Package ‘regMMD’

October 25, 2024

Version 0.0.1

Title Robust Regression and Estimation Through Maximum Mean Discrepancy Minimization

Description The functions in this package compute robust estimators by minimizing a kernel-based distance known as MMD (Maximum Mean Discrepancy) between the sample and a statistical model. Recent works proved that these estimators enjoy a universal consistency property, and are extremely robust to outliers. Various optimization algorithms are implemented: stochastic gradient is available for most models, but the package also allows gradient descent in a few models for which an exact formula is available for the gradient. In terms of distribution fit, a large number of continuous and discrete distributions are available: Gaussian, exponential, uniform, gamma, Poisson, geometric, etc. In terms of regression, the models available are: linear, logistic, gamma, beta and Poisson.
Alquier, P. and Gerber, M. (2024) <doi:10.1093/biomet/asad031>
Cherief-Abdellatif, B.-E. and Alquier, P. (2022) <doi:10.3150/21-BEJ1338>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

RdMacros Rdpack

Imports Rdpack (>= 0.7)

Author Pierre Alquier [aut, cre] (<`https://orcid.org/0000-0003-4249-7337`>),
Mathieu Gerber [aut] (<`https://orcid.org/0000-0001-6774-2330`>)

Maintainer Pierre Alquier <`pierre.alquier.stat@gmail.com`>

NeedsCompilation no

Repository CRAN

Date/Publication 2024-10-25 08:10:02 UTC

Contents

mmd_est *MMD estimation*

Description

Fits a statistical models to the data, using the robust procedure based on maximum mean discrepancy (MMD) minimization introduced and studied in Briol et al. (2019); Chérief-Abdellatif and Alquier (2022).

Usage

```
mmd_est(x, model, par1, par2, kernel, bdwth, control= list())
```

Arguments

`x` Data. Must be a vector for univariate models, a matrix of dimension n by d, where n is the sample size and d the dimension of the model.

`model` Parametric model to be fitted to the data. No default. See details for the list of available models.

`par1` First parameter of the model. In models where the first parameter is fixed, it is necessary to provide a value for `par1`. In models where the first parameter is estimated, `par1` can be used to provide an alternative to the default initialization of the optimization algorithms.

`par2` Second parameter of the model (if any). In models where the second parameter is fixed, it is necessary to provide a value for `par2`. In models where the first parameter is estimated,`par2` can be used to provide an alternative to the default initialization of the optimization algorithms.

`kernel` Kernel to be used in the MMD. Available options for `kernel` are "Gaussian" (Gaussian kernel), "Laplace" (Laplace, or exponential, kernel) and "Cauchy" (Cauchy kernel). By default, `kernel="Gaussian"`

`bdwth` Bandwidth parameter for the kernel. `bdwth` must be a strictly positive real number. By default, the value of `bdwth` is chosen using the median heuristic (Garreau et al. 2017).

`control` A `list` of control parameters for the numerical optimization of the objective function. See details.

Details

Available options for `model` are:

`"beta"` Beta distribution with pdf $x^{a-1}(1-x)^{(}b-1)$ on $[0,1]$, `par1`$=a$ and `par2`$=b$ are both estimated.

`"binomial"` Binomial distribution with pmf $p^x(1-p)^{N-x}$ on $\{0,1,...,N\}$, `par1`$=N$ and `par2`$=p$ are both estimated. Note that in this case, if the user specifies a value for N, it is used as an upper bound rather than an initialization.

`"binomial.prob"` Binomial distribution with pmf $p^x(1-p)^{N-x}$ on $\{0,1,...,N\}$, `par1`$=N$ is fixed and must be specified by the user while `par2`$=p$ is estimated.

`"binomial.size"` Binomial distribution with pmf $p^x(1-p)^{N-x}$ on $\{0,1,...,N\}$, `par1`$=N$ is estimated while `par2`$=p$ fixed and must be specified by the user. Note that in this case, if the user specifies a value for N, it is used as an upper bound rather than an initialization.

`"Cauchy"` Cauchy distribution with pdf $1/(1+(x-m)^2)$, `par1`$=m$ is estimated.

`"continuous.uniform.loc"` Uniform distribution with pdf 1 on $[m-L/2, m+L/2]$, `par1`$=m$ is estimated while `par2`$=L$ is fixed and must be specified by the user.

`"continuous.uniform.upper"` Uniform distribution with pdf 1 on $[a,b]$, `par1`$=a$ is fixed and must be specified by the user while `par2`$=b$ is estimated.

`"continuous.uniform.lower.upper"` Uniform distribution with pdf 1 on $[a,b]$, `par1`$=a$ and `par2`$=b$ are estimated.

`"Dirac"` Dirac mass at point a on the reals, `par1`$=a$ is estimated.

`"discrete.uniform"` Uniform distribution with pmf 1 on $\{1,2,..,M\}$, `par1`$=M$ is estimated. Note that in this case, if the user specifies a value for M, it is used as an upper bound rather than an initialization.

`"exponential"` Exponential distribution with pdf $\exp(-bx)$ on positive reals R_+, `par1`$=b$ is estimated.

`"gamma"` Gamma distribution with pdf $x^{a-1}\exp(-bx)$ on positive reals R_+, `par1`$=a>=0.5$ and `par2`$=b$ are estimated.

`"gamma.shape"` Gamma distribution with pdf $x^{a-1}\exp(-bx)$ on positive reals R_+, `par1`$=a>=0.5$ is estimated while `par2`$=b$ is fixed and must be specified by the user.

`"gamma.rate"` Gamma distribution with pdf $x^{a-1}\exp(-bx)$ on positive reals R_+, `par1`$=a>=0.5$ is fixed and must be specified by the user while `par2`$=b$ is estimated.

`"Gaussian"` Gaussian distribution with pdf $\exp(-(x-m)^2/2s^2)$ on reals R, `par1`$=m$ and `par2`$=s$ are estimated.

`"Gaussian.loc"` Gaussian distribution with pdf $\exp(-(x-m)^2/2s^2)$ on reals R, `par1`$=m$ is estimated while `par2`$=s$ is fixed and must be specified by the user.

`"Gaussian.scale"` Gaussian distribution with pdf $\exp(-(x-m)^2/2s^2)$ on reals R, `par1`$=m$ is fixed and must be specified by the user while `par2`$=s$ is estimated.

`"geometric"` Geometric distribution with pmf $p(1-p)^x$ on $\{0,1,2,...\}$, `par1`$=p$ is estimated.

`"multidim.Dirac"` Dirac mass at point a on R^d, `par1`$=a$ (d-dimensional vector) is estimated.

`"multidim.Gaussian"` Gaussian distribution with pdf $\exp(-(x-m)'U'U(x-m)$ on R^d, `par1`$=m$ (d-dimensional vector) and `par2`$=U$ (d-d matrix) are estimated.

`"multidim.Gaussian.loc"` Gaussian distribution with pdf $\exp(-\|x-m\|^2/2s^2)$ on R^d, `par1`$=m$ (d-dimensional vector) is estimated while `par2`$=s$ is fixed.

`"multidim.Gaussian.scale"` Gaussian distribution with pdf $\exp(-(x-m)'U'U(x-m)$ on R^d, `par1`$=m$ (d-dimensional vector) is fixed and must be specified by the user while and `par2`$=U$ (d-d matrix) is estimated.

`"Pareto"` Pareto distribution with pmf $1/x^{a+1}$ on the reals >1, `par1`$=a$ is estimated.

`"Poisson"` Poisson distribution with pmf $b^x/x!$ on $\{0,1,2,...\}$, `par1`$=b$ is estimated.

The `control` argument is a list that can supply any of the following components:

burnin Length of the burn-in period in GD or SGD. burnin must be a non-negative integer and defaut burnin==500.

nsteps Number of iterations performed after the burn-in period in GD or SGD. nsetps must be an integer strictly larger than 2 and by default nsteps=1000

stepsize Stepsize parameter. An adaptive gradient step is used (adagrad), but it is possible to pre-multiply it by stepsize. It must be strictly positive number and by default stepsize=1

epsilon Parameter used in adagrad to avoid numerical errors in the computation of the step-size. epsilon must be a strictly positive real number and by default epsilon=10^{-4}.

method Optimization method to be used: "EXACT" for exact, "GD" for gradient descent and "SGD" for stochastic gradient descent. Not all methods are available for all models. By default, exact is preferred to GD which is prefered to SGD.

Value

MMD_est returns an object of class "estMMD".

The functions summary can be used to print the results.

An object of class estMMD is a list containing the following components:

model	Model estimated
par1	In models where the first parameter is fixed, this is the value par1 fixed by the user. In models where the first parameter is estimated, this is the initialization of the optimization procedure
par2	In models where the second parameter is fixed, this is the value par2 fixed by the user. In models where the second parameter is estimated, this is the initialization of the optimization procedure
kernel	Kernel used in the MMD
bdwth	Bandwidth used. That is, either the value specified by the user, either the bandwidth computedby the median heuristic
burnin	Number of steps in the burnin of the GD or SGD algorithm
nstep	Number of steps in the GD or SGD algorithm
stepsize	Stepize parameter in GD or SGD
epsilon	Parameter used in adagrad to avoid numerical errors in the computation of the step-size
method	Optimization method used
error	Error message (if any)
estimator	Estimated parameter(s)
type	Takes the value "est"

References

Briol F, Barp A, Duncan AB, Girolami M (2019). “Statistical inference for generative models with maximum mean discrepancy.” *arXiv preprint arXiv:1906.05944*.

Chérief-Abdellatif B, Alquier P (2022). “Finite Sample Properties of Parametric MMD Estimation: Robustness to Misspecification and Dependence.” *Bernoulli*, **28**(1), 181-213.

Garreau D, Jitkrittum W, Kanagawa M (2017). “Large sample analysis of the median heuristic.” *arXiv preprint arXiv:1707.07269*.

Examples

```
#simulate data
x = rnorm(50,0,1.5)

# estimate the mean and variance (assuming the data is Gaussian)
Est = mmd_est(x, model="Gaussian")

# print a summary
summary(Est)

# estimate the mean (assuming the data is Gaussian with known standard deviation =1.5)
Est2 = mmd_est(x, model="Gaussian.loc", par2=1.5)

# print a summary
summary(Est2)

# estimate the standard deviation (assuming the data is Gaussian with known mean = 0)
Est3 = mmd_est(x, model="Gaussian.scale", par1=0)

# print a summary
summary(Est3)

# test of the robustness
x[42] = 100

mean(x)

# estimate the mean and variance (assuming the data is Gaussian)
Est4 = mmd_est(x, model="Gaussian")
summary(Est4)
```

mmd_reg *MMD regression*

Description

Fits a regression model to the data, using the robust procedure based on maximum mean discrepancy (MMD) minimization introduced and studied in Alquier and Gerber (2024).

Usage

```
mmd_reg(y, X, model, intercept, par1, par2, kernel.y, kernel.x, bdwth.y, bdwth.x,
         control= list())
```

Arguments

`y` Response variable. Must be a vector of length $n \geq 3$.

`X` Design matrix. `X` must be either a matrix of dimension $n \times p$ or a vector of size n, where n is the size of `y`.

`model` Regression model to be fitted to the data. By default, the linear regression model with $\mathcal{N}_1(0, \phi^2)$ error terms is used. See details for the list of available models.

`intercept` If `intercept=TRUE` an intercept is added to the model, while no intercept is added if `intercept=FALSE`. By default, `intercept=TRUE`.

`par1` Values of the regression coefficients of the model used as starting values to numerically optimize the objective function. `par1` must be either a vector of size p, with p the number of columns of `X` (or with $p = 1$ if `X` is a vector), or equal to `"auto"`, in which case a non-robust estimate of the regression coefficients of the model is used as starting values. By default, `par1="auto"`.

`par2` A value for the auxilliary parameter ϕ of the model. `par2` needs to be specified only if relevant (see details for the list of models having an auxilliary parameter ϕ). If the model assumes that ϕ is known (see details) then `par2` must be a strictly positive real number and the model is estimated with ϕ =`par2`. If the model assumes that ϕ is unknown (see details) then the value specified by `par2` is used as starting value to numerically optimize the objective function. For such models `par2` must be either a strictly positive real number or equal to `"auto"`, in which case a non-robust estimate of ϕ is used as starting value. By default, `par2="auto"`.

`kernel.y` Kernel applied on the response variable. Available options for `kernel.y` are `"Gaussian"` (Gaussian kernel), `"Laplace"` (Laplace, or exponential, kernel) and `"Cauchy"` (Cauchy kernel). By default, `kernel.y="Gaussian"` for the linear regression model and `kernel.y="Laplace"` for the other models.

`kernel.x` Kernel applied on the explanatory variables. Available options for `kernel.x` are `"Gaussian"` (Gaussian kernel), `"Laplace"` (Laplace, or exponential, kernel) and `"Cauchy"` (Cauchy kernel). By default, `kernel.x="Laplace"`.

`bdwth.y` Bandwidth parameter for the kernel `kernel.y`. `bdwth.y` must be eiter a strictly positive real number or equal to `"auto"`, in which case the median heuristic is used to select the bandwidth parameter of `kernel.y` (see details). By default, `bdwth.y="auto"`.

`bdwth.x` Bandwidth parameter for the kernel `kernel.x`. `bdwth.x` must be either a non-negative real number or equal to `"auto"`, in which case a rescaled version of the median heuristic is used to specify the bandwidth parameter of `kernel.x` (see details). By default, `bdwth.x`=0. Remark: for computational reasons, for large dataset (i.e.~when the sample size is bigger than a few thousands) it is recommended to choose `bdwth.x`=0 (see details).

`control` A `list` of control parameters for the numerical optimization of the objective function. See details.

Details

Available options for `model` are:

"linearGaussian" Linear regression model with $\mathcal{N}_1(0,\phi^2)$ error terms, with ϕ unknown.

"linearGaussian.loc" Linear regression model with $\mathcal{N}_1(0,\phi^2)$ error terms, with ϕ known.

"gamma" Gamma regression model with unknown shape parameter ϕ. The inverse function is used as link function.

"gamma.loc" Gamma regression model with known shape parameter ϕ. The inverse function is used as link function.

"beta" Beta regression model with unknown precision parameter ϕ. The logistic function is used as link function.

"beta.loc" Beta regression model with known precision parameter ϕ. The logistic function is used as link function.

"logistic" Logistic regression model.

"exponential" Exponential regression.

"poisson" Poisson regression model.

When `bdwth.x`>0 the function `reg_mmd` computes the estimator $\hat{\theta}_n$ introduced in Alquier and Gerber (2024). When `bdwth.x`=0 the function `reg_mmd` computes the estimator $\tilde{\theta}_n$ introduced in Alquier and Gerber (2024). The former estimator has stronger theoretical properties but is more expensive to compute (see below).

When `bdwth.x`=0 and `model` is "linearGaussian", "linearGaussian.loc" or "logistic", the objective function and its gradient can be computed on $\mathcal{O}(n)$ operations, where n is the sample size (i.e. the dimension of `y`). In this case, gradient descent with backtraking line search is used to perform the minimizatiom. The algorithm stops when the maximum number of iterations `maxit` is reached, or as soon as the change in the objective function is less than `eps_gd` times the current function value. In the former case, a warning message is generated. By defaut, `maxit`=5×10^4 and `eps_gd=sqrt(.Machine$double.eps)`, and the value of these two parameters can be changed using the `control` argument of `reg_mmd`.

When `bdwth.x`>0 and `model` is "linearGaussian", "linearGaussian.loc" or "logistic", the objective function and its gradient can be computed on $\mathcal{O}(n^2)$ operations. To reduce the computational cost the objective function is minimized using norm adagrad (Duchi et al. 2011), an adaptive step size stochastic gradient algorithm. Each iteration of the algorithm requires $\mathcal{O}(n)$ operations. However, the algorithm has an intialization step that requires $\mathcal{O}(n^2)$ operations and has a memory requirement of size $\mathcal{O}(n^2)$.

When `model` is not in `c("linearGaussian", "linearGaussian.loc", "logistic")`, the objective function and its gradient cannot be computed explicitly and the minimization is performed using norm adagrad. The cost per iteration of the algorithm is $\mathcal{O}(n)$ but, for `bdwth.x`>0, the memory requirement and the initialization cost are both of size $\mathcal{O}(n^2)$.

When adagrad is used, `burnin` iterations are performed as a warm-up step. The algorithm then stops when `burnin+maxit` iterations are performed, or as soon as the norm of the average value of the gradient evaluations computed in all the previous iterations is less than `eps_sg`. A warning message is generated if the maximum number of iterations is reached. By default, `burnin`=10^3, `nsteps`=5×10^4 and `eps_sg`=10^{-5} and the value of these three parameters can be changed using the `control` argument of `reg_mmd`.

If `bdwth.y="auto"` then the value of the bandwidth parameter of `kernel.y` is equal to $H_n/\sqrt{2}$ with H_n the median value of the set $\{|y_i-y_j|\}_{i,j=1}^n$, where y_i denote the ith component of `y`. This definition of bdwth.y is motivated by the results in Garreau et al. (2017). If $H_n = 0$ the bandwidth parameter of kernel.y is set to 1.

If bdwth.x="auto" then the value of the bandwidth parameter of kernel.x is equal to $0.01H_n/\sqrt{2}$ with H_n is the median value of the set $\{\|x_i - x_j\|\}_{i,j=1}^n$, where x_i denote the ith row of the design matrix X. If $H_n = 0$ the bandwidth parameter of kernel.x is set to 1.

The control argument is a list that can supply any of the following components:

rescale: If rescale=TRUE the (non-constant) columns of X are rescalled before perfoming the optimization, while if rescale=FLASE no rescaling is applied. By default rescale=TRUE.

burnin A non-negative integer.

eps_gd A non-negative real number.

eps_sg A non-negative real number.

maxit A integer strictly larger than 2.

stepsize Scaling constant for the step-sizes used by adagrad. stepsize must be a stictly positive number and by default stepsize=1.

trace: If trace=TRUE then the parameter value obtained at the end of each iteration (after the burn-in perdiod for adagrad) is returned. By default, trace=TRUE and trace is automatically set to TRUE if the maximum number of iterations is reached.

epsilon Parameter used in adagrad to avoid numerical errors in the computation of the step-sizes. epsilon must be a strictly positive real number and by default epsilon=10^{-4}.

alpha Parameter for the backtraking line search. alpha must be a real number in $(0, 1)$ and by default alpha=0.8.

c_det Parameter used to control the computational cost of the algorithm when gamma.x> 0, see the Suplementary material in Alquier and Gerber (2024) for mode details. c_det must be a real number in $(0, 1)$ and by default c_det=0.2.

c_rand Parameter used to control the computational cost of the algorithm when bdwth.x> 0, see the Suplementary material in Alquier and Gerber (2024) for mode details. c_rand must be a real number in $(0, 1)$ and by default c_rand=0.1.

Value

MMD_reg returns an object of class "regMMD".

The function summary can be used to print the results.

An object of class regMMD is a list containing the following components:

coefficients Estimated regression coefficients.

intercept If intercept=TRUE an intercept has been added to model, if intercept=FALSE no intercept has been added.

phi If relevant (see details), either the estimated value of the ϕ parameter of model, or the value of ϕ used to fit the model if ϕ is assumed to be known.

kernel.y Kernel applied on the response variable used to fit the model.

bdwth.y Value of the bandwidth for the kernel applied on the response variable used to fit the model.

`kernel.x` Kernel applied on the explanatory variables used to fit the model.

`bdwth.x` Value of the bandwidth for the kernel applied on the explanatory variables used to fit the model.

`par1` Value of the parameter `par1` used to fit the model.

`par2` Value of parameter `par2` used to fit the model.

`trace` If the control parameter `trace=TRUE`, `trace` is a matrix containing the parameter values obtained at the end of each iteration of the optimization algorithm.

References

Alquier P, Gerber M (2024). “Universal robust regression via maximum mean discrepancy.” *Biometrika*, **111**(1), 71-92.

Duchi J, Hazan E, Singer Y (2011). “Adaptive subgradient methods for online learning and stochastic optimization.” *Journal of machine learning research*, **12**(7), 2121-2159.

Garreau D, Jitkrittum W, Kanagawa M (2017). “Large sample analysis of the median heuristic.” *arXiv preprint arXiv:1707.07269*.

Examples

```
#Simulate data
n<-1000
p<-4
beta<-1:p
phi<-1
X<-matrix(data=rnorm(n*p,0,1),nrow=n,ncol=p)
data<-1+X%*%beta+rnorm(n,0,phi)

##Example 1: Linear Gaussian model
y<-data
Est<-mmd_reg(y, X)
summary(Est)

##Example 2: Logisitic regression model
y<-data
y[data>5]<-1
y[data<=5]<-0
Est<-mmd_reg(y, X, model="logistic")
summary(Est)
Est<-mmd_reg(y, X, model="logistic", bdwth.x="auto")
summary(Est)
```

`summary.estMMD` *Summary method for the* `class "estMMD"`

Description

Summary method for the `class "estMMD"`

Usage

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

Arguments

`object`	An object of `class` `"estMMD"`.
`...`	Additional arguments (not used).

Value

No return value, called only to print information on the output of `"estMMD"`.

`summary.regMMD` *Summary method for the* `class` `"regMMD"`

Description

Summary method for the `class` `"regMMD"`

Usage

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

Arguments

`object`	An object of `class` `"regMMD"`.
`...`	Additional arguments (not used).

Value

No return value, called only to print information on the output of `"regMMD"`.

Index