High-Accuracy Wireless Traffic Prediction: A GP-Based Machine Learning Approach

Yue Xu[1,2], Wenjun Xu[1], Feng Yin[2], Jiaru Lin[1], Shuguang Cui[3,2]

[1]Beijing University of Posts and Telecommunications

[2]SRIBD and The Chinese University of Hong Kong, Shenzhen

[3]University of California, Davis

xuy@bupt.edu.cn

December 7th, 2017

Overview

Overview

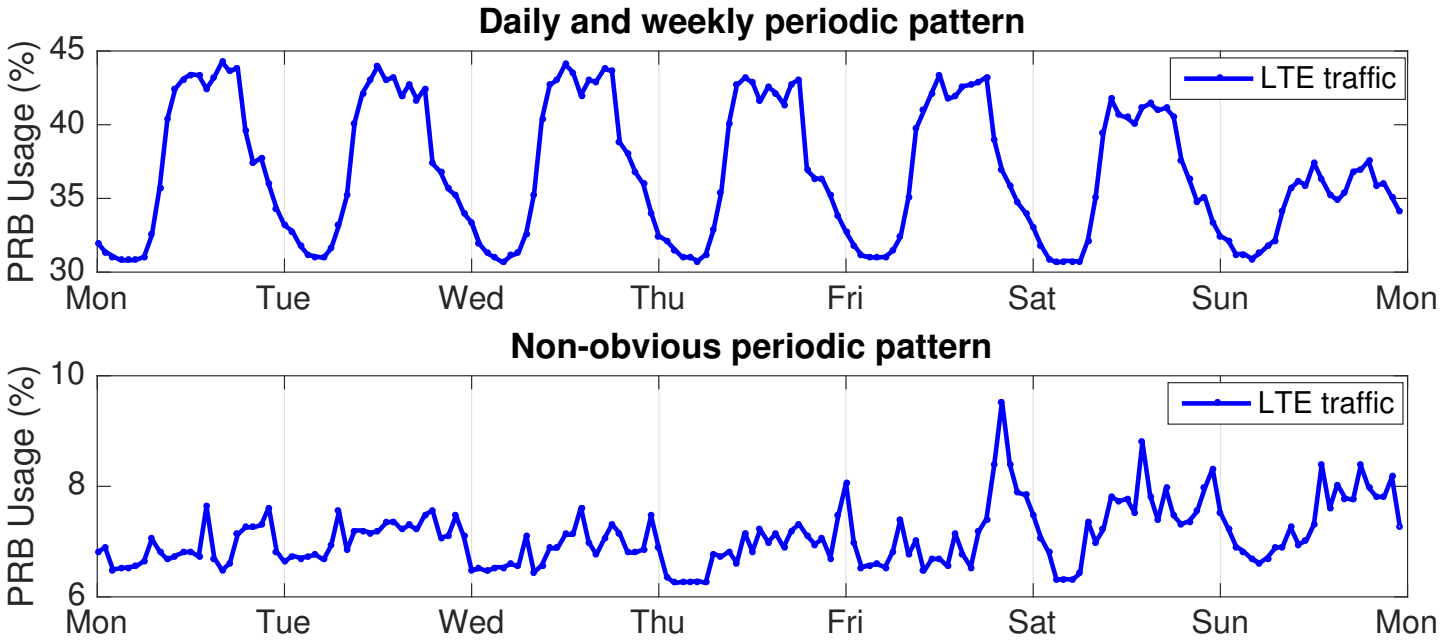
Background

Wireless traffic prediction is essential for efficient management of wireless networks, e.g.,

- Energy-saving in cognitive wireless networks
  - Design proper BS switch on/off mechanisms based on peak and average wireless traffic
- Mobile load balancing in self-organizing networks (SON)
  - Design proper user handover mechanisms based on predicted traffic load.

The task of wireless traffic prediction is to learn from the historical and current behavior of the system, and relates the past and current traffic with the future traffic

Characteristic of the LTE traffic



- The overall variation is complicated due to user mobility, user behavior and diverse applications
- However, regular human activities drive the wireless traffic to form certain regular patterns

Related work

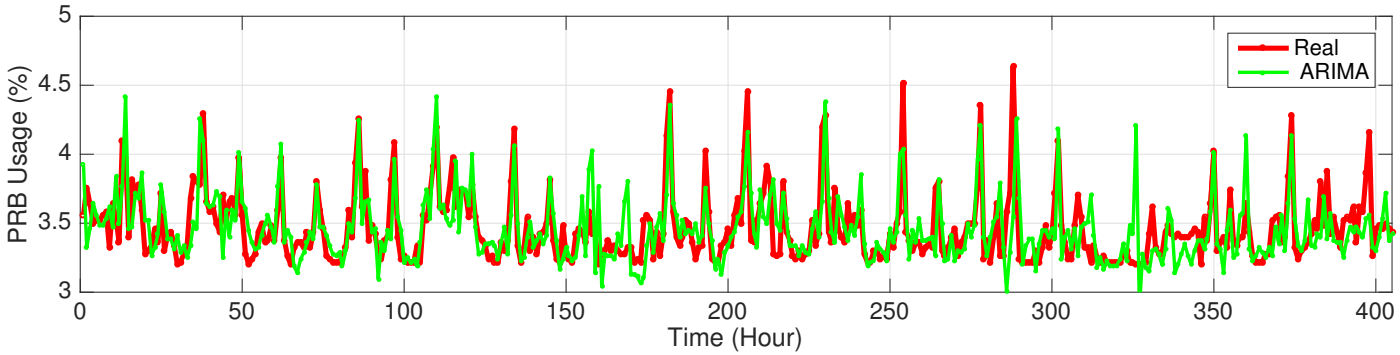
How to find and describe the patterns?

Existing statistical time series modeling and analysis methods, e.g.,

- Autoregressive integrated moving average (ARIMA)
- Sinusoid superposition, based on fast Fourier transform (FFT)

ARIMA

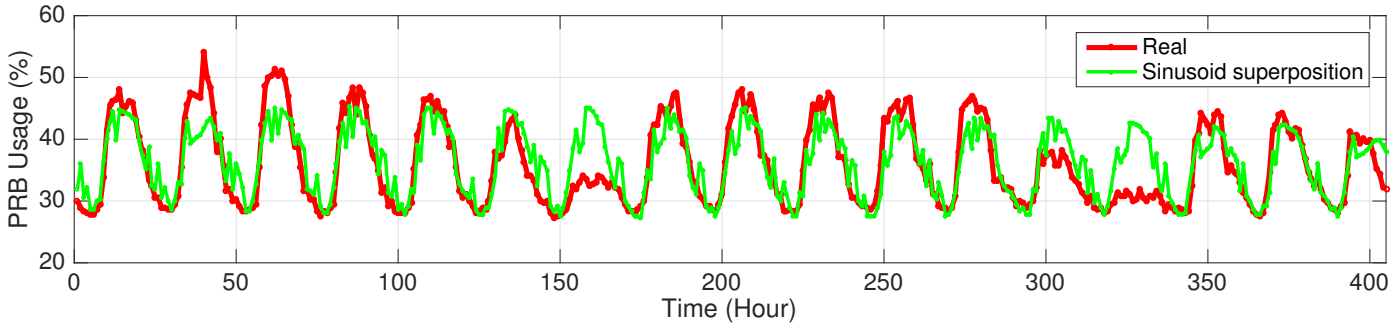
However, ARIMA models are unable to capture **long-range dependent characteristics**



- ARIMA predicts the future mainly by using **linear combination** of past information, e.g., past traffic, regression error
- ARIMA could be largely influenced by the sudden peaks, results in unstable performance.

Sinusoid Superposition

Sinusoid superposition only can generate rigid periodic patterns according to the main frequency components based on FFT.



- Sinusoid superposition needs to **manually** choose main frequency components
- The coefficient parameters is based on curve fitting

Strength of Gaussian Process

Recently, Gaussian process, which is one type of **nonparametric** machine learning methods, has achieved outstanding results in various fields, which has the ability to

- capture the uncertainty and nonlinearity of traffic time series
- be free from black-box operations
- encode domain/expert knowledge into kernel functions
- optimize the hyperparameters explicitly

However, the bottleneck of current GP methods lies in the **high computational complexity**.

Contributions

The main contributions of this paper are as follows.

- The first to apply GP in wireless traffic prediction.
- The first to predict traffic based on real 4G traffic data
- Achieve prediction accuracy up to 97%, much higher than the existing methods.
- Exploit the Toeplitz structure, reduce the complexity of
  - hyperparameter learning: from $O(n^3)$ to $O(n^2)$
  - inference: from $O(n^3)$ to $O(n \log n)$

  without sacrificing any prediction accuracy

Overview

Regression model

We consider the regression model

$$y = f(t) + \epsilon \tag{1}$$

where

- t is the input, i.e., the time index
- y is the output, i.e., the traffic flow
- $f(\cdot)$ is the underlying regression function
- ϵ is the observation noise, assumed to be Gaussian i.i.d follow $\mathcal{N}(0, \sigma^2)$, but the noise variance σ^2 is supposed to be **unknown**

Prediction task

Given a training dataset $\mathcal{S} = \{\boldsymbol{t}, \boldsymbol{y}\}$ where

- $\boldsymbol{t} = [t_1, t_2, ..., t_n]$ contains the training inputs
- $\boldsymbol{y} = [y_1, y_2, ..., y_n]$ contains the training outputs

We aim to predict $\boldsymbol{y}_* = [y_{*,1}, y_{*,2}, ..., y_{*,n_*}]$, whose test inputs are $\boldsymbol{t}_* = [\boldsymbol{t}_{*,1}, \boldsymbol{t}_{*,2}, ..., \boldsymbol{t}_{*,n_*}]$.

Gaussian Process Definition

Definition

A Gaussian process is a collection of random variables, any finite number of which have (consistent) Gaussian distributions.

A real-valued Gaussian process, which is completely specified by a mean function and covariance function (a.k.a. kernel function). Concretely,

$$f(\boldsymbol{t}) \sim \mathcal{GP}(m(\boldsymbol{t}), k(\boldsymbol{t}, \boldsymbol{t}'; \boldsymbol{\theta})) \tag{2}$$

where

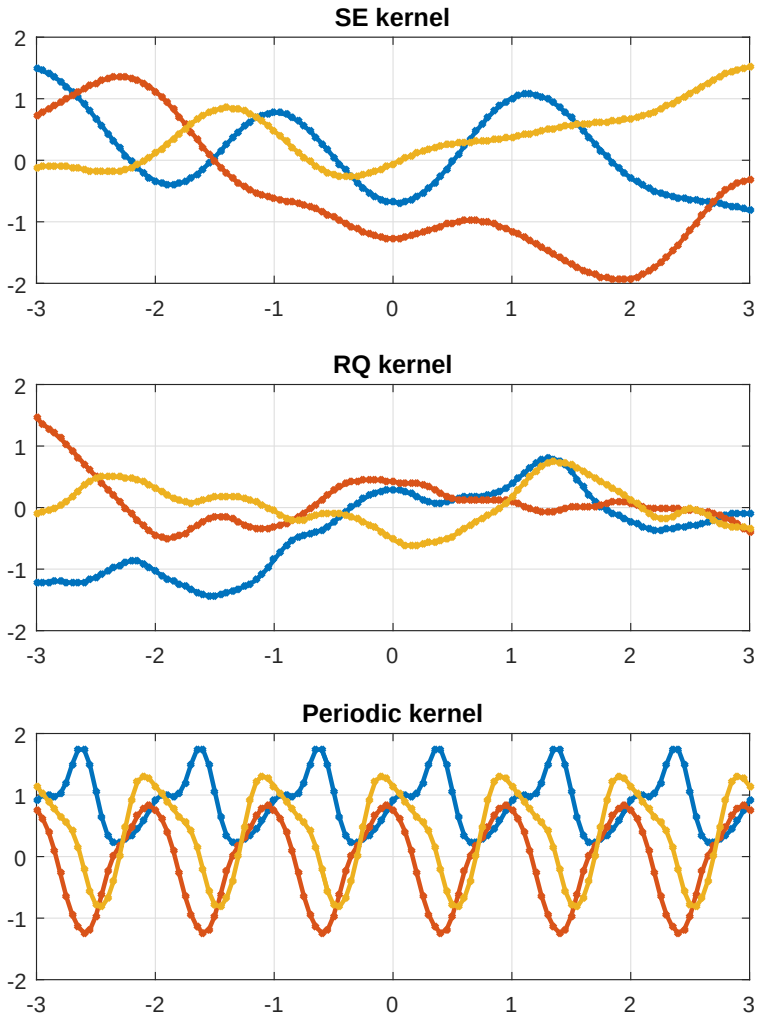
- $m(\boldsymbol{t})$ is the mean function, often set to zero in practice, if no prior knowledge is available
- $k(\boldsymbol{t}, \boldsymbol{t}'; \boldsymbol{\theta})$ is the covariance kernel function
- $\boldsymbol{\theta}$ denotes the **unknown** hyper-parameters to be optimized

Kernel function

The choice of kernel function is critical to GP, for that kernels

- encode prior information about underlying data
- define distinct distributive curves

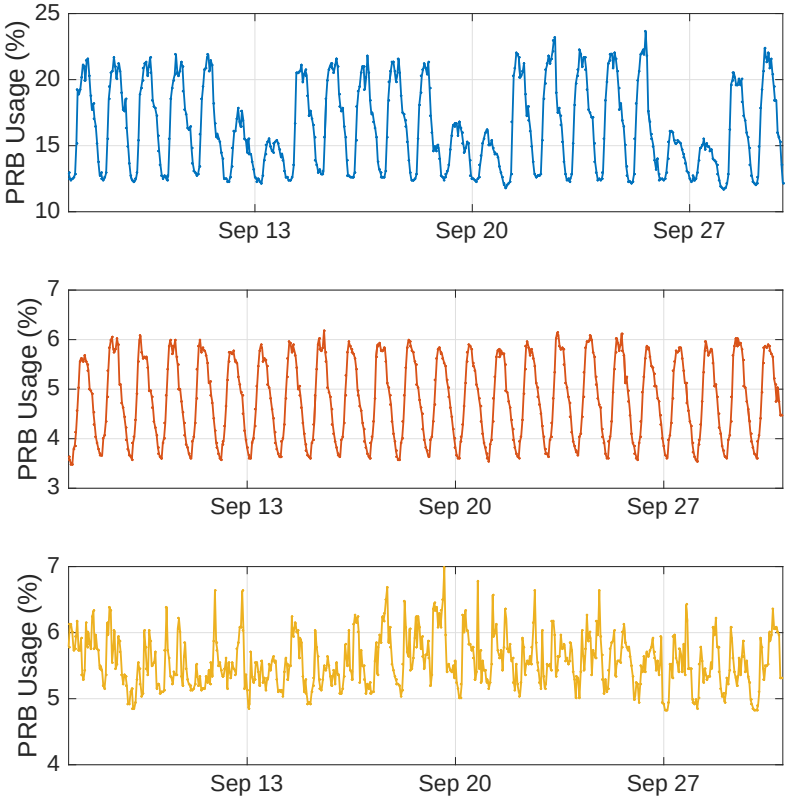
Therefore, we need to design a proper kernel to best fit the wireless traffic.



Observed Patterns

In general, the 4G wireless traffic in our dataset shows four common patterns

1. the weekly periodic pattern
2. the daily periodic pattern
3. the dynamic deviations
   - irregular traffic variations
4. the residue noise
   - local disturbing noise, e.g., measurement inaccuracy



Kernel selection

We sum four kernels to model the four observed patterns respectively

- a periodic kernel function for weekly pattern:

$$k_1(t_i, t_j) = \sigma_{p_1}^2 \exp\left[-\frac{\sin^2 \frac{\pi(t_i - t_j)}{\lambda_1}}{l_{p_1}^2}\right]$$

- a periodic kernel function for daily pattern:

$$k_2(t_i, t_j) = \sigma_{p_2}^2 \exp\left[-\frac{\sin^2 \frac{\pi(t_i - t_j)}{\lambda_2}}{l_{p_2}^2}\right]$$

- a rational quadratic (RQ) kernel for dynamic deviation:

$$k_3(t_i, t_j) = \sigma_{s_t}^2 \left[1 + \frac{(t_i - t_j)^2}{2\alpha l_{s_t}^2}\right]^{-\alpha}$$

- a squared exponential (SE) kernel for residue system noise:

$$k_4(t_i, t_j) = \sigma_{l_t}^2 \exp\left[-\frac{(t_i - t_j)^2}{2 l_{l_t}^2}\right]$$

Log-likelihood function

The prediction performance of GP regression depends on not only the kernel but also the hyper-parameters.

The *log-likelihood function* of the Gaussian prior is

$$\log p(\boldsymbol{y};\boldsymbol{\theta}) = -\frac{1}{2}\left\{\log|\boldsymbol{C}(\boldsymbol{\theta})| + \boldsymbol{y}^T\boldsymbol{C}^{-1}(\boldsymbol{\theta})\boldsymbol{y} + n\log(2\pi)\right\} \tag{3}$$

where $\boldsymbol{C}(\boldsymbol{\theta}) \triangleq \boldsymbol{K}(\boldsymbol{t},\boldsymbol{t}) + \sigma^2\boldsymbol{I}_n$.

The dominant method for learning the optimal set of hyper-parameters is via maximizing the *log-likelihood function* $\log p(\boldsymbol{y};\boldsymbol{\theta})$.

Hyper-parameter Learning

Maximizing the log-likelihood function is equivalent to minimizing the following *cost function*

$$\boldsymbol{\theta}_{ML} = \arg\min_{\boldsymbol{\theta}} g(\boldsymbol{\theta}) = \boldsymbol{y}^T \boldsymbol{C}^{-1}(\boldsymbol{\theta})\boldsymbol{y} + \ln|\boldsymbol{C}(\boldsymbol{\theta})| \tag{4}$$

In the GP society, this is mostly solved via gradient based algorithms, such as LFGS-Newton or conjugate gradients, with the gradient:

$$\frac{\partial}{\partial\theta_i} g(\boldsymbol{\theta}) = tr\left(\boldsymbol{C}^{-1}(\boldsymbol{\theta})\frac{\partial \boldsymbol{C}(\boldsymbol{\theta})}{\partial\theta_i}\right) - \boldsymbol{y}^T \boldsymbol{C}^{-1}(\boldsymbol{\theta})\frac{\partial \boldsymbol{C}(\boldsymbol{\theta})}{\partial\theta_i}\boldsymbol{C}^{-1}(\boldsymbol{\theta})\boldsymbol{y} \tag{5}$$

$$= tr\left(\left(\boldsymbol{C}^{-1}(\boldsymbol{\theta}) - \boldsymbol{\alpha}\boldsymbol{\alpha}^T\right)\frac{\partial \boldsymbol{C}(\boldsymbol{\theta})}{\partial\theta_i}\right) \tag{6}$$

where $\boldsymbol{\alpha} = \boldsymbol{C}^{-1}(\boldsymbol{\theta})\boldsymbol{y}$.

Inference

We can derive the posterior distribution which is also Gaussian with the mean $\hat{\boldsymbol{\mu}}$ and covariance matrix $\hat{\boldsymbol{\sigma}}$ as follows:

$$\hat{\boldsymbol{\mu}} = \boldsymbol{K}(\boldsymbol{t}_*, \boldsymbol{t}) \left[\boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t}) + \sigma^2 \boldsymbol{I}_n\right]^{-1} \boldsymbol{y} \tag{7}$$

$$\hat{\boldsymbol{\sigma}} = \boldsymbol{K}(\boldsymbol{t}_*, \boldsymbol{t}_*) + \sigma^2 \boldsymbol{I}_{n_*} - \boldsymbol{K}(\boldsymbol{t}_*, \boldsymbol{t}) \left[\boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t}) + \sigma^2 \boldsymbol{I}_n\right]^{-1} \boldsymbol{K}(\boldsymbol{t}, \boldsymbol{t}_*) \tag{8}$$

Note that the variance $\hat{\boldsymbol{\sigma}}$ is independent of the observed outputs $\boldsymbol{y}$.

Therefore, the posterior distribution of a set of test outputs can then be written as

$$p(\boldsymbol{y}_*|\mathcal{D}, \boldsymbol{x}_*, \boldsymbol{\theta}) \sim \mathcal{N}\left(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\sigma}}\right). \tag{9}$$

Computational Complexity

- The inference and learning in GP requires evaluating $\boldsymbol{C}^{-1}(\boldsymbol{\theta})$ and $\log|\mathbf{C}(\boldsymbol{\theta})|$
- The standard practice is to take the *Cholesky decomposition*[1] of $\boldsymbol{C}$ which requires $\mathcal{O}(n^3)$ computations
- When the dimension of $\boldsymbol{y}$ and the number of hyper-parameters to be optimized $\boldsymbol{\theta}$ is high, better strategy for parameter optimization is sought after.

[1] Decompose a symmetric, positive definite matrix C into a product of a lower triangular matrix L and its transpose, i.e. $LL^T = C$

Overview

Toeplitz Structure

The regularly-spaced input grid (time series) and stationary kernels give rise to a Toeplitz structure in the covariance matrix:

$$T = \begin{bmatrix} c_0 & c_{-1} & c_{-2} & \cdots & c_{-(n-1)} \\ c_1 & c_0 & c_{-1} & & \\ c_2 & c_1 & c_0 & & \\ \vdots & & & \ddots & \vdots \\ c_{n-1} & & & \cdots & c_0 \end{bmatrix}$$

The Toeplitz matrix T has repetive diagonal structures, i.e.,

$$T_{i,j} = T_{i+1,j+1}$$

Fast Matrix Inversion

- We expand the $n \times n$ Toeplitz matrix T to a $(2n-1) \times (2n-1)$ circulant matrix R, whose first column of R can be written as

$$\mathbf{r} = [c_1, c_2, \ldots, c_{n-1}, c_n, c_{n-1}, \ldots, c_2]$$

  and each subsequent column of R is shifted one position from its adjacent columns

- Based on the circulant structure, the matrix inversion $\boldsymbol{C}^{-1}(\boldsymbol{\theta})$ can be computed in $\mathcal{O}(n \log n)$ by preconditioned conjugate gradients (PCG)

Fast Log-Determinant

- The Trench algorithm defines a parameter γ_i as

$$|\mathbf{C}_{i+1}| = \gamma_i |\mathbf{C}_i|, 1 \leq i \leq n.$$

  where i indicates the order of the Toeplitz matrix $\mathbf{C}$.

- The computation of $|\mathbf{C}_n|$ can then be obtained by a recursion of n multiplications

$$|C_n| = \prod_{i=1}^{n-1} \gamma_i$$

- Based on the Trench algorithm, the log-determinant operation $\log|\mathbf{C}|$ can be solved in $\mathcal{O}(n^2)$ operations.

Structured Gaussian Process

In general, based on the Toeplitz structure,

- the $\mathbf{C}^{-1}\mathbf{y}$ in training and inference can be solved in $\mathcal{O}(n \log n)$ operations based on preconditioned conjugate gradients (PCG)
- the $\log |\mathbf{C}|$ in hyperparamater learning can be solved in $\mathcal{O}(n^2)$ operations based on Trench algorithm.

Reduced complexity

Therefore, the structured GP can reduce the complexity of

- inference from $\mathcal{O}(n^3)$ to $\mathcal{O}(n \log n)$
- hyperparameter learning from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$

without performance loss.

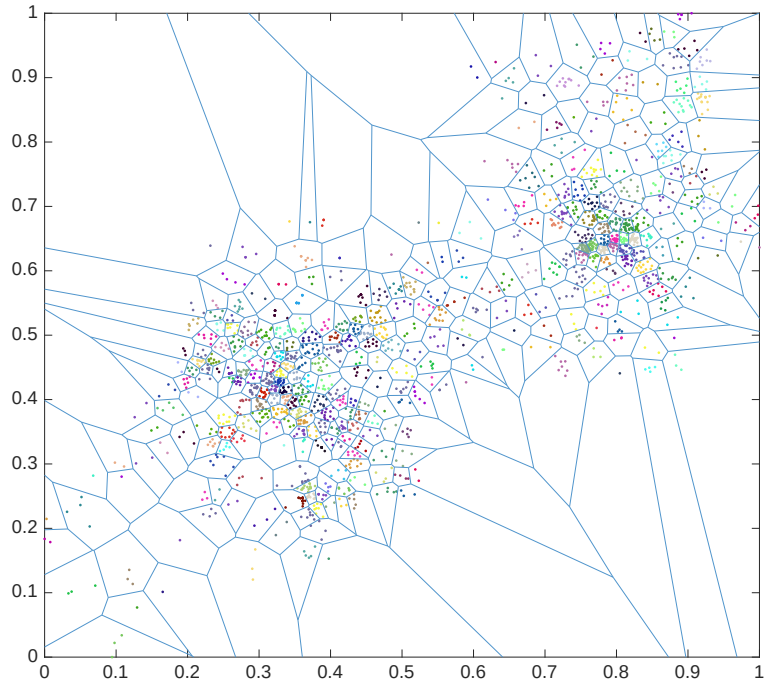
Overview

Data Description and Pre-processing

The dataset is

- hourly recorded downlink physical resource block (PRB) usage histories
- containing 3072 base stations in three southern cities in China
- from September 7th to September 30th, 2015

We cluster the base stations into 360 groups by K-means.



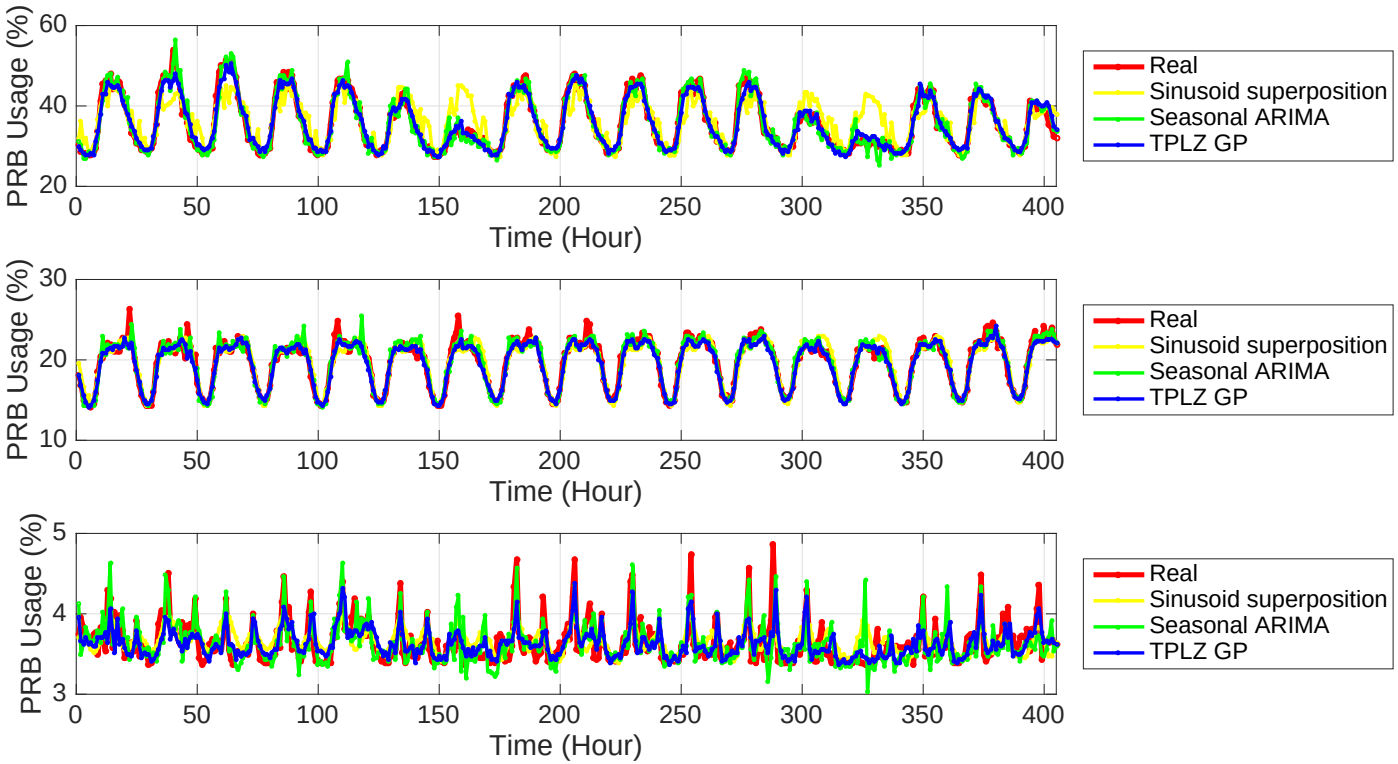
Performance Metrics

We use the mean absolute percentage error (MAPE) as the performance metric:

$$\text{MAPE} = \frac{1}{N}\sum_{t=1}^{N}\left|\frac{y(t)-r(t)}{r(t)}\right| \times 100$$

where $r(t)$ is the real PRB usage and $y(t)$ is the predicted PRB usage.

Result Analysis I

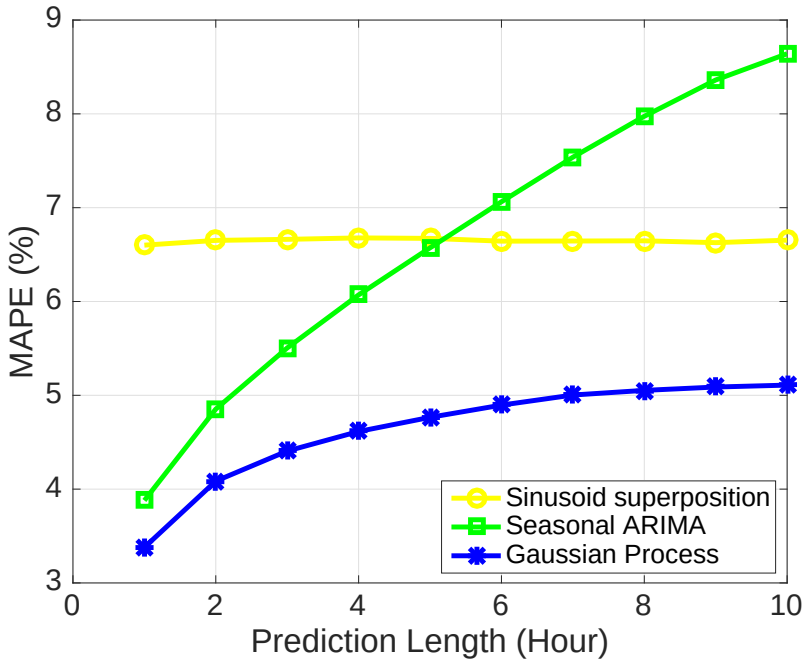


The one-hour look-ahead prediction of three virtual base stations, where TPLZ (Toeplitz) GP refers to the structured GP-based model

Result Analysis II

We show the averaged MAPE with prediction length varying from one hour to ten hours, and

- the GP model can achieve prediction error from 3% to 5%
- the SARIMA model can achieve prediction error from 4% to 8.6%
- the GP model can achieve prediction error around 6.6%



Overview

Conclusion

In this paper, we

- proposed a GP-based prediction model over real 4G traffic data
- selected kernels based on the observed regular and irregular patterns
- leveraged the Toeplitz structure to reduce the complexity
- outperformed the seasonal ARIMA and the sinusoid superposition methods