A General $O(n^2)$ Gaussian Process Regression Hyper-Parameter Optimization Using Cross-Validation and Non-linearly Constrained ADMM

Linning XU

The Chinese University of Hong Kong (Shenzhen)

116010248@link.cuhk.edu.cn

February 17, 2019

Outline

1. Motivations
   - GP Basics and GP Regression
   - Hardcore Problem in GP Hyper-parameter Tuning
2. New Schemes
   - Hold-out Cross-Validation Based Scheme
   - K-fold Cross Validation Based Scheme
   - Complexity Analysis
3. Experimental Results
   - Synthetic Data
   - Real Atmospheric CO_2 Concentration Data
4. Outlook

Outline

1. Motivations
   - GP Basics and GP Regression
   - Hardcore Problem in GP Hyper-parameter Tuning
2. New Schemes
   - Hold-out Cross-Validation Based Scheme
   - K-fold Cross Validation Based Scheme
   - Complexity Analysis
3. Experimental Results
   - Synthetic Data
   - Real Atmospheric CO_2 Concentration Data
4. Outlook

Gaussian Process

Definition [Rasmussen and Williams, 06]

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

A Gaussian process defines a distribution over function.

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}_h)), \quad \mathbb{R}^p \to \mathbb{R} \tag{1}$$

where

- $m(\boldsymbol{x})$ is the mean function, often set to zero in practice, especially when there is no prior knowledge available.
- $k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}_h)$ is the covariance/kernel function controlled by the kernel hyper-parameters $\boldsymbol{\theta}_h$.

Gaussian Process Regression

We consider the general GP regression model:

$$y = f(\boldsymbol{x}) + e, \tag{2}$$

GP regression comprises the following two steps in sequence:

1. In the training phase, a kernel function is selected, and the associated hyper-parameter $\boldsymbol{\theta}$ is tuned using the training data set $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{y}\}$ where $\boldsymbol{y} = [y_1, y_2, ..., y_n]^T$ is training outputs and $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n]$ is training input.
2. In the test phase, we obtain the posterior distribution $p(\boldsymbol{y}_*|\mathcal{D}, \boldsymbol{X}_*; \boldsymbol{\theta})$ for the test dataset $\mathcal{T} = \{\boldsymbol{X}_*, \boldsymbol{y}_*\}$ where $\boldsymbol{y}_* = [y_{*,1}, y_{*,2}, ..., y_{*,n_*}]^T$ is test output and $\boldsymbol{X}_* = [\boldsymbol{x}_{*,1}, \boldsymbol{x}_{*,2}, ..., \boldsymbol{x}_{*,n_*}]$ is test input, given the training data set $\mathcal{D}$.

Gaussian Process Regression

The posterior distribution is $p(\boldsymbol{y}_*|\mathcal{D}, \boldsymbol{X}_*; \boldsymbol{\theta}) \sim \mathcal{N}\left(\bar{\boldsymbol{m}}, \bar{\boldsymbol{V}}\right)$, where,

$$\bar{\boldsymbol{m}} = \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_e^2 \boldsymbol{I}_n\right]^{-1} \boldsymbol{y}, \tag{3}$$

$$\begin{aligned}\bar{\boldsymbol{V}} = &\boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) + \sigma_e^2 \boldsymbol{I}_{n_*} \\ &- \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) \left[\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_e^2 \boldsymbol{I}_n\right]^{-1} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*).\end{aligned} \tag{4}$$

Motivations

Importance of GP: GP models have outstanding performance in function approximation with a natural uncertainty bound. GP models constitute a class of important Bayesian non-parametric models for machine learning and are tightly connect to several other salient models, such as SVM, BNNs, RLS, RVM, ARMA and DNN nowadays.

Critical Role of GP Hyper-parameter Tuning: The predictive performance of GP regression depends on the goodness of hyper-parameter tuned in the training phase.

Deterministic ML based hyper-parameter optimization is the benchmark.

Outline

1. Motivations
   - GP Basics and GP Regression
   - Hardcore Problem in GP Hyper-parameter Tuning
2. New Schemes
   - Hold-out Cross-Validation Based Scheme
   - K-fold Cross Validation Based Scheme
   - Complexity Analysis
3. Experimental Results
   - Synthetic Data
   - Real Atmospheric CO_2 Concentration Data
4. Outlook

Benchmark Methods

The standard ML based methods minimize the negative log-marginal likelihood function as

$$\boldsymbol{\theta}_{ML} \triangleq \arg\min_{\boldsymbol{\theta}} \; l(\boldsymbol{\theta}) = \boldsymbol{y}^T \boldsymbol{C}^{-1}(\boldsymbol{\theta}) \boldsymbol{y} + \log\det\left(\boldsymbol{C}(\boldsymbol{\theta})\right), \tag{5}$$

where $\boldsymbol{C}(\boldsymbol{\theta}) \triangleq \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}; \boldsymbol{\theta}_h) + \sigma_e^2 \boldsymbol{I}_n$. Gradient Descend type methods are most widely used for this optimization task. In each iteration, each hyper-parameter is updated as follows:

$$\theta_i^{k+1} = \theta_i^k - \mu \cdot \frac{\partial l(\boldsymbol{\theta})}{\partial \theta_i}|_{\boldsymbol{\theta}=\boldsymbol{\theta}^k}, \quad \forall i = 1, 2, ..., p, \tag{6}$$

Motivations: Calls for Lox-complexity Methods

Problem: For ML schemes, in each iteration, $\boldsymbol{C}^{-1}(\boldsymbol{\theta})$ has to be re-evaluated with updated $\boldsymbol{\theta} = \boldsymbol{\theta}^k$ and multiplication of $n \times n$ matrices has to be performed for several times.

The computational complexity scales as $\mathcal{O}(n^3)$ impractical for big data!

Motivations: Calls for Lox-complexity Methods

Existing low-complexity GP methods all rely on the use of ML estimation with different types of approximations:

1. find a smaller subset ($m \ll n$) of the complete data set and construct a sparse representation of the original kernel matrix with $\mathcal{O}(m^2 n)$ complexity;
2. adopt low-rank approximation of the kernel matrices, e.g., hierarchical factoriazation of the covariance matrix into a product of block low-rank updates of the identity matrix with $\mathcal{O}(n \log^2 n)$ complexity;
3. employ a number of K local computing units, implement GP models in smaller scale with a subset of data on each local computing unit, and merge the hyper-parameter estimates with $\mathcal{O}(n^3/K^3)$ complexity.

Motivations: Calls for Lox-complexity Methods

Our Aim: a new GP hyper-parameter optimization scheme suitable for big data regime in itself ($\mathcal{O}(n^2)$) without any approximation.

Outline

1. Motivations
   - GP Basics and GP Regression
   - Hardcore Problem in GP Hyper-parameter Tuning
2. New Schemes
   - Hold-out Cross-Validation Based Scheme
   - K-fold Cross Validation Based Scheme
   - Complexity Analysis
3. Experimental Results
   - Synthetic Data
   - Real Atmospheric CO_2 Concentration Data
4. Outlook

Scheme 1: Hold-out Cross-Validation Based Scheme

We introduce a new GP hyper-parameter optimization scheme by

- replacing ML with CV
- replacing GD with ADMM

We divide the data set $\mathcal{D}$ into two non-overlapping subsets, namely the training set $\mathcal{D}_T = \{\boldsymbol{X}_T, \boldsymbol{y}_T\}$ with $|\mathcal{D}_T| = n_t$ and the validation set $\mathcal{D}_V = \{\boldsymbol{X}_V, \boldsymbol{y}_V\}$ with $|\mathcal{D}_V| = n_v$. The posterior mean of the validation points in $\mathcal{D}_V$ given the training data set $\mathcal{D}_T$ is

$$\bar{\boldsymbol{m}}(\boldsymbol{X}_V; \boldsymbol{\theta}) \triangleq \boldsymbol{K}(\boldsymbol{X}_V, \boldsymbol{X}_T; \boldsymbol{\theta}_h)\boldsymbol{z}_T \tag{7}$$

where $\boldsymbol{z}_T \triangleq \left[\boldsymbol{K}(\boldsymbol{X}_T, \boldsymbol{X}_T; \boldsymbol{\theta}_h) + \sigma_e^2 \boldsymbol{I}_n\right]^{-1} \boldsymbol{y}_T$ is our newly introduced auxiliary variable satisfying the nonlinear equality constraint:

$$\boldsymbol{C}(\boldsymbol{\theta})\boldsymbol{z}_T = \boldsymbol{y}_T. \tag{8}$$

Then optimization problem is then formulated as follows:

$$\boldsymbol{\theta}_{CV} = \arg\min_{\boldsymbol{\theta}} ||\boldsymbol{y}_V - \bar{\boldsymbol{m}}(\boldsymbol{X}_V; \boldsymbol{\theta})||_2^2. \tag{9}$$

Scheme 1: Hold-out Cross-Validation Based Scheme

We use the alternating direction method of multipliers (ADMM) for this optimization task. The augmented Lagrangian function is as follows:

$$L_\rho(\boldsymbol{\theta}, \boldsymbol{z}_T, \boldsymbol{\lambda}) \triangleq ||\boldsymbol{y}_V - \boldsymbol{K}_{VT}(\boldsymbol{\theta}_h)\boldsymbol{z}_T||_2^2 + \boldsymbol{\lambda}^T(\boldsymbol{C}(\boldsymbol{\theta})\boldsymbol{z}_T - \boldsymbol{y}_T) + \frac{\rho}{2}||\boldsymbol{C}(\boldsymbol{\theta})\boldsymbol{z}_T - \boldsymbol{y}_T||_2^2,$$

The complete method consists of a $\boldsymbol{\theta}$-minimization step, a $\boldsymbol{z}_T$ minimization step, and a closed-form dual variable update step. Concretely, in the $(\eta+1)$-th iteration,

$$\boldsymbol{\theta}^{\eta+1} = \arg\min_{\boldsymbol{\theta}} L_\rho(\boldsymbol{\theta}, \boldsymbol{z}_T^\eta, \boldsymbol{\lambda}^\eta), \tag{10a}$$

$$\boldsymbol{z}_T^{\eta+1} = \arg\min_{\boldsymbol{z}_T} L_\rho(\boldsymbol{\theta}^{\eta+1}, \boldsymbol{z}_T, \boldsymbol{\lambda}^\eta), \tag{10b}$$

$$\boldsymbol{\lambda}^{\eta+1} = \boldsymbol{\lambda}^\eta + \rho\left[\boldsymbol{C}(\boldsymbol{\theta}^{\eta+1})\boldsymbol{z}_T^{\eta+1} - \boldsymbol{y}_T\right]. \tag{10c}$$

Step 1: θ-minimization

We elaborate on the $\boldsymbol{\theta}$-minimization step in the first place. Note that $L_\rho(\boldsymbol{\theta}, \boldsymbol{z}_T^\eta, \boldsymbol{\lambda}^\eta)$ is often a non-convex function in terms of $\boldsymbol{\theta}$, and we solve it using GD type update as follows:

$$\begin{aligned} \boldsymbol{\theta}^{\eta+1} &= \boldsymbol{\theta}^\eta - \mu_1 \cdot \nabla_{\boldsymbol{\theta}} L_\rho(\boldsymbol{\theta}, \boldsymbol{z}_T^\eta, \boldsymbol{\lambda}^\eta) \\ &\equiv \boldsymbol{\theta}^\eta - \mu_1 \cdot \nabla_{\boldsymbol{\theta}} \boldsymbol{g}^{(\eta)}(\boldsymbol{\theta}) \mid_{\boldsymbol{\theta}=\boldsymbol{\theta}^\eta}, \end{aligned} \tag{11}$$

where μ_1 is a positive step size chosen via Armijo's rule, $\boldsymbol{g}^{(\eta)}(\boldsymbol{\theta})$ and its gradient are given in the appendix.

Step 2: z_T-minimization

Second, we elaborate on the z_T-minimization step. In the subproblem (10b), it is easy to verify that $L_\rho(\theta^{\eta+1}, z_T, \lambda^\eta)$ is a quadratic function of z_T. Minimizing $L_\rho(\theta^{\eta+1}, z_T, \lambda^\eta)$ with respect to z_T is equivalent to

$$\arg\min_{z_T} g^{(\eta)}(z_T) = (b^\eta)^T z_T + z_T^T \Sigma^\eta z_T, \tag{12}$$

where

$$b^\eta \triangleq C(\theta^{\eta+1})\lambda - \rho C(\theta^{\eta+1}) y_T - 2K_{VT}^T(\theta_h^{\eta+1}) y_V,$$
$$\Sigma^\eta \triangleq K_{VT}^T(\theta_h^{\eta+1}) K_{VT}(\theta_h^{\eta+1}) + \frac{\rho}{2} C^2(\theta^{\eta+1}).$$

we solve the quadratic minimization problem via conjugate gradient method (CGM).

Step 3: $\boldsymbol{\lambda}$-minimization

Lastly, the update of the Lagrange multipliers $\boldsymbol{\lambda}^{\eta+1}$ is conducted in light of (10c) after $\boldsymbol{\theta}^{\eta+1}$ and $\boldsymbol{z}_T^{\eta+1}$ are obtained.

$$\boldsymbol{\lambda}^{\eta+1} = \boldsymbol{\lambda}^{\eta} + \rho \left[\boldsymbol{C}(\boldsymbol{\theta}^{\eta+1}) \boldsymbol{z}_T^{\eta+1} - \boldsymbol{y}_T \right].$$

where the augmented Lagrangian parameter $\rho \geq 0$ is pre-selected.

Remark: The magnitude of ρ controls the descent speed and the convexity of the ADMM objective function. When a suitable ρ value is difficult to determine, one possible remedy as suggested in [Hong et al., 2016] is to use a smaller ρ' instead of ρ in (10c) for updating the dual variable.

Outline

1. Motivations
   - GP Basics and GP Regression
   - Hardcore Problem in GP Hyper-parameter Tuning
2. New Schemes
   - Hold-out Cross-Validation Based Scheme
   - K-fold Cross Validation Based Scheme
   - Complexity Analysis
3. Experimental Results
   - Synthetic Data
   - Real Atmospheric CO_2 Concentration Data
4. Outlook

Scheme 2: K-fold Cross Validation Based Scheme

HOCV can be seen as a special case of the more general K-fold CV with $K = 1$. We extend the HOCV scheme to K-fold CV, which is able to generate more robust result and exploit parallel computing.

A Naive Scheme: We can train GP hyper-parameter, $\hat{\theta}_k$, heuristically for every partition, $k = 1, 2, ..., K$, using the same routine in the HOCV based scheme, and average the results to get $\boldsymbol{\theta}_{CV} = 1/K \cdot \sum_{k=1}^{K} \hat{\boldsymbol{\theta}}_k$.

A Principled Scheme: Alternatively, we can formulate an optimization problem for the same purpose but with sound rationale, by introducing some local copies of $\boldsymbol{\theta}$ and solve the following linear equality-constrained optimization problem:

$$\begin{aligned} \boldsymbol{\theta}_{CV} = \arg \min_{\boldsymbol{\theta}_1,...,\boldsymbol{\theta}_K} & \sum_{k=1}^{K} l_k(\boldsymbol{\theta}_k) \\ \text{s.t.} \quad & \boldsymbol{\theta}_1 = \boldsymbol{\theta}_2 =, ..., = \boldsymbol{\theta}_K = \boldsymbol{z}, \end{aligned} \tag{13}$$

Outline

1. Motivations
   - GP Basics and GP Regression
   - Hardcore Problem in GP Hyper-parameter Tuning
2. New Schemes
   - Hold-out Cross-Validation Based Scheme
   - K-fold Cross Validation Based Scheme
   - Complexity Analysis
3. Experimental Results
   - Synthetic Data
   - Real Atmospheric CO_2 Concentration Data
4. Outlook

Complexity Analysis

HOCV based scheme: Updating one particular element of hyper-parameter (out of p) involves the computations of $\frac{\partial \boldsymbol{K}_{VT}(\boldsymbol{\theta}_h)}{\partial \boldsymbol{\theta}_i}\boldsymbol{z}_T^{\eta}$, $\boldsymbol{K}_{VT}(\boldsymbol{\theta}_h)\boldsymbol{z}_T^{\eta}$, $\frac{\partial \boldsymbol{K}_{TT}(\boldsymbol{\theta}_h)}{\partial \boldsymbol{\theta}_i}\boldsymbol{z}_T^{\eta}$, $\boldsymbol{K}_{TT}(\boldsymbol{\theta}_h)\boldsymbol{z}_T^{\eta}$ and some cheap vector inner products, with the computational complexity scales as $\mathcal{O}(n_v \cdot n_t + n_t^2) = \mathcal{O}(n \cdot n_t)$ for this step. Similarly, updating the auxiliary parameter $\boldsymbol{z}_T$ also scales as $\mathcal{O}(n_v \cdot n_t + n_t^2) = \mathcal{O}(n \cdot n_t)$. The third step involves only a closed-form update, whose complexity scales as $\mathcal{O}(n_t^2)$. Therefore, the overall computational complexity for running one complete ADMM iteration scales as $\mathcal{O}(p \cdot n \cdot n_t) \approx \mathcal{O}(n^2)$ for $p \ll n$, which is much lower than $\mathcal{O}(n^3)$.

K-fold CV based scheme: Each local computing unit updates a copy of the global variable, $\boldsymbol{\theta}_i$, requiring approximately $\mathcal{O}(n^2)$ complexity according to the above analysis. A central node performs certain consensus operation. The total complexity remains low for practical K and p values.

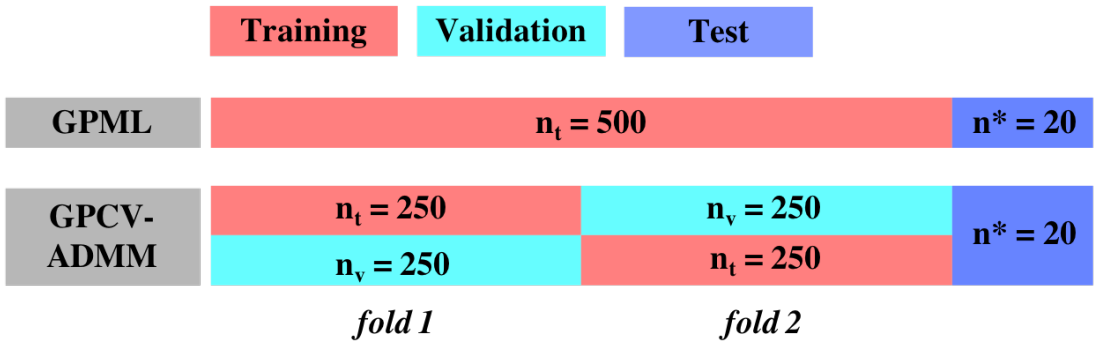
Outline

1. Motivations
   - GP Basics and GP Regression
   - Hardcore Problem in GP Hyper-parameter Tuning
2. New Schemes
   - Hold-out Cross-Validation Based Scheme
   - K-fold Cross Validation Based Scheme
   - Complexity Analysis
3. Experimental Results
   - Synthetic Data
   - Real Atmospheric CO_2 Concentration Data
4. Outlook

Experimental Results: Synthetic Data

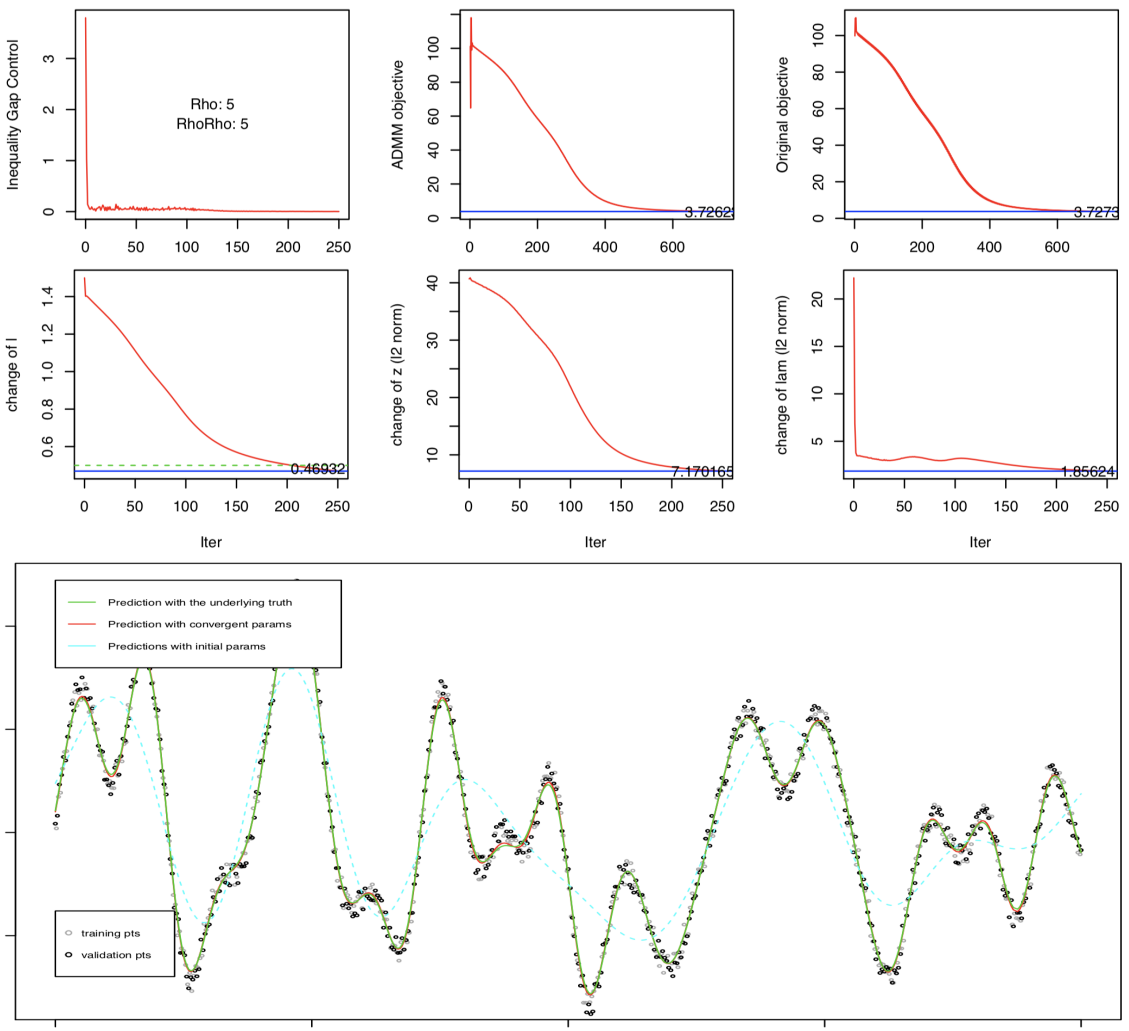
Comparison: GPCV-ADMM vs. GPML

Synthetic Data: We generated synthetic data sets from
(1) squared exponential (SE) kernel,
(2) local periodic (LP) kernel, and
(3) a composite SE plus LP (SE+LP).



For each kernel configuration with fixed sample size $n = 500, 1000, 2000$, (with the primary purpose to verify the reduced complexity), we ran 50 independent Monte Carlo trials.

Experimental Results: Synthetic Data



A Monitoring View in Training

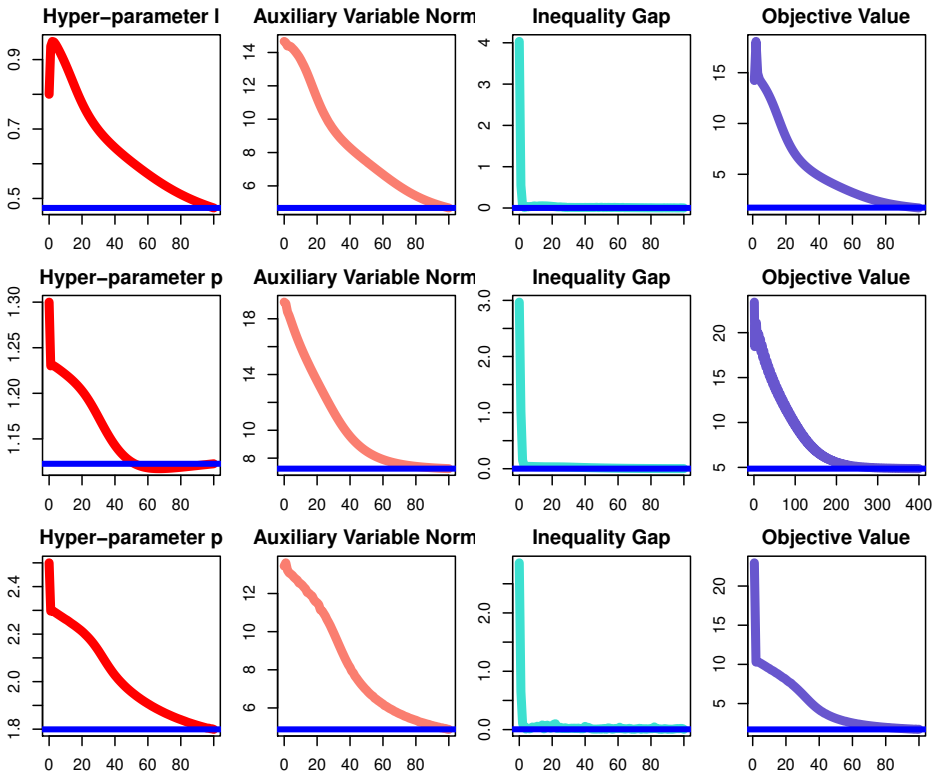
Experimental Results: Estimation Performance

Methods	GPCV-ADMM		GPML	
Kernel	Hyper-parameter Estimates (std)	Test MSE (std)	Hyper-parameter Estimates (std)	Test MSE (std)
SE	True hyper-parameter setting: $l = 0.5$			
(n=500)	[0.36 (0.054)]	0.34(0.044)	[0.52(0.018)]	0.36(0.055)
(n=1000)	[0.50 (0.044)]	0.3402(0.038)	[0.498 (0.017)]	0.3720(0.043)
(n=2000)	[0.503(0.017)]	0.35(0.026)	[0.53(0.02)]	0.37 (0.0267)
LP	True hyper-parameter setting: $l = 0.5, p = 1$			
(n=500)	[0.34(0.023),1.13(0.08)]	0.36(0.069)	[0.55(0.052),1.06(0.090)]	0.60(0.12)
(n=1000)	[0.39 (0.014), 1.06 (0.063)]	0.41(0.058)	[0.52(0.058),1.16(0.13)]	0.67(0.073)
(n=2000)	[0.44(0.082),1.19(0.008)]	0.51(0.02)	[0.53(0.013),1.02(0.019)]	0.52(0.048)
SE+LP	True hyper-parameter setting: $l_1 = 3, l_2 = 1, p = 2$			
(n=500)	[3.73(0.28),0.94(0.10),2.38(0.15)]	0.42(0.083)	[3.62(0.63), 1.08(0.21),2.38(0.86)]	0.46(0.07)
(n=1000)	[3.74(0.25),0.95(0.11),2.14(0.09)]	0.36(0.053)	[3.61(0.56),1.05(0.18),2.27(0.60)]	0.39(0.054)
(n=2000)	[3.94(0.075),0.99(0.12),2.1(0.08)]	0.59(0.2)	[3.69(0.39),1.09(0.13),2.13(0.29)]	0.61(0.19)

Table: Quantitative comparisons between the GPCV-ADMM and GPML methods across nine synthetic data sets (combining three kernels and three data lengths).

The results show that GPCV-ADMM hyper-parameter estimates are fairly close to both GPML estimates and the true values. According to the Monte Carlo simulation results, GPCV-ADMM estimation is more robust, with smaller sample standard deviation of both hyper-parameter estimator and the test MSE.

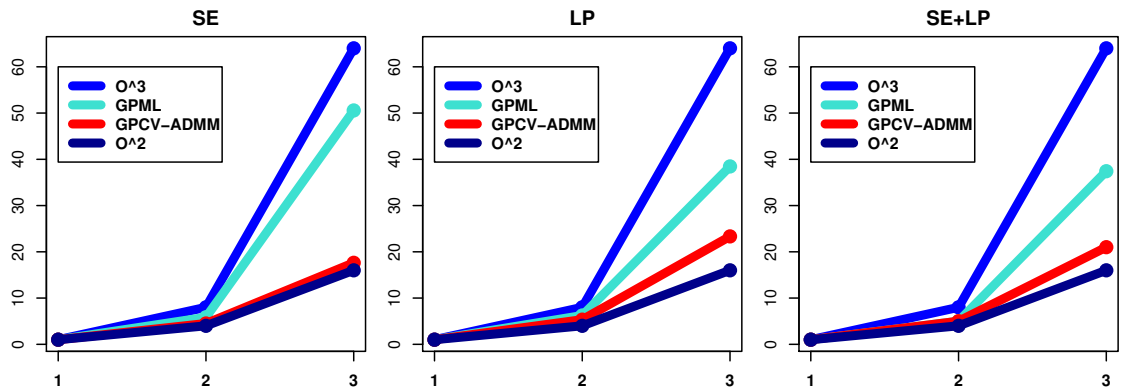
Experimental Results: Convergence Performance



Sample convergence plots of one particular Monte-Carlo trial (SE / LP / SE+LP)
(1) a representative GP hyper-parameter,
(2) the L_2 norm of the auxiliary variable, $||\boldsymbol{z}_T||_2$,
(3) the inequality gap, defined as $||\boldsymbol{z}_T - \left[\boldsymbol{K}(\boldsymbol{X}_T, \boldsymbol{X}_T; \boldsymbol{\theta}_h) + \sigma_e^2 \boldsymbol{I}_n\right]^{-1} \boldsymbol{y}_T||_2^2$, and
(4) the ADMM objective value, $L_\rho(\boldsymbol{\theta}^*, \boldsymbol{z}_T^*, \boldsymbol{\lambda}^*)$

Experimental Results: Computational Complexity

We want to test whether a quadratic increase in the computational time (CT) would be witnessed when we doubling the data size.
We compare the scaling factor criterion (defined as $\frac{CT_A/CT_B}{n_A/n_B}$, where B stands for our baseline data sets with $n = 500$, and A stands for data sets with greater sample size $n_A = 1000, 2000$). A greater scaling factor means that the algorithm scales worse when data size goes up.

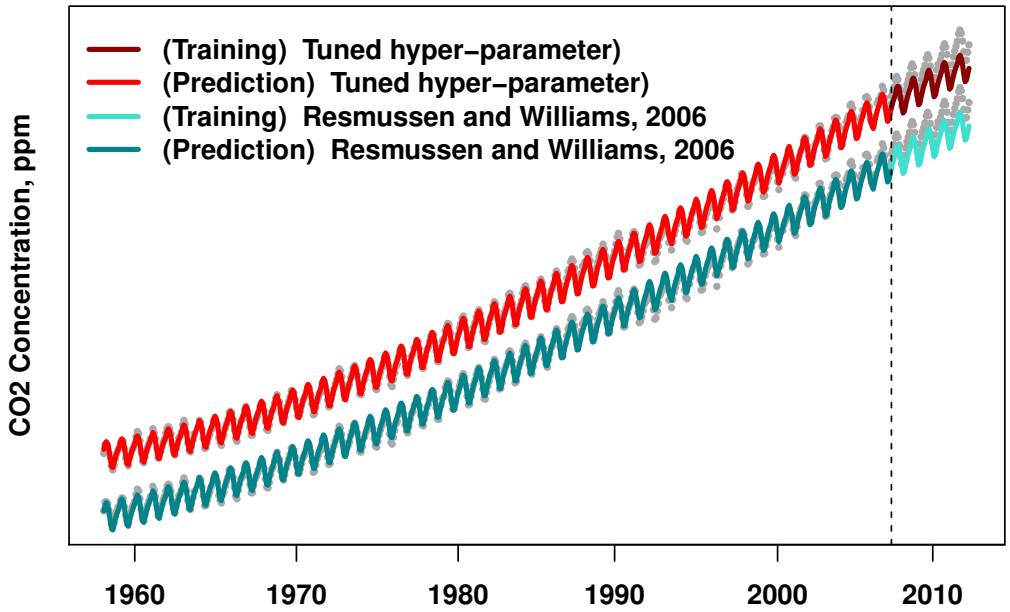


It is clear that the scaling factor of GPML is consistently larger than that of GPCV-ADMM. The quadratic increase of the CT of GPCV-ADMM and the cubic increase of the CT of GPML would become more apparent as data size increases.

Outline

1. Motivations
   - GP Basics and GP Regression
   - Hardcore Problem in GP Hyper-parameter Tuning
2. New Schemes
   - Hold-out Cross-Validation Based Scheme
   - K-fold Cross Validation Based Scheme
   - Complexity Analysis
3. Experimental Results
   - Synthetic Data
   - Real Atmospheric CO_2 Concentration Data
4. Outlook

Experimental Results: Atmospheric CO_2



Fitting and prediction performance test on CO_2 concentration data, tuned with SE+LP kernel setting.

- GPML reference estimates: $l_1 = 67$ years, $l_2 = 90$ years, $l_3 = 1.3$ standardized training MSE 0.95, standardized test MSE $= 1.408$.
- GPCV-ADMM estimates: $l_1 = 27$ years, $l_2 = 51$ years, $l_3 = 1.26$ standardized training MSE 0.85, standardized test MSE $= 1.307$.

Outline

1. Motivations
   - GP Basics and GP Regression
   - Hardcore Problem in GP Hyper-parameter Tuning
2. New Schemes
   - Hold-out Cross-Validation Based Scheme
   - K-fold Cross Validation Based Scheme
   - Complexity Analysis
3. Experimental Results
   - Synthetic Data
   - Real Atmospheric CO_2 Concentration Data
4. Outlook

Summary

We proposed two plain CV based GP hyper-parameter optimization schemes with reduced $O(n^2)$ computational complexity

- without any approximation like the existing low-complex schemes
- have robust performance
- extremely easy to implement
- can exploit multi-core processing of modern computing platforms

Outlook

More sophisticated designs, using for instance new-fashioned numerical search (e.g., ADAM), distributed data processing, and information consensus strategies, could bring in further acceleration and stability of our current vanilla version.

Thanks for your attention!

Appendix: Partial Derivatives and Gradient

The expression of $\boldsymbol{g}^{(\eta)}(\boldsymbol{\theta})$ used for updating GP hyper-parameter $\boldsymbol{\theta}$ in (12) is as follows,

$$\begin{aligned}\boldsymbol{g}^{(\eta)}(\boldsymbol{\theta}) = & -2 \cdot \boldsymbol{y}_V^T \boldsymbol{K}_{VT}(\boldsymbol{\theta}_h) \boldsymbol{z}_T^{\eta} \\ & + (\boldsymbol{z}_T^{\eta})^T \boldsymbol{K}_{VT}(\boldsymbol{\theta}_h)^T \boldsymbol{K}_{VT}(\boldsymbol{\theta}_h) \boldsymbol{z}_T^{\eta} \\ & + \boldsymbol{\lambda}^T \boldsymbol{K}_{TT}(\boldsymbol{\theta}_h) \boldsymbol{z}_T^{\eta} \\ & + \rho(\sigma_e^2 \boldsymbol{z}_T^{\eta} - \boldsymbol{y}_T)^T \boldsymbol{K}_{TT}(\boldsymbol{\theta}_h) \boldsymbol{z}_T^{\eta} \\ & + \frac{\rho}{2} (\boldsymbol{z}_T^{\eta})^T \boldsymbol{K}_{TT}(\boldsymbol{\theta}_h) \boldsymbol{K}_{TT}(\boldsymbol{\theta}_h) \boldsymbol{z}_T^{\eta}. \end{aligned} \tag{14}$$

Appendix: Partial Derivatives and Gradient

For each element of $\boldsymbol{\theta}$, denoted as $\boldsymbol{\theta}_i$, its partial derivative is computed as:

$$\begin{aligned}
\frac{\partial \boldsymbol{g}^{(\eta)}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i} &= -2 \cdot \boldsymbol{y}_V^T \frac{\partial \boldsymbol{K}_{VT}(\boldsymbol{\theta}_h)}{\partial \boldsymbol{\theta}_i} \boldsymbol{z}_T^\eta \\
&+ (\boldsymbol{z}_T^\eta)^T \frac{\partial \boldsymbol{K}_{VT}^T(\boldsymbol{\theta}_h)}{\partial \boldsymbol{\theta}_i} \boldsymbol{K}_{VT}(\boldsymbol{\theta}_h) \boldsymbol{z}_T^\eta \\
&+ (\boldsymbol{z}_T^\eta)^T \boldsymbol{K}_{VT}^T(\boldsymbol{\theta}_h) \frac{\partial \boldsymbol{K}_{VT}(\boldsymbol{\theta}_h)}{\partial \boldsymbol{\theta}_i} \boldsymbol{z}_T^\eta \\
&+ \boldsymbol{\lambda}^T \frac{\partial \boldsymbol{K}_{TT}(\boldsymbol{\theta}_h)}{\partial \boldsymbol{\theta}_i} \boldsymbol{z}_T^\eta \\
&+ \rho(\sigma_e^2 \boldsymbol{z}_T^\eta - \boldsymbol{y}_T)^T \frac{\partial \boldsymbol{K}_{TT}(\boldsymbol{\theta}_h)}{\partial \boldsymbol{\theta}_i} \boldsymbol{z}_T^\eta \\
&+ \frac{\rho}{2} (\boldsymbol{z}_T^\eta)^T \boldsymbol{K}_{TT}(\boldsymbol{\theta}_h) \frac{\partial \boldsymbol{K}_{TT}(\boldsymbol{\theta}_h)}{\partial \boldsymbol{\theta}_i} \boldsymbol{z}_T^\eta \\
&+ \frac{\rho}{2} (\boldsymbol{z}_T^\eta)^T \frac{\partial \boldsymbol{K}_{TT}(\boldsymbol{\theta}_h)}{\partial \boldsymbol{\theta}_i} \boldsymbol{K}_{TT}(\boldsymbol{\theta}_h) \boldsymbol{z}_T^\eta. \qquad (15)
\end{aligned}$$

Appendix: Expressions for the Selected Kernels

- **Squared Exponential (SE) Kernel** ($l = 0.5$)
  SE kernel is usually regarded as the default kernel for GP models, due to its great universality as well as many good properties such as infinite many derivatives. The lengthscale l in an SE kernel specifies the width of the kernel and thereby determines the smoothness of the functions in the model.

$$K_{se}(x, x') = \sigma^2 exp(-\frac{(x - x')^2}{2l^2}) \tag{16}$$

Appendix: Expressions for the Selected Kernels

- **Locally Periodic (LP) Kernel** ($l = 0.5, p = 1$)
  Periodicity is another important pattern that people always get interested, especially in modeling time series data. Most periodic kernel functions don't repeat themselves exactly. Therefore a local kernel with a periodic kernel, combined to form a locally periodic kernel, is considered to allow the shape of the repeating pattern to evolve over time.

$$K_{lp}(x, x') = \sigma^2 exp(-\frac{2sin^2(\pi|x - x'|/p)}{l^2}) exp(-\frac{(x - x')^2}{2l^2}) \quad (17)$$

Appendix: Expressions for the Selected Kernels

- **Composite (SE + LP) Kernel** ($l_1 = 4, l_2 = 3, p = 2$)
  One good thing about using kernel function is its flexibility in combining various kernel components, which allows multiplications and/or additions over different kernels to capture different features of the data. In our experiments, we added up one SE kernel and one LP kernel to model local periodicity with trend.

$$K_{se+lp}(x, x') = \sigma^2 exp(-\frac{(x - x')^2}{2l_1^2}) + \sigma^2 exp(-\frac{2sin^2(\pi|x - x'|/p)}{l_2^2}) exp(-\frac{(x - x')^2}{2l_2^2}) \quad (18)$$