to cluster similar tasks. Obviously, unrelated time series will not provide additional information to improve the prediction accuracies. When to used multi-task method is as important as how to use it. Several researches have been raised on this issue (Rosenstein et al. [2005]) recently. However, this problem may be asymmetric. Though the data set of task A contributes to the task B, the improvement in the converse direction cannot be guaranteed. This point also indicates the slight difference between transfer learning and multi-task learning. The former focus on the improvement of one particular single task with the help of the other tasks, whereas the latter focus on the improvement for all the tasks.

REFERENCES

Z. A. Bashir and M. E. El-Hawary. Applying wavelets to short-term load forecasting using pso-based neural networks. *Power Systems, IEEE Transactions on*, 24(1):20–27, 2009.

E. Bonilla, K. M. Chai, and C. Williams. Multi-task gaussian process prediction. In *Advances in Neural Information Processing Systems*, pages 153–160, 2008.

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

N. Chen, Z. Qian, X. Meng, and I. T. Nabney. Short-term wind power forecasting using gaussian processes. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2790–2796. AAAI Press, 2013.

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

L. Ljung. System identification toolbox for use with matlab. 2007.

R. Murray-Smith and A. Girard. Gaussian process priors with arma noise models. In *Irish Signals and Systems Conference, Maynooth*, pages 147–152, 2001.

S. J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10): 1345–1359, 2010.

D. K. Ranaweera, G. Karady, and R. Farmer. Economic impact analysis of load forecasting. *Power Systems, IEEE Transactions on*, 12(3):1388–1392, 1997.

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

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

M. T Rosenstein, Z. Marx, L.P. Kaelbling, and T. Dietterich. To transfer or not to transfer. In *NIPS 2005 Workshop on Inductive Transfer: 10 Years Later*, volume 2, page 7, 2005.

J. Wang, A. Hertzmann, and D. M. Blei. Gaussian process dynamical models. In *Advances in neural information processing systems*, pages 1441–1448, 2005.

Y. Zhang, Z. Yan, and G. Luo. A new recursive kernel regression algorithm and its application in ultra-short time power load forecasting. In *Proceedings of the IFAC 18th World Congress*, volume 18, pages 12177–12182, 2011.