Package 'NORMA'

October 12, 2022

Type Package

Title Builds General Noise SVRs

Version 0.1

Date 2017-01-22

Author Jesus Prada [aut,cre]

Maintainer Jesus Prada <jesus.prada@estudiante.uam.es>

Description Builds general noise SVR models using Naive Online R Minimization Algorithm, NORMA, an optimization method based on classical stochastic gradient descent suitable for computing SVR models in an online setting.

Depends rootSolve

License GPL-2

LazyData TRUE

BuildVignettes TRUE

RoxygenNote 5.0.1

URL http://link.springer.com/chapter/10.1007/978-3-319-19222-2_47

NeedsCompilation no

Repository CRAN

Date/Publication 2017-01-24 01:03:12

R topics documented:

f *Predictor Function*

Description

Computes the predictor function of a general noise SVR based on NORMA optimization.

Usage

```
f(point, t, x, alpha, beta, f_0, kernel = function(x, y, gamma) {
  exp(-gamma * (norm(x - y, type = "2")^2)) }, gamma, no_beta)
```

Arguments

point	numeric with the value of the point where we want to evaluate the predictor function.
t	time parameter value indicating the iteration we want to consider.
x	matrix containing training points. Each row must be a point.
alpha	matrix representing α parameters of NORMA optimization in each iteration, one per row.
beta	numeric representing β parameter of NORMA optimization in each iteration.
f_0	initial hypothesis.
kernel	kernel function to use.
gamma	gaussian kernel parameter γ.
no_beta	boolean indicating if an offset b is used (FALSE) or not (TRUE).

Value

Returns a numeric representing the prediction value.

Author(s)

Jesus Prada, <jesus.prada@estudiante.uam.es>

References

Link to the scientific paper

Kivinen J., Smola A. J., Williamson R.C.: Online learning with kernels. In: IEEE transactions on signal processing, vol. 52, pp. 2165-2176, IEEE (2004).

with theoretical background for NORMA optimization is provided below.

http://realm.sics.se/papers/KivSmoWil04(1).pdf

Examples

```
f(c(1,2,3),2,matrix(c(1,2,3,4,5,6),nrow=2,ncol=3,byrow=TRUE),
matrix(c(1,2,3,4,5,6),nrow=2,ncol=3,byrow=TRUE),
c(1,2),0,function(x,y,gamma=0){x%*%y},0.1,FALSE)
```

ILF_cost_der *Cost Functions Derivatives*

Description

`ILF_cost_der` computes the ILF derivative value at a given point.

`zero_laplace_cost_der` computes the value at a given point of the loss function derivative corresponding to a zero-mean Laplace distribution.

`general_laplace_cost_der` computes the value at a given point of the loss function derivative corresponding to a general Laplace distribution.

`zero_gaussian_cost_der` computes the value at a given point of the loss function derivative corresponding to a zero-mean Gaussian distribution.

`general_gaussian_cost_der` computes the value at a given point of the loss function derivative corresponding to a general Gaussian distribution.

`beta_cost_der` computes the value at a given point of the loss function derivative corresponding to a Beta distribution.

`weibull_cost_der` computes the value at a given point of the loss function derivative corresponding to a Weibull distribution.

`moge_cost_der` computes the value at a given point of the loss function derivative corresponding to a MOGE distribution.

Usage

```
ILF_cost_der(phi, epsilon = 0.1, nu = 0)

zero_laplace_cost_der(phi, sigma)

general_laplace_cost_der(phi, sigma, mu)

zero_gaussian_cost_der(phi, sigma_cuad)

general_gaussian_cost_der(phi, sigma_cuad, mu)

beta_cost_der(phi, alpha, beta)

weibull_cost_der(phi, lambda, kappa)

moge_cost_der(phi, lambda, alpha, theta)
```

Arguments

phi	point to use as argument of the loss function derivative.
epsilon	width of the insensitive band.
nu	parameter to control value of epsilon.
sigma	scale parameter of the Laplace distribution.
mu	location or mean parameter of the Laplace or Gaussian distribution, respectively.
sigma_cuad	variance parameter of the Gaussian distribution.
alpha	shape1 parameter of the Beta distribution or second parameter of the MOGE distribution.
beta	shape2 parameter of the Beta distribution.
lambda	lambda scale parameter of the Weibull distribution or first parameter of the MOGE distribution.
kappa	shape parameter of the Weibull distribution.
theta	third parameter of the MOGE distribution.

Details

See also 'References'.

Value

Returns a numeric representing the derivative value at a given point.

Author(s)

Jesus Prada, <jesus.prada@estudiante.uam.es>

References

Link to the scientific paper

Prada, Jesus, and Jose Ramon Dorronsoro. "SVRs and Uncertainty Estimates in Wind Energy Prediction." Advances in Computational Intelligence. Springer International Publishing, 2015. 564-577,

with theoretical background for this package is provided below.

http://link.springer.com/chapter/10.1007/978-3-319-19222-2_47

Examples

```
# ILF derivative value at point phi=1 with default epsilon.
ILF_cost_der(1)

# ILF derivative value at point phi=1 with epsilon=2.
ILF_cost_der(1,2)

# Zero-mean Laplace loss function derivative value at point phi=1 with sigma=1.
zero_laplace_cost_der(1,1)
```

```
# General Laplace loss function derivative value at point phi=1 with mu=0 and sigma=1.
general_laplace_cost_der(1,1,0)

# Zero-mean Gaussian loss function derivative value at point phi=1 with sigma_cuad=1.
zero_gaussian_cost_der(1,1)

# General Gaussian loss function derivative value at point phi=1 with mu=0 and sigma_cuad=1.
general_gaussian_cost_der(1,1,0)

# Beta loss function derivative value at point phi=1 with alpha=2 and beta=3.
beta_cost_der(1,2,3)

# Weibull loss function derivative value at point phi=1 with lambda=2 and kappa=3.
weibull_cost_der(1,2,3)

# MOGE loss function derivative value at point phi=1 with lambda=2 ,alpha=3 and theta=4.
moge_cost_der(1,2,3,4)
```

`linear_kernel` *Kernels*

Description

`linear_kernel` computes the linear kernel between two given vector, x and y.

`gaussian_kernel` computes the gaussian kernel between two given vectors, x and y.

Usage

```
linear_kernel(x, y, gamma = 0)

gaussian_kernel(x, y, gamma)
```

Arguments

`x`	`numeric` vector indicating value of x.
`y`	`numeric` vector indicating value of y.
`gamma`	gaussian kernel parameter γ.

Details

Linear kernel:

$$k(x, y) = x * y$$

Gaussian kernel:

$$k(x, y) = exp(-\gamma||x - y||^2)$$

Value

Returns a numeric representing the kernel value.

Author(s)

Jesus Prada, <jesus.prada@estudiante.uam.es>

Examples

```
# Linear kernel value between point x=c(1,2,3) and point y=c(2,3,4).
linear_kernel(c(1,2,3),c(2,3,4))

# Gaussian kernel value between point x=c(1,2,3) and point y=c(2,3,4) with gamma=0.1.
gaussian_kernel(c(1,2,3),c(2,3,4),0.1)
```

---

mle_parameters *MLE Parameters*

Description

mle_parameters computes the optimal parameters via MLE of a given distribution.

zero_laplace_mle computes the optimal parameters via MLE assuming a zero-mean Laplace as noise distribution.

general_laplace_mle computes the optimal parameters via MLE assuming a general Laplace as noise distribution.

zero_gaussian_mle computes the optimal parameters via MLE assuming a zero-mean Gaussian as noise distribution.

general_gaussian_mle computes the optimal parameters via MLE assuming a general Gaussian as noise distribution.

beta_mle computes the optimal parameters via MLE assuming a Beta as noise distribution.

weibull_mle computes the optimal parameters via MLE assuming a Weibull as noise distribution.

moge_mle computes the optimal parameters via MLE assuming a MOGE as noise distribution.

Usage

```
mle_parameters(phi, dist = "nm", ...)

zero_laplace_mle(phi)

general_laplace_mle(phi)

zero_gaussian_mle(phi)

general_gaussian_mle(phi)
```

```
beta_mle(phi, m1 = mean(phi, na.rm = T), m2 = mean(phi^2, na.rm = T),
  alpha_0 = (m1 * (m1 - m2))/(m2 - m1^2), beta_0 = (alpha_0 * (1 - m1)/m1))

weibull_mle(phi, k_0 = 1)

moge_mle(phi, lambda_0 = 1, alpha_0 = 1, theta_0 = 1)
```

Arguments

`phi`	a vector with residual values used to estimate the parameters.
`dist`	assumed distribution for the noise in the data. Possible values to take: • l: Zero-mean Laplace distribution. • lm: General Laplace distribution. • n: Zero-mean Gaussian distribution. • nm: General Gaussian distribution. • b: Beta distribution. • w: Weibull distribution. • moge: MOGE distribution.
`...`	additional arguments to be passed to the low level functions (see below).
`m1`	first moment of the residuals. Used to compute `alpha_0`.
`m2`	second moment of the residuals. Used to compute `beta_0`.
`alpha_0`	initial value for Newton-Raphson method for the parameter α.
`beta_0`	initial value for Newton-Raphson method for the parameter β.
`k_0`	initial value for Newton-Raphson method for the parameter κ.
`lambda_0`	initial value for Newton-Raphson method for the parameter λ.
`theta_0`	initial value for Newton-Raphson method for the parameter θ. See also 'Details' and multiroot.

Details

For the zero-μ Laplace distribution the optimal MLE parameters are

$$\sigma = mean(|\phi_i|)$$

, where ϕ_i are the residuals passed as argument.

For the general Laplace distribution the optimal MLE parameters are

$$\mu = median(\phi_i)$$

$$\sigma = mean(|\phi_i - \mu|)$$

, where ϕ_i are the residuals passed as argument.

For the zero-μ Gaussian distribution the optimal MLE parameters are

$$\sigma^2 = mean(\phi_i^2)$$

, where ϕ_i are the residuals passed as argument.

For the general Gaussian distribution the optimal MLE parameters are

$$\mu = mean(\phi_i)$$

$$\sigma^2 = mean((\phi_i - \mu)^2)$$

, where ϕ_i are the residuals passed as argument.

For the Beta distribution values of parameters α and β are estimated using Newton-Raphson method.

For the Weibull distribution value of parameter κ is estimated using Newton-Raphson method and then estimated value of λ is computed using the following closed form that depends on κ:

$$\lambda = mean(\phi_i^k appa)^(1/\kappa)$$

For the MOGE distribution values of parameters λ, α and θ are estimated using Newton-Raphson method.

See also 'References'.

Value

`mle_parameters` returns a list with the estimated parameters. Depending on the distribution these parameters will be one or more of the following ones:

sigma scale parameter of the Laplace distribution.

mu location or mean parameter of the Laplace or Gaussian distribution, respectively.

sigma_cuad variance parameter of the Gaussian distribution.

alpha shape1 parameter of the Beta distribution or second parameter of the MOGE distribution.

beta shape2 parameter of the Beta distribution.

k shape parameter of the Weibull distribution.

lambda lambda scale parameter of the Weibull distribution or first parameter of the MOGE distribution.

theta third parameter of the MOGE distribution.

Author(s)

Jesus Prada, <jesus.prada@estudiante.uam.es>

References

Link to the scientific paper

Prada, Jesus, and Jose Ramon Dorronsoro. "SVRs and Uncertainty Estimates in Wind Energy Prediction." Advances in Computational Intelligence. Springer International Publishing, 2015. 564-577,

with theoretical background for this package is provided below.

http://link.springer.com/chapter/10.1007/978-3-319-19222-2_47

Examples

```
# Estimate optimal parameters using default distribution ("nm").
mle_parameters(rnorm(100))

# Estimate optimal parameters using "lm" distribution.
mle_parameters(rnorm(100),dist="lm")

# Equivalent to mle_parameters(rnorm(100),dist="l")
zero_laplace_mle(rnorm(100))

# Equivalent to mle_parameters(rnorm(100),dist="lm")
general_laplace_mle(rnorm(100))

# Equivalent to mle_parameters(rnorm(100),dist="n")
zero_gaussian_mle(rnorm(100))

# Equivalent to mle_parameters(rnorm(100),dist="nm")
general_gaussian_mle(rnorm(100))

# Equivalent to mle_parameters(rnorm(100),dist="b")
beta_mle(rnorm(100))

# Equivalent to mle_parameters(rnorm(100),dist="w")
weibull_mle(rnorm(100))

# Equivalent to mle_parameters(rnorm(100),dist="moge")
moge_mle(rnorm(100))
```

NORMA *NORMA Optimization*

Description

Computes general noise SVR based on NORMA optimization.

Usage

```
NORMA(x, y, f_0 = 0, beta_0 = 0, lambda = 0, rate = function(t) {     1
  }, kernel = linear_kernel, cost_der = ILF_cost_der,
  cost_name = "ILF_cost_der", gamma = 1, max_iterations = nrow(x),
  stopping_threshold = 0, trace = TRUE, no_beta = TRUE,
  fixed_epsilon = TRUE, ...)
```

Arguments

x	`matrix` containing training points. Each row must be a point.
y	`numeric` containing target for training points x.
f_0	initial hypothesis.

`beta_0`	initial value for offset b.
`lambda`	NORMA optimization parameter $lambda$
`rate`	learning rate for NORMA optimization. Must be a function with one argument.
`kernel`	kernel function to use. Must be a function with three arguments such as `gaussian_kernel`. See also linear_kernel
`cost_der`	Loss function derivative to use. See also ILF_cost_der. Must be "ILF_cost_der" when ILF derivative is used.
`cost_name`	`character` indicating the symbolic name of `cost_der`.
`gamma`	gaussian kernel parameter γ.
`max_iterations`	maximum number of NORMA iterations computed.
`stopping_threshold`	value indicating when to stop NORMA optimization. See also 'Details'.
`trace`	`boolean` indicating if information messages should be printed (TRUE) or not (FALSE).
`no_beta`	`boolean` indicating if an offset b is used (FALSE) or not (TRUE).
`fixed_epsilon`	`boolean` indicating if `epsilon` should be updated (FALSE) or not (TRUE).
`...`	additional arguments to be passed to the low level functions.

Details

Optimization will stop when the sum of the differences between all training predicted values of present iteration versus values from previous iteration does not exceeds `stopping_threshold`.

Value

Returns a `list` containing:

alpha `matrix` representing α parameters of NORMA optimization in each iteration, one per row.

beta `numeric` representing β parameter of NORMA optimization in each iteration.

n_iterations Number of NORMA iterations performed.

Author(s)

Jesus Prada, <jesus.prada@estudiante.uam.es>

References

Link to the scientific paper

Kivinen J., Smola A. J., Williamson R.C.: Online learning with kernels. In: IEEE transactions on signal processing, vol. 52, pp. 2165-2176, IEEE (2004).

with theoretical background for NORMA optimization is provided below.

http://realm.sics.se/papers/KivSmoWil04(1).pdf

Examples

```
NORMA(x=matrix(rnorm(10),nrow=10,ncol=1,byrow=TRUE),y=rnorm(10),kernel=function(x,y,gamma=0){x%*%y},
cost_der=function(phi,sigma_cuad,mu){return((phi-mu)/sigma_cuad)},cost_name="example",
sigma_cuad=1,mu=0)
```

Index