Package 'SPQR'

October 12, 2022

Type Package

Title Semi-Parametric Quantile Regression

Version 0.1.0

Date 2022-4-26

Archs x64

Description Methods for flexible estimation of conditional density and quantile function, as well as model agnostic tools for analyzing quantile covariate effect and variable importance. The estimation method implements the semi-parametric quantile regression model described in Xu and Reich (2021) <doi:10.1111/biom.13576>, and the model agnostic tools extend accumulative local effects (ALE) to quantile regression setting.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.1.2

Depends R (>= 3.6)

Imports Rcpp (>= 1.0.8), stats, torch, splines2, ggplot2, loo, progress, progressr, interp, RColorBrewer, yaImpute, coro (>= 1.0.2)

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

SystemRequirements C++11

Author Steven Xu [aut, cre], Reetam Majumder [aut], Brian Reich [ctb]

Maintainer Steven Xu <sgxu@ncsu.edu>

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/stevengxu/SPQR

BugReports https://github.com/stevengxu/SPQR/issues

Repository CRAN

Date/Publication 2022-05-02 20:02:03 UTC

R topics documented:

`autoplot.SPQR` *autoplot method for class* `SPQR`

Description

The function calls one of the following functions: `plotEstimator()`, `plotGOF()`, `plotMCMCtrace()`, `plotQALE()`, `plotQVI()`

Usage

```
## S3 method for class 'SPQR'
autoplot(object, output = c("GOF", "estimator", "trace", "QALE", "QVI"), ...)
```

Arguments

`object`	An object of class `SPQR`.
`output`	A character indicating the type of plot to be returned.

- `"GOF"`: goodness of fit test by comparing the quantiles of probability integral transform (PIT) to that of uniform distribution.
- `"estimator"`: visualization of various estimates, including probability density function (PDF), cumulative density function (CDF) and quantile function (QF).
- `"trace"`: diagnostic trace plots for SPQR fitted with `method = "MCMC"`.
- `"QALE"`: quantile accumulative local effects (ALE) for visualizing covariate effects on predicted quantiles.

- "QVI": quantile variable importance comparison.

... arguments passed into specific plot function, see plotEstimator(), plotGOF(), plotMCMCtrace(), plotQALE() or plotQVI() for required arguments.

Value

a ggplot object

Examples

```
set.seed(919)
n <- 200
X <- rbinom(n, 1, 0.5)
Y <- rnorm(n, X, 0.8)
control <- list(iter = 200, warmup = 150, thin = 1)
fit <- SPQR(X = X, Y = Y, method = "MCMC", control = control,
            normalize = TRUE, verbose = FALSE)

## Goodness-of-fit test
autoplot(fit, output = "GOF")
```

---

coef.SPQR *coef method for class* SPQR

Description

Computes the estimated spline coefficients of a SPQR class object

Usage

```
## S3 method for class 'SPQR'
coef(object, X, ...)
```

Arguments

object	An object of class SPQR.
X	The covariate vector/matrix for which the coefficient is calculated.
...	Other arguments.

Value

A NROW(X) by K matrix containing values of the estimated coefficient, where K is the number of basis functions.

Examples

```
set.seed(919)
n <- 200
X <- rbinom(n, 1, 0.5)
Y <- rnorm(n, X, 0.8)
control <- list(iter = 200, warmup = 150, thin = 1)
fit <- SPQR(X = X, Y = Y, method = "MCMC", control = control,
            normalize = TRUE, verbose = FALSE)
coef(fit, X = 0)
```

`createFolds.SPQR` *generate cross-validation folds*

Description

Helper function to generate cross-validation folds that can be used by `cv.SPQR`.

Usage

```
createFolds.SPQR(Y, nfold, stratified = FALSE)
```

Arguments

`Y`	The response vector.
`nfold`	The number of cross-validation folds.
`stratified`	If `TRUE`, stratified folds based on quantiles of `Y` are generated.

Value

A list of size `nfold` containing indices of the observations for each fold.

Examples

```
set.seed(919)
n <- 1000
X <- rbinom(n, 1, 0.5)
Y <- rnorm(n, X, 0.8)
folds <- createFolds.SPQR(Y, nfold = 5)
```

cv.SPQR *cross-validation for SPQR estimator*

Description

Fits SPQR using either MLE or MAP method and computes K-fold cross-validation error based on pre-computed folds.

Usage

```
cv.SPQR(
  folds,
  X,
  Y,
  n.knots = 10,
  n.hidden = 10,
  activation = c("tanh", "relu", "sigmoid"),
  method = c("MLE", "MAP", "MCMC"),
  prior = c("ARD", "GP", "GSM"),
  hyperpar = list(),
  control = list(),
  normalize = FALSE,
  verbose = TRUE,
  seed = NULL,
  ...
)
```

Arguments

folds	A list of CV folds, possibly that generated from createFolds.SPQR().
X	The covariate matrix (without intercept column)
Y	The response vector.
n.knots	The number of basis functions. Default: 10.
n.hidden	A vector specifying the number of hidden neurons in each hidden layer. Default: 10.
activation	The hidden layer activation. Either "tanh" (default) or "relu".
method	Method for estimating SPQR. One of "MLE", "MAP" (default) or "MCMC".
prior	The prior model for variance hyperparameters. One of "GP", "ARD" (default) or "GSM".
hyperpar	A list of named hyper-prior hyperparameters to use instead of the default values, including a_lambda, b_lambda, a_sigma and b_sigma. The default value is 0.001 for all four hyperparameters.
control	A list of named and method-dependent parameters that allows finer control of the behavior of the computational approaches. 1. Parameters for MLE and MAP methods

- use.GPU If TRUE GPU computing will be used if torch::cuda_is_available() returns TRUE. Default: FALSE.
- lr The learning rate used by the Adam optimizer, i.e., torch::optim_adam().
- dropout A length two vector specifying the dropout probabilities in the input and hidden layers respectively. The default is c(0,0) indicating no dropout.
- batchnorm If TRUE batch normalization will be used after each hidden activation.
- epochs The number of passes of the entire training dataset in gradient descent optimization. If early.stopping.epochs is used then this is the maximum number of passes. Default: 200.
- batch.size The size of mini batches for gradient calculation. Default: 128.
- valid.pct The fraction of data used as validation set. Default: 0.2.
- early.stopping.epochs The number of epochs before stopping if the validation loss does not decrease. Default: 10.
- print.every.epochs The number of epochs before next training progress in printed. Default: 10.
- save.path The path to save the fitted torch model. By default a folder named "SPQR_model" is created in the current working directory to store the model.
- save.name The name of the file to save the fitted torch model. Default is "SPQR.model.pt".

2. Parameters for MCMC method

These parameters are similar to those in rstan::stan(). Detailed explanations can be found in the Stan reference manual.

- algorithm The sampling algorithm; "HMC": Hamiltonian Monte Carlo with dual-averaging, "NUTS": No-U-Turn sampler (default).
- iter The number of MCMC iterations (including warmup). Default: 2000.
- warmup The number of warm-up/burn-in iterations for step-size and mass matrix adaptation. Default: 500.
- thin The number of iterations before saving next post-warmup samples. Default: 1.
- stepsize The discretization interval/step-size ϵ of leap-frog integrator. Default is NULL which indicates that it will be adaptively selected during warm-up iterations.
- metric The type of mass matrix; "unit": diagonal matrix of ones, "diag": diagonal matrix with positive diagonal entries estimated during warmup iterations (default), "dense": a dense, symmetric positive definite matrix with entries estimated during warm-up iterations.
- delta The target Metropolis acceptance rate. Default: 0.9.
- max.treedepth The maximum tree depth in NUTS. Default: 6.
- int.time The integration time in HMC. The number of leap-frog steps is calculated as $L_\epsilon = \lfloor t/\epsilon \rfloor$. Default: 0.3.

normalize If TRUE, all covariates will be normalized to take values between [0,1].

verbose	If TRUE (default), training progress will be printed.
seed	Random number generation seed.
...	other parameters to pass to control.

Value

control	the list of all control parameters.
cve	the cross-validation error.
folds	the CV folds.

See Also

createFolds.SPQR()

Examples

```
set.seed(919)
n <- 200
X <- rbinom(n, 1, 0.5)
Y <- rnorm(n, X, 0.8)
folds <- createFolds.SPQR(Y, nfold = 5)
## compute 5-fold CV error
cv.out <- cv.SPQR(folds=folds, X=X, Y=Y, method="MLE",
                  normalize = TRUE, verbose = FALSE)
```

---

load.SPQR *load saved SPQR model*

Description

Load saved SPQR model from a designated path. The function first loads the .SPQR file that stores the SPQR object. It then checks whether the SPQR model is fitted with method = "MCMC". If not, it also loads the .pt file storing the torch model with the same name and attach it to the SPQR object.

Usage

```
load.SPQR(name = stop("`name` must be specified"), path = NULL)
```

Arguments

name	The name of the saved object excluding extension.
path	The path to look for the saved object. Default is the current working directory.

Value

An object of class SPQR.

Examples

```
set.seed(919)
n <- 200
X <- rbinom(n, 1, 0.5)
Y <- rnorm(n, X, 0.8)
fit <- SPQR(X = X, Y = Y, method = "MCMC", normalize = TRUE, verbose = FALSE)
# save.SPQR(fit, name = "mcmc_fit")
# fit <- load.SPQR("mcmc_fit")
```

`plotEstimator` *plot SPQR estimators*

Description

Computes and plots the estimated PDF/CDF/QF curves.

Usage

```
plotEstimator(object, X, ...)
```

Arguments

`object`	An object of class `"SPQR"`
`X`	A row vector indicating covariate values for which the conditional PDF/CDF/QF is computed and plotted.
`...`	Arguments passed on to `predict.SPQR`
	`nY` An integer number indicating length of grid when `Y` is not specified. Default: 101.
	`type` The function to be predicted; `"PDF"`: probability density function, `"CDF"`: cumulative distribution function, and `"QF"`: the quantile function (default).
	`tau` The grid of quantiles for which the quantile function is computed. Default: `seq(0.1,0.9,0.1)`.
	`ci.level` The credible level for computing the pointwise credible intervals. The default is 0 indicating no credible intervals should be computed.
	`getAll` If `TRUE`, extracts all posterior samples of the prediction. Default: `FALSE`.

Value

A `ggplot` object.

Examples

```
set.seed(919)
n <- 200
X <- rbinom(n, 1, 0.5)
Y <- rnorm(n, X, 0.8)
control <- list(iter = 200, warmup = 150, thin = 1)
fit <- SPQR(X = X, Y = Y, method = "MCMC", control = control,
            normalize = TRUE, verbose = FALSE)


## plot estimated PDF
plotEstimator(fit, type = "PDF", X = 0)
```

---

plotGOF *goodness-of-fit test for SPQR estimator*

Description

Performs a goodness-of-fit test for the estimated conditional probability density function (PDF) using probability inverse transformation method.

Usage

```
plotGOF(object, getAll = FALSE)
```

Arguments

object	An object of class SPQR.
getAll	If TRUE and SPQR is fitted with method = "MCMC", plots all posterior samples of Q-Q lines. Default: FALSE.

Value

A ggplot object.

Examples

```
set.seed(919)
n <- 200
X <- rbinom(n, 1, 0.5)
Y <- rnorm(n, X, 0.8)
control <- list(iter = 200, warmup = 150, thin = 1)
fit <- SPQR(X = X, Y = Y, method = "MCMC", control = control,
            normalize = TRUE, verbose = FALSE)

## Goodness-of-fit test
```

```
plotGOF(fit)
```

`plotMCMCtrace` *plot MCMC trace plots*

Description

Show trace plot of the log-likelihood or estimates, of a `"SPQR"` class object fitted using the MCMC method

Usage

```
plotMCMCtrace(
  object,
  target = c("loglik", "PDF", "CDF", "QF"),
  X = NULL,
  Y = NULL,
  tau = 0.5,
  window = NULL
)
```

Arguments

`object`	An object of class `SPQR`.
`target`	A character indicating the statistic/estimate for which traceplot should be plotted; `"loglik"`: log-likelihood (default), `"PDF"`: probability density function, `"CDF"`: cumulative density function, `"QF"`: quantile function.
`X`	If `target != "loglik"`, a row vector specifying the covariate values for which the estimates are computed. Default: `NULL`.
`Y`	If `target = "PDF"` or `target = "CDF"` a scalar specifying the response value for which the estimates are computed. Default: `NULL`.
`tau`	If `target != "QF"`, a scalar specifying the quantile level for which the estimates are computed. Default: 0.5.
`window`	A vector specifying the range of index of the MCMC samples for which the traceplot should be plotted. Default is `NULL` indicating that the whole chain is plotted.

Value

A `ggplot` object.

Examples

```
set.seed(919)
n <- 200
X <- rbinom(n, 1, 0.5)
Y <- rnorm(n, X, 0.8)
control <- list(iter = 200, warmup = 150, thin = 1)
fit <- SPQR(X = X, Y = Y, method = "MCMC", control = control,
            normalize = TRUE, verbose = FALSE)

## trace plot for log-likelihood
plotMCMCtrace(fit, target = "loglik")
```

---

plotQALE *plot accumulated local effects (ALE)*

Description

Computes and plots the quantile ALEs of a SPQR class object. The function plots the ALE main effects across tau for a single covariate using line plots, and the ALE interaction effects between two covariates across tau using contour plots.

Usage

```
plotQALE(object, ...)
```

Arguments

object	An object of class "SPQR".
...	Arguments passed on to QALE
	var.index a numeric scalar or length-two vector of indices of the covariates for which the ALEs will be calculated. When length(var.index) = 1, the function computes the main effect for X[,var.index]. When length(var.index) = 2, the function computes the interaction effect between X[,var.index[1]] and X[,var.index[2]].
	tau The quantiles of interest.
	n.bins the maximum number of intervals into which the covariate range is divided when calculating the ALEs. The actual number of intervals depends on the number of unique values in X[,var.index]. When length(var.index) = 2, n.bins is applied to both covariates.
	ci.level The credible level for computing the pointwise credible intervals for ALE when length(var.index) = 1. The default is 0 indicating no credible intervals should be computed.
	getAll If TRUE and length(var.index) = 1, extracts all posterior samples of ALE.

pred.fun A function that will be used instead of predict.SPQR() for computing predicted quantiles given covariates. This can be useful when the user wants to compare the QALE calculated using SPQR to that using other quantile regression models, or maybe that using the true model in a simulation study.

Value

A ggplot object.

Examples

```
set.seed(919)
n <- 200
X <- runif(n,0,2)
Y <- rnorm(n,X^2,0.3+X/2)
control <- list(iter = 200, warmup = 150, thin = 1)
fit <- SPQR(X=X, Y=Y, n.knots=12, n.hidden=3, method="MCMC",
            control=control, normalize=TRUE)

## compute quantile ALE main effect of X at tau = 0.2,0.5,0.8
plotQALE(fit, var.index=1, tau=c(0.2,0.5,0.8))
```

---

plotQVI *plot variable importance comparison by quantile*

Description

Computes the quantile ALE-induced variable importance (VI) measure for each of the covariate specified in var.index, and produces a ranking plot of the covariates using bar plot for each quantile of interest.

Usage

```
plotQVI(object, var.index = NULL, var.names = NULL, ...)
```

Arguments

object	An object of class SPQR.
var.index	A vector specifying the index of the covariates for which VI measures should be computed. Default is NULL indicating all covariates are considered.
var.names	The names of the covariates to appear in the bar plots. Default is NULL and the the function will use generic names generated by parse(text=paste0("X[",var.index,"]")).
...	Arguments passed on to QALE
	tau The quantiles of interest.

`n.bins` the maximum number of intervals into which the covariate range is divided when calculating the ALEs. The actual number of intervals depends on the number of unique values in `X[,var.index]`. When `length(var.index) = 2`, `n.bins` is applied to both covariates.

`ci.level` The credible level for computing the pointwise credible intervals for ALE when `length(var.index) = 1`. The default is 0 indicating no credible intervals should be computed.

`pred.fun` A function that will be used instead of `predict.SPQR()` for computing predicted quantiles given covariates. This can be useful when the user wants to compare the QALE calculated using SPQR to that using other quantile regression models, or maybe that using the true model in a simulation study.

Value

A `ggplot` object.

Examples

```
set.seed(919)
n <- 200
X <- matrix(runif(n*2, 0, 2), nrow = n, ncol = 2)
Y <- rnorm(n, X[,1]^2, 0.3+X[,1]/2)
control <- list(iter = 200, warmup = 150, thin = 1)
fit <- SPQR(X=X, Y=Y, n.knots=12, n.hidden=5, method="MCMC",
            control=control, normalize=TRUE, verbose = FALSE)

## compute quantile VI of at tau = 0.2,0.5,0.8
plotQVI(fit, tau=c(0.2,0.5,0.8))
```

`predict.SPQR` *predict method for class* `SPQR`

Description

Computes the predicted values for different functions based on the fitted `"SPQR"` object.

Usage

```
## S3 method for class 'SPQR'
predict(
  object,
  X,
  Y = NULL,
  nY = 101,
```

```
  type = c("QF", "PDF", "CDF"),
  tau = seq(0.1, 0.9, 0.1),
  ci.level = 0,
  getAll = FALSE,
  ...
)
```

Arguments

object	An object of class SPQR.
X	The covariate vector/matrix for which the predictions are computed.
Y	The response vector for which the predictions are computed. Default is NULL indicating that a equi-distant grid vector on [0,1] of length nY is used.
nY	An integer number indicating length of grid when Y is not specified. Default: 101.
type	The function to be predicted; "PDF": probability density function, "CDF": cumulative distribution function, and "QF": the quantile function (default).
tau	The grid of quantiles for which the quantile function is computed. Default: seq(0.1,0.9,0.1).
ci.level	The credible level for computing the pointwise credible intervals. The default is 0 indicating no credible intervals should be computed.
getAll	If TRUE, extracts all posterior samples of the prediction. Default: FALSE.
...	Other arguments.

Value

A named array containing all predicted values.

Examples

```
set.seed(919)
n <- 200
X <- rbinom(n, 1, 0.5)
Y <- rnorm(n, X, 0.8)
control <- list(iter = 200, warmup = 150, thin = 1)
fit <- SPQR(X = X, Y = Y, method = "MCMC", control = control,
            normalize = TRUE, verbose = FALSE)

## compute the estimated PDF of Y conditioned on X = 0
pdf <- predict(fit, type = "PDF", X = 0, Y = seq(0, 1, 0.01))
plot(seq(0, 1, 0.01), pdf, xlab = "Y", ylab = "Density")
```

`print.SPQR` *print method for class* `SPQR`

Description

Summarizes and print the output produced by SPQR() in an organized way.

Usage

```
## S3 method for class 'SPQR'
print(x, ...)
```

Arguments

`x`	An object of class `SPQR`
`...`	Arguments passed on to `print.summary.SPQR`
	`showModel` If `TRUE`, prints the detailed NN architecture by layer.

Details

This is equivalent to the function call `print.summary.SPQR(summary.SPQR(object), ...)`.

Value

No return value, called for side effects.

Examples

```
set.seed(919)
n <- 200
X <- rbinom(n, 1, 0.5)
Y <- rnorm(n, X, 0.8)
control <- list(iter = 200, warmup = 150, thin = 1)
fit <- SPQR(X = X, Y = Y, method = "MCMC", control = control,
            normalize = TRUE, verbose = FALSE)
print(fit, showModel = TRUE)
```

print.summary.SPQR *print method for* "summary.SPQR"

Description

Print the output produced by summary.SPQR().

Usage

```
## S3 method for class 'summary.SPQR'
print(x, showModel = FALSE, ...)
```

Arguments

x	An object of class summary.SPQR
showModel	If TRUE, prints the detailed NN architecture by layer.
...	Other arguments.

Value

No return value, called for side effects.

Examples

```
set.seed(919)
n <- 200
X <- rbinom(n, 1, 0.5)
Y <- rnorm(n, X, 0.8)
control <- list(iter = 200, warmup = 150, thin = 1)
fit <- SPQR(X = X, Y = Y, method = "MCMC", control = control,
            normalize = TRUE, verbose = FALSE)

## summarize output
summary(fit)
```

QALE *quantile accumulated local effects (ALE)*

Description

Computes the quantile ALEs of a SPQR class object. The function plots the ALE main effects across tau using line plots for a single covariate, and the ALE interaction effects across tau using contour plots for two covariates .

Usage

```
QALE(
  object,
  var.index,
  tau = seq(0.1, 0.9, 0.1),
  n.bins = 40,
  ci.level = 0,
  getAll = FALSE,
  pred.fun = NULL
)
```

Arguments

object	An object of class SPQR.
var.index	a numeric scalar or length-two vector of indices of the covariates for which the ALEs will be calculated. When length(var.index) = 1, the function computes the main effect for X[,var.index]. When length(var.index) = 2, the function computes the interaction effect between X[,var.index[1]] and X[,var.index[2]].
tau	The quantiles of interest.
n.bins	the maximum number of intervals into which the covariate range is divided when calculating the ALEs. The actual number of intervals depends on the number of unique values in X[,var.index]. When length(var.index) = 2, n.bins is applied to both covariates.
ci.level	The credible level for computing the pointwise credible intervals for ALE when length(var.index) = 1. The default is 0 indicating no credible intervals should be computed.
getAll	If TRUE and length(var.index) = 1, extracts all posterior samples of ALE.
pred.fun	A function that will be used instead of predict.SPQR() for computing predicted quantiles given covariates. This can be useful when the user wants to compare the QALE calculated using SPQR to that using other quantile regression models, or maybe that using the true model in a simulation study.

Value

x	If length(var.index) = 1, a (n.bins+1)-length vector specifying the ordered predictor values at which the ALE plot function is calculated. These are the break points for the n.bins intervals into which the predictor range is divided, plus the lower boundary of the first interval and the upper boundary of the last interval. If length(var.index) = 2, a list of two such vectors, the first containing the X[,var.index[1]] values and the second containing the X[,var.index[2]] values at which the ALE plot function is calculated.
ALE	If length(var.index) = 1, a (n.bins+1) by length(tau) matrix of predicted ALE values for every combination of x and tau. If length(var.index) = 2, a 3-dimensional array of predicted ALE values. Each slice corresponds to a quantile. Within each slice, the rows correspond to X[,var.index[1]] and the columns correspond to X[,var.index[2]].

Examples

```
set.seed(919)
n <- 200
X <- runif(n,0,2)
Y <- rnorm(n,X^2,0.3+X/2)
control <- list(iter = 200, warmup = 150, thin = 1)
fit <- SPQR(X=X, Y=Y, n.knots=12, n.hidden=3, method="MCMC",
            control=control, normalize=TRUE, verbose = FALSE)

## compute quantile ALE main effect of X at tau = 0.2,0.5,0.8
ale <- QALE(fit, var.index=1, tau=c(0.2,0.5,0.8))
```

---

save.SPQR *save fitted SPQR model*

Description

Save SPQR object in a designated directory. If SPQR is fitted with method = "MCMC" then only a .SPQR file is saved; otherwise, a .pt file storing the fitted torch model is also saved.

Usage

```
save.SPQR(object, name = stop("`name` must be specified"), path = NULL)
```

Arguments

object	An object of class SPQR.
name	The name of the saved object excluding extension.
path	The path to save the object. Default is the current working directory.

Value

No return value, called for side effects.

Examples

```
set.seed(919)
n <- 200
X <- rbinom(n, 1, 0.5)
Y <- rnorm(n, X, 0.8)
fit <- SPQR(X = X, Y = Y, method = "MCMC", normalize = TRUE, verbose = FALSE)
# save.SPQR(fit, name = "mcmc_fit")
```

SPQR *Fitting SPQR models*

Description

Main function of the package. Fits SPQR using the maximum likelihood estimation (MLE), maximum *a posterior* (MAP) or Markov chain Monte Carlo (MCMC) method. Returns an object of S3 class SPQR.

Usage

```
SPQR(
  X,
  Y,
  n.knots = 10,
  n.hidden = 10,
  activation = c("tanh", "relu", "sigmoid"),
  method = c("MLE", "MAP", "MCMC"),
  prior = c("ARD", "GP", "GSM"),
  hyperpar = list(),
  control = list(),
  normalize = FALSE,
  verbose = TRUE,
  seed = NULL,
  ...
)
```

Arguments

X	The covariate matrix (without intercept column)
Y	The response vector.
n.knots	The number of basis functions. Default: 10.
n.hidden	A vector specifying the number of hidden neurons in each hidden layer. Default: 10.
activation	The hidden layer activation. Either "tanh" (default) or "relu".
method	Method for estimating SPQR. One of "MLE", "MAP" (default) or "MCMC".
prior	The prior model for variance hyperparameters. One of "GP", "ARD" (default) or "GSM".
hyperpar	A list of named hyper-prior hyperparameters to use instead of the default values, including a_lambda, b_lambda, a_sigma and b_sigma. The default value is 0.001 for all four hyperparameters.
control	A list of named and method-dependent parameters that allows finer control of the behavior of the computational approaches. 1. Parameters for MLE and MAP methods

- use.GPU If TRUE GPU computing will be used if torch::cuda_is_available() returns TRUE. Default: FALSE.
- lr The learning rate used by the Adam optimizer, i.e., torch::optim_adam().
- dropout A length two vector specifying the dropout probabilities in the input and hidden layers respectively. The default is c(0,0) indicating no dropout.
- batchnorm If TRUE batch normalization will be used after each hidden activation.
- epochs The number of passes of the entire training dataset in gradient descent optimization. If early.stopping.epochs is used then this is the maximum number of passes. Default: 200.
- batch.size The size of mini batches for gradient calculation. Default: 128.
- valid.pct The fraction of data used as validation set. Default: 0.2.
- early.stopping.epochs The number of epochs before stopping if the validation loss does not decrease. Default: 10.
- print.every.epochs The number of epochs before next training progress in printed. Default: 10.
- save.path The path to save the fitted torch model. By default a folder named "SPQR_model" is created in the current working directory to store the model.
- save.name The name of the file to save the fitted torch model. Default is "SPQR.model.pt".

2. Parameters for MCMC method

These parameters are similar to those in rstan::stan(). Detailed explanations can be found in the Stan reference manual.

- algorithm The sampling algorithm; "HMC": Hamiltonian Monte Carlo with dual-averaging, "NUTS": No-U-Turn sampler (default).
- iter The number of MCMC iterations (including warmup). Default: 2000.
- warmup The number of warm-up/burn-in iterations for step-size and mass matrix adaptation. Default: 500.
- thin The number of iterations before saving next post-warmup samples. Default: 1.
- stepsize The discretization interval/step-size ϵ of leap-frog integrator. Default is NULL which indicates that it will be adaptively selected during warm-up iterations.
- metric The type of mass matrix; "unit": diagonal matrix of ones, "diag": diagonal matrix with positive diagonal entries estimated during warmup iterations (default), "dense": a dense, symmetric positive definite matrix with entries estimated during warm-up iterations.
- delta The target Metropolis acceptance rate. Default: 0.9.
- max.treedepth The maximum tree depth in NUTS. Default: 6.
- int.time The integration time in HMC. The number of leap-frog steps is calculated as $L_\epsilon = \lfloor t/\epsilon \rfloor$. Default: 0.3.

normalize If TRUE, all covariates will be normalized to take values between [0,1].

verbose	If TRUE (default), training progress will be printed.
seed	Random number generation seed.
...	other parameters to pass to control.

Value

An object of class SPQR. A list containing mostly internal model fitting information to be used by helper functions.

References

Xu SG, Reich BJ (2021). *Bayesian Nonparametric Quantile Process Regression and Estimation of Marginal Quantile Effects.* Biometrics. doi: 10.1111/biom.13576

Examples

```
set.seed(919)
n <- 200
X <- rbinom(n, 1, 0.5)
Y <- rnorm(n, X, 0.8)
control <- list(iter = 200, warmup = 150, thin = 1)
fit <- SPQR(X = X, Y = Y, method = "MCMC", control = control,
            normalize = TRUE, verbose = FALSE)

## summarize output
summary(fit)

## plot estimated PDF
plotEstimator(fit, type = "PDF", X = 0)
```

---

summary.SPQR *summary method for class* SPQR

Description

summarizes the output produced by SPQR() and structures them in a more organized way to be examined by the user.

Usage

```
## S3 method for class 'SPQR'
summary(object, ...)
```

Arguments

object	An object of class SPQR.
...	Other arguments.

Value

An object of class `summary.SPQR`. A list containing summary information of the fitted model.

`method`	The estimation method
`time`	The elapsed time
`prior`	If `method = "MAP"` or `method = "MCMC"`, the hyperprior model for the variance hyperparameters
`model`	If `method = "MLE"` or `method = "MAP"`, the fitted `torch` model. If `method = "MCMC"`, the posterior samples of neural network parameters
`loss`	If `method = "MLE"` or `method = "MAP"`, the train and validation loss
`optim.info`	If `method = "MLE"` or `method = "MAP"`, configuration information of the Adam routine
`elpd`	If `method = "MCMC"`, the expected log-predictive density
`diagnostics`	If `method = "MCMC"`, diagnostic information of the MCMC chain

Examples

```
set.seed(919)
n <- 200
X <- rbinom(n, 1, 0.5)
Y <- rnorm(n, X, 0.8)
control <- list(iter = 200, warmup = 150, thin = 1)
fit <- SPQR(X = X, Y = Y, method = "MCMC", control = control,
            normalize = TRUE, verbose = FALSE)

## summarize output
summary(fit)
```

Index