Robust Gaussian Process for proton radius puzzle – example

The function RobustGP.R is the main function which implements our constrained GP approach to estimate the proton radius from electron scattering data. The methodology is very closely related to the nonparametric Bayes approach in (Zhou et al. 2019), with the only difference being a small tweak in the prior which nevertheless makes the computation significantly more efficient.

In this markdown file, we explain the input parameters as well as provide details on interpreting the model output.

Model description

We consider the model, for $i = 1, \ldots, n$,

$$y_i = f(x_i) + \epsilon_i, \quad f \in C_f,$$

where

$$C_f := \{f : [0,1] \to \mathbb{R} : f(0) = 1, f'(x) < 0, f''(x) > 0\}.$$

We add i.i.d. normal errors $\epsilon_i \sim N(0, \sigma^2)$ and σ^2 is unknown. For twice differentiable function f, we apply the following approximation

$$f(x) \approx f(0) + f'(0)\, x + \sum_{j=0}^{N} f''(u_j)\phi_j(x),$$

with fixed knots u_j and suitable basis functions ϕ_j for $j = 0, \ldots, n$. Letting $\xi_0 = f(0), \xi_1 = f'(0), \xi_{j+2} = f''(u_j), j = 0, \ldots, N$, the model turns into

$$y_i = \xi_0 + \xi_1\, x_i + \sum_{j=0}^{N} \xi_{j+2}\, \phi_j(x_i) + \epsilon_i,$$

and $\xi_0 = f(0)$, $\xi_1 = f'(0)$, $\xi_{j+2} = f''(u_j)$, $j = 0, \ldots, n$ are unknown parameters.

As for constraints $f \in C_f$, we find the equivalent constraints on ξs,

$f \in C_f$ if and only if $\xi = (\xi_1, \ldots, \xi_{N+2})' \in C_\xi$, where

$$C_\xi = \left\{ \xi \in \mathbb{R}^{N+2} : \xi_1 + \sum_{j=0}^{N} c_j\ \xi_{j+2} \le 0, \xi_{j+2} \ge 0,\ j = 0, \ldots, N \right\}.$$

We consider the following models:

cGP: $\xi_0 = 1$ and $\xi \in C_\xi$; $\quad$ c_1GP: $\xi_0 \in \mathbb{R}$ and $\xi \in C_\xi$.

Prior specifications

We start from assigning an unconstrained Gaussian process prior on the second derivative of the true function

$$f'' \sim GP(0, \tau^2 K)$$

with matern kernel K and a scaling parameter τ^2. Note that such GP prior on the true f'' induces a multivariate normal prior on the parameters ξ's and we restrict ξ's to C_ξ to satisfy the constraints. Eventually, the prior becomes

$$\xi_1, \xi_2, \ldots, \xi_{N+2} \sim N(0, \tau^2 \Gamma) \cdot \mathbb{1}_{C_\xi},$$

the covariance matrix Γ can be calculated analytically.

For c_1GP model, we assign an uniform prior on the normalizing factor $\xi_0 \sim U(1-\sigma_0, 1+\sigma_0)$. We consider σ_0 to be known and fix the value ahead. For cGP model, we fix $\xi_0 = 1$.

We consider obejective priors for τ^2 and σ^2,

$$p(\tau^2) \propto 1/\tau^2, \quad p(\sigma^2) \propto 1/\sigma^2.$$

Model implementation in R

First install and call the packages (to install use install.packages("package_name"))

```
library(MASS)
library(FastGP)
library(mcmcplots)
library(fields)
library(HDInterval)
library(mcmcse)
library(ggplot2)
```

Source the external functions

```
source('RobustGP.R')
source('ESS_pro.R')
source('ESS_joint.R')
source('function_pro.R')
```

Input variables

To implement the algorithm, first we need to input the data and specify the hyperparaters.

Y: input G_E

```
 Y = read.table("YF0.txt")
```

X: input Q^2

```
X = read.table("xvals.txt")
```

n: sample size

```
n = length(as.numeric(as.matrix(X)))
```

sig: If the input G_E are noiseless, we add i.i.d. normal noise ϵ_i with noise level σ. We fix σ as the sample mean of measured errors from Mainz data (or other reasonable values). If not adding errors, ignore this step. (The value of σ is only to add random errors to the true data, in the algorithm we consider the true σ as unknown and update it.)

```
sig.pro = 0.00136
```

nu: smoothness parameter ν for Matern kernel, default is 0.5, suggested values are between 0.5 and 1.

```
nu = 0.5
```

N: number of basis function N in $\{n/8, n/4, n/3, n/2, n\}$ or any number no larger than sample size n. (In general, less the number of the basis functions, smaller the bias; it may cause larger variance.)

```
N = ceiling(n/8)-1
```

l: length-scale parameter ℓ of Matern kernel. If the values of Q^2 are mostly small however few points are large (e.g. consider the whole Mainz data), smaller $\ell < 1$ is suggested; if Q^2 are all small, larger $1 < \ell \leq 10$ is suggested.

```
l = 10
```

sig0: hyperparameter $0 < \sigma_0 \leq 0.001$ of the normalizing factor $\xi_0 \sim U(1-\sigma_0, 1+\sigma_0)$. It determines how far ξ_0 can float around 1.

```
sig0 = 0.0005
```

prob: a scalar [0, 1] specifying the probability within the credible interval, default is 0.95.

```
prob = 0.95
```

sseed: value of a random seed; used to fix the generated random numbers for the purposes of replications

```
sseed = 67
```

mcmc: number of MCMC iterations

```
mcmc = 10000
```

brn: number of burn-in samples; if *brn = 4000*, then the first *4000* samples were discarded.

```
brn = 4000
```

thin: any integer ≥ 1; if *thin = 10*, then every 10^{th} sample is saved as posterior sample.

```
thin = 10
```

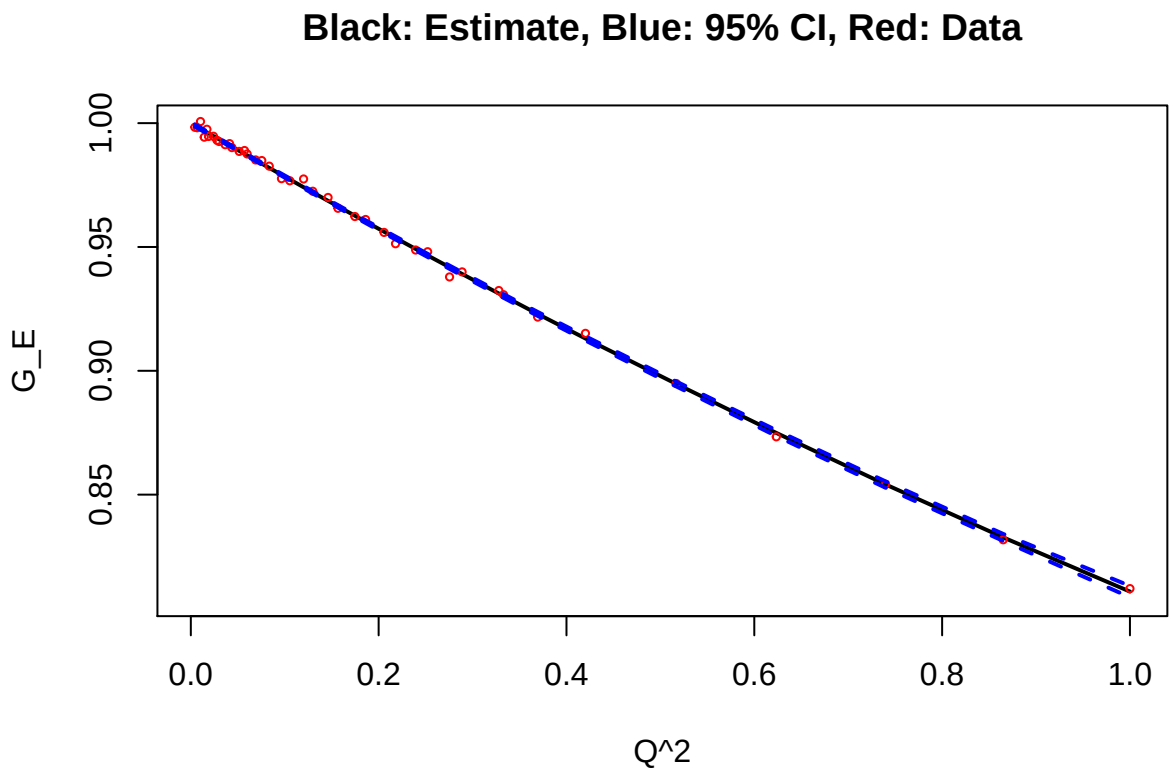
return.plot: logical; if true a plot of estimate with 95% CI is returned, default is `TRUE`.

return.traplot: logical; if true traceplots are returned; default is `TRUE`.

Running the main algorithm

```
# c1GP
Results = RobustGP(Y,X,sig=sig.pro,nu=nu,N=N,l=l,sig0=sig0,prob=prob,sseed=sseed,
                   mcmc=mcmc,brn=brn,thin=thin,return.plot = TRUE,return.traplot = FALSE)
```

```
## [1] "MCMC sample draws:"
## [1] 2000
## [1] 4000
## [1] 6000
## [1] 8000
## [1] 10000
## [1] 12000
## [1] 14000
```




```
#cGP (set xi0.fix=1)
#Results = RobustGP(Y,X,sig=sig.pro,nu=nu,N=N,l=l,sig0=sig0,prob=prob,sseed=sseed,
#                    mcmc=mcmc,brn=brn,thin=thin,xi0.fix=1,return.plot = TRUE,return.traplot = FALSE)
```

Outputs

Results: a list object including the following:

Estimated Radius: Posterior median:

```
r_est = Results[[1]]
r_est
```

```
## [1] 0.8437903
```

95% confidence interval of the proton radius:

```
r_CI = Results[[2]]
r_CI
```

```
##      2.5%     97.5%
## 0.8326667 0.8547434
```

MCMC samples of the proton radius:

```
r_sam = Results[[3]]
```

Data of G_E and Q^2, saved for replications

```
data = Results[[4]]
```

Posterior samples on $\xi, \xi_0, \xi_1, \tau^2, \sigma^2$ and posterior mean as well as 95% CI of G_E estimates:

```
post.sam = Results[[5]]
```

Watanabe's AIC (a model choice criterion) for the current choice of hyperparameters:

```
model.check = Results[[6]]
WAIC = model.check[[1]][1]
WAIC
```

```
## [[1]]
## [1] -416.0639
```

When comparing different hyperparameter choices, pick the one with the smallest WAIC.

The shortest set (interval in this case) with "prob"% posterior probability:

```
HD_rp = Results[[7]]
HD_rp
```

```
##     lower     upper
## 0.8343487 0.8562917
## attr(,"credMass")
## [1] 0.95
```

Estimate effective sample size for the MCMC path:

```
r_Ess = Results[[8]]
r_Ess
```

```
## [1] 191.1444
```

The number of samples > 500 of the chain is suggested in the proton problem.

Monte Carlo standard error:

```
r_mcse = Results[[9]]
```

The estimate of expectation of proton radius:

```
r_mcse[1]
```

```
## $est
## [1] 0.8437765
```

Standard error of MCMC samples comparing to the standard error of the data (using `sd()`, defined as $\sqrt{\sum(x_i - \bar{x})^2/(n-1)}$ with sample mean $\bar{x}$ and sample size n). MCMC iteration stops when the Monte Carlo standard error is small compared to the variability in the target distribution.

```
r_mcse[2]
```

```
## $se
## [1] 0.0004344675
```

```
sd(as.numeric(as.matrix(Y)))
```

```
## [1] 0.04679808
```

References

Zhou, Shuang, P. Giulani, J. Piekarewicz, Anirban Bhattacharya, and Debdeep Pati. 2019. "Reexamining the Proton-Radius Problem Using Constrained Gaussian Processes." *Phys. Rev. C* 99 (5). American Physical Society: 055202. doi:10.1103/PhysRevC.99.055202.