Example on Bayesian Spectral Analysis Generalized Linear Regression

Statlab2

2017-08-04

Preliminaries

To use `bsamGP`, you need to first install the package. Use RGui menu "Package/Install package(s) from local zip files…" or R command "install.packages".

You then load the `bsamGP` package using the `library` or `require` function:

```
library(bsamGP)
```

This needs to be done every time you start R.

To get help on the functions in R (and in `bsamGP`), use `help()` or `?`. For example, to view the help file for the `gbsar` function, type one of the following:

```
help(gbsar) # ?gbsar
```

Bayesian Spectral Analysis Generalized Linear Regression (GBSAR)

Let's now proceed to the monotone convex function estimation via the following GBSAR probit model.

$$y_i = I(z_i > 0)z_i = \boldsymbol{w}_i^T\boldsymbol{\beta} + \sum_{k=1}^{K} f_k(x_{i,k}) + \epsilon_i, \quad \epsilon_i \overset{iid}{\sim} \mathcal{N}(0,1). \tag{1}$$

where $\boldsymbol{w}_i$ and $\boldsymbol{\beta}$ are $p+1$–dimensional vectors of covariates and coefficients; f is an unknown function of the scalar x_i estimated by

$$\begin{aligned} f(x) &= \delta \sum_{j=0}^{\infty}\sum_{k=0}^{\infty} \theta_j\theta_k\varphi_{j,k}^b(x) + \alpha(x-0.5) \\ \varphi_{j,k}^b(x) &= \int_0^x\int_0^s \varphi_j(t)\varphi_k(t)dt\,ds - \int_0^1\int_0^x\int_0^s \varphi_j(t)\varphi_k(t)dt\,ds\,dx. \end{aligned}$$

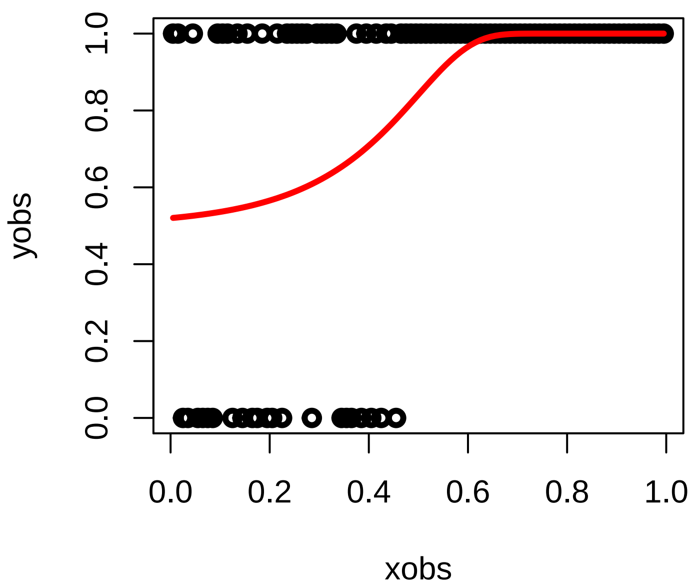
Data generation

We will consider estimating the following monotone convex function where $x \in [0,1]$.

$$f(x) = \exp(6x-3)$$

```
set.seed(1)
ftn = function(x) exp(6*x - 3)
n = 100
xobs = (2 * (1:n)-1)/(2 * n)
```

```
yobs = rbinom(n,1,pnorm(ftn(xobs)))
plot(xobs,yobs,lwd=3)
lines(xobs,pnorm(ftn(xobs)),lwd=3,col=2)
```



Model Fitting

To fit GBSAR model, we first set up the MCMC parameters, the number of basis and prior information for spectral coefficients.

```
# MCMC parameters
nblow0 = 1000; # Initialization period for adpative metropolis
nblow = 70000; # Number of MCMC in transition period
smcmc = 3000; # Number of MCMC for analysis
nskip = 50; # Number of MCMC to skip after nblow
ndisp = 1000; # number of saved draws to be displayed on screen
maxmodmet = 5; # Maximum number of times to modify metropolis
# Prior information
iflagprior = 0; # 1 = Lasso Smoother, 0 = T Smoother
nbasis = 50; # number of cosine basis functions @
```

To generate a posterior sample for the Bayesian semiparametric regression model, use the function `gbsar`.

```
# Fit the model
fout = gbsar(y=yobs, x=xobs, w=NULL, shape='IncreasingConvex',
           xmin=0, xmax=1,nbasis=nbasis, family="bernoulli", link="probit",
           mcmc=list(maxmodmet=maxmodmet,nblow0=nblow0,nblow=nblow,smcmc=smcmc,nskip=nskip,ndisp=ndisp)
           prior=list(iflagprior=iflagprior))
```

```
## Initializing MCMC parameters ...
```

```
## Burnin ...
## function[1]: pmet = 0.4626
## Main iterations ...
## MCMC draws 1000 of 3000 (CPU time: 13.969 s)
## MCMC draws 2000 of 3000 (CPU time: 19.672 s)
## MCMC draws 3000 of 3000 (CPU time: 25.656 s)
## function[1]: pmet = 0.5124
```

The output returns `bsam` class object. The `print` function on `bsam` object summarizes the fit.

```
print(fout)
```

```
##
## Call:
## gbsar.default(y = yobs, w = NULL, x = xobs, xmin = 0, xmax = 1,
##     family = "bernoulli", link = "probit", nbasis = nbasis, mcmc = list(maxmodmet = maxmodmet,
##         nblow0 = nblow0, nblow = nblow, smcmc = smcmc, nskip = nskip,
##         ndisp = ndisp), prior = list(iflagprior = iflagprior),
##     shape = "IncreasingConvex")
##
## Bayesian Spectral Analysis Probit Regression (Alber-Chib)
##
## Model:
## Y ~ Ber(p), E(Y) = p
## p = F(w'beta + sum_k f_k(x_k)), F : normal cdf
## xmin < x < xmax
## Normalize f: int_a^b f(x)dx = 0
## beta has the Intercept
##
## Define
## Z(x) = sum_j=0^J theta_j*phi_j(x)
## g^a(x) = int_0^x |Z(s)|^2 dx
##        = theta' Phi^a(x)*theta
## where Phi^a(x) is matrix with
## phi^a_{j,k}(x) = int_0^x phi_j(s) phi_k(s) ds in (j,k)
## g^b(x)   = int_0^x g^a(s) ds
##         = theta' Phi^b(x)*theta
## where Phi^b(x) is a matrix with
## phi^b_{j,k}(x) = int_0^x phi^a_{j,k}(s) ds
##
## *************************************************************************************
##
## Number of nonparametric components =  1
##
## Model for f
## Removed theta_0^2 from f but kept theta_0*theta_j
## Increasing convex f
## f(x) = g^b(x) +  alpha*(x-xmid)
## alpha > 0
## Model for f includes linear term alpha
##
## *************************************************************************************
##
## Cosine basis
## xrange   = xmax-xmin
```

```
## phi_0(x) = 1/sqrt(xmax-xmin) for xmin < x < xmax
## phi_k(x) = sqrt(2/(xmax-xmin))*cos(pi*k*(x-xmin)/(xmax-xmin))
##
## Priors:
## beta    ~ N(b0,B0)
## alpha   ~ N(m0,v0)I(delta*alpha>0)
## theta_0 ~ N(0,v0)I(theta_0 > 0)
## theta_k ~ N(0,tau^2*exp(-gamma*k)
##
## Smoothing parameters tau and gamma
## Choice of two priors for tau2
## 1.) T Smoother: tau2 ~ IG(r0/2,s0/2)
## gamma ~ Exp(w0)
##
## Note: posterior of tau and gamma have banana contours
## zeta = ln(tau2) - kbar*gamma is less dependent with gamma or log(gamma)
##
## *********************************************************************************
##
## Number of Cosine basis functions     =  50
## Number of observations               =  100
## Number of covariates (no intercept)  =  0
##
## MCMC transition draws                =  70000
## MCMC draws saved for estimation      =  3000
## Save every nskip draws               =  50
## MCMC draws total                     =  220000
##
## Function =  1
## Proportion of Theta Accepted after burn-in    =  0.5123533
##
## Log Integrated Likelihood
## LIL Gelfand & Dey              =  -44.6706
## LIL Newton & Raftery (biased)  =  -37.2573
##
## beta
## const is Y intercept
##         PostM   PostStd PostM/STD
## const 1.75841 0.5569496  3.157215
##
## *********************************************************************************
##
## Function = 1
## Increasing convex function
##
## Linear term alpha in x for constrainted f
## Posterior mean   of alpha =  1.864053
## Posterior stddev of alpha =  1.254771
##
## theta_k ~ N(0,tau2*exp(-gamma*k))
##
## Tau
## PostM   PostS
## 1.019212 0.5970096
```

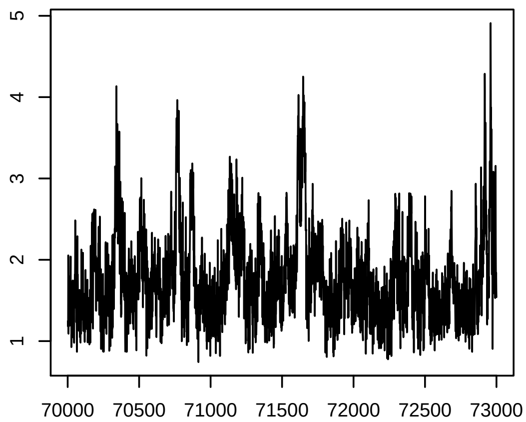
```
##
## Gamma
##      PostM      PostS
## 0.48688668 0.07947418
##
## Zeta = ln(tau2) - mean(k)*gamma
##      PostM      PostS
## -12.381860   1.867457
##
## Cosine Basis weights theta
##     PostMean PostSTD    Ratio
## T0   2.36850 1.29020  1.83575
## T1  -0.19077 0.78766 -0.24220
## T2  -0.22508 0.69148 -0.32550
## T3  -0.14955 0.54988 -0.27197
## T4   0.07925 0.47608  0.16645
## T5   0.06994 0.31675  0.22080
## T6  -0.00279 0.24154 -0.01154
## T7   0.00596 0.20904  0.02850
## T8  -0.00964 0.15661 -0.06158
## T9   0.02542 0.11691  0.21745
## T10  0.00123 0.09045  0.01365
## T11  0.01205 0.07740  0.15568
## T12 -0.00296 0.05752 -0.05140
## T13  0.00321 0.04902  0.06541
## T14  0.00163 0.03912  0.04176
## T15  0.00051 0.02952  0.01722
## T16 -0.00110 0.02606 -0.04221
## T17 -0.00097 0.01738 -0.05608
## T18 -0.00130 0.01632 -0.07967
## T19 -0.00130 0.01258 -0.10301
## T20 -0.00035 0.00920 -0.03779
## T21 -0.00135 0.00849 -0.15869
## T22  0.00108 0.00726  0.14862
## T23  0.00012 0.00541  0.02182
## T24 -0.00091 0.00502 -0.18054
## T25 -0.00014 0.00358 -0.03975
## T26 -0.00037 0.00288 -0.13006
## T27  0.00050 0.00239  0.20898
## T28  0.00021 0.00199  0.10316
## T29 -0.00003 0.00137 -0.01918
## T30 -0.00016 0.00120 -0.13610
## T31 -0.00002 0.00094 -0.02527
## T32  0.00003 0.00093  0.02784
## T33  0.00003 0.00074  0.04073
## T34  0.00001 0.00059  0.01193
## T35  0.00002 0.00046  0.04454
## T36  0.00001 0.00039  0.02927
## T37  0.00002 0.00030  0.05733
## T38 -0.00005 0.00024 -0.19529
## T39  0.00000 0.00022  0.00400
## T40 -0.00002 0.00016 -0.11331
## T41  0.00000 0.00014  0.03180
## T42 -0.00001 0.00010 -0.06146
```

```
## T43  0.00000 0.00010  0.03589
## T44  0.00001 0.00008  0.08037
## T45  0.00000 0.00007 -0.04639
## T46  0.00000 0.00005 -0.00945
## T47  0.00000 0.00004 -0.06807
## T48  0.00000 0.00004  0.09040
## T49  0.00000 0.00004  0.08789
## T50  0.00000 0.00002 -0.16876
## ***************************************************************************************
```

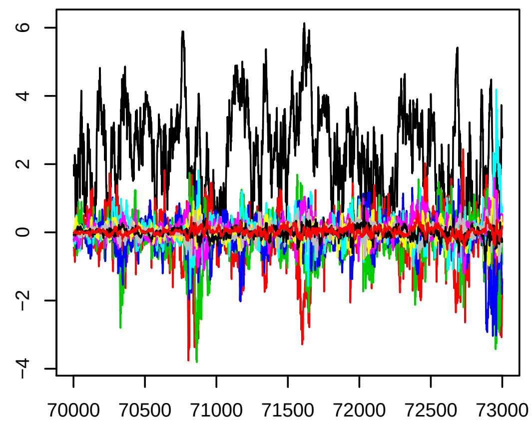
The `plot` function on `bsam` object returns traceplots of each parameters for diagnostics.

```
plot(fout)
```

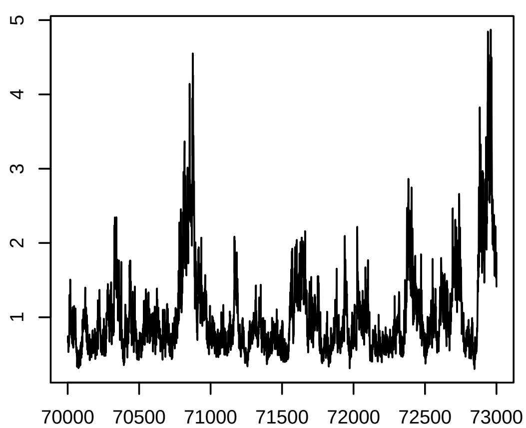
Beta



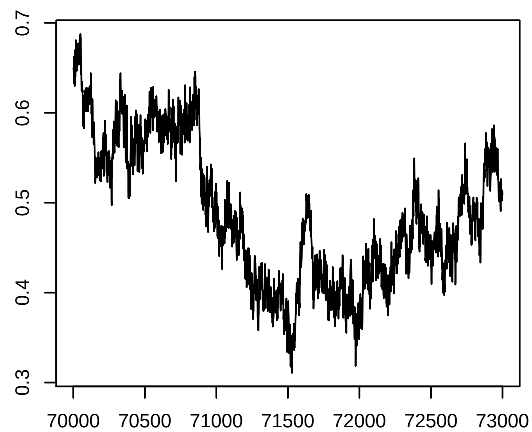
Function 1: Theta



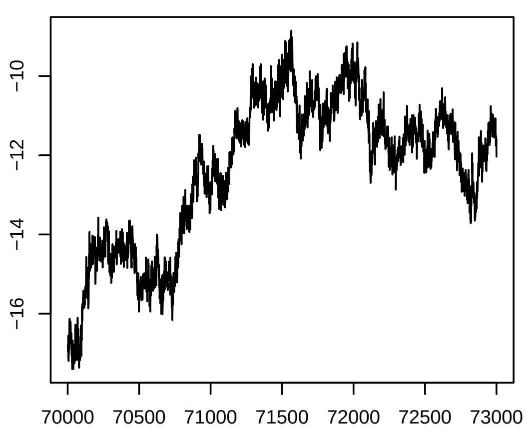
Function 1: Tau



Function 1: Gamma

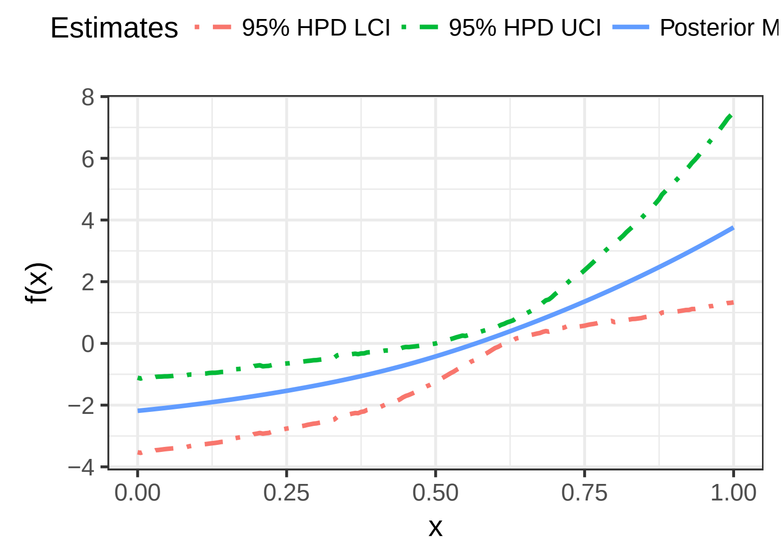


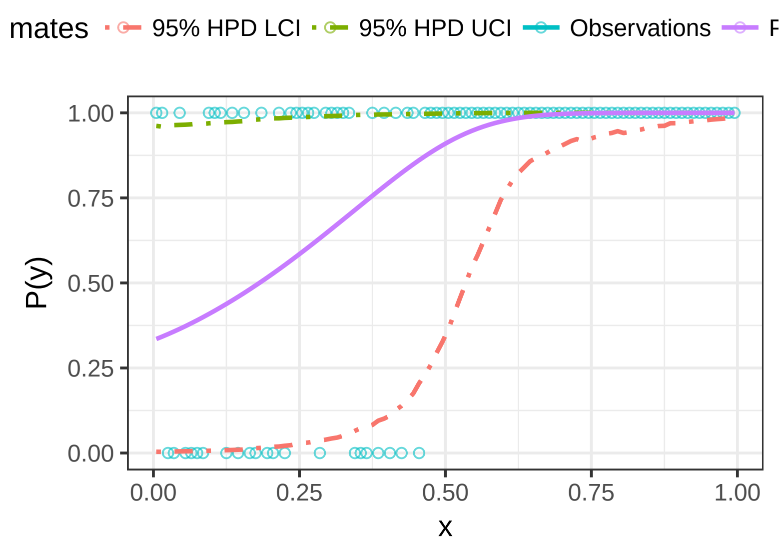
Function 1: Zeta



We may visualize fitted mean curve and 95% highest posterior density (HPD) interval with `plot` function on fitted object from `fitted` method.

```
fit = fitted(fout)
plot(fit)
```





For more detailed examples and real data applications, see Jo, S., Choi, T., Park, B., & Lenk, P. (2017) "bsamGP: An R package for Bayesian Spectral Analysis Models using Gaussian Process Priors", *Preprint* .