Package 'SAM'

January 20, 2025

Type Package

Title Sparse Additive Modelling

Version 1.1.3

Author Haoming Jiang, Yukun Ma, Han Liu, Kathryn Roeder, Xingguo Li, and Tuo Zhao

Maintainer Haoming Jiang <jianghm.ustc@gmail.com>

Depends R (>= 2.14), splines

Description Computationally efficient tools for high dimensional predictive modeling (regression and classification). SAM is short for sparse additive modeling, and adopts the computationally efficient basis spline technique. We solve the optimization problems by various computational algorithms including the block coordinate descent algorithm, fast iterative soft-thresholding algorithm, and newton method. The computation is further accelerated by warm-start and active-set tricks.

License GPL-2

Repository CRAN

NeedsCompilation yes

RoxygenNote 7.1.1

LinkingTo Rcpp, RcppEigen

Imports Rcpp

Date/Publication 2021-07-01 07:10:27 UTC

Contents

SAM-package *Sparse Additive Modelling*

Description

The package SAM targets at high dimensional predictive modeling (regression and classification) for complex data analysis. SAM is short for sparse additive modeling, and adopts the computationally efficient basis spline technique. We solve the optimization problems by various computational algorithms including the block coordinate descent algorithm, fast iterative soft-thresholding algorithm, and newton method. The computation is further accelerated by warm-start and active-set tricks.

Details

Package:	SAM
Type:	Package
Version:	1.0.5
Date:	2014-02-11
License:	GPL-2

Author(s)

Tuo Zhao, Xingguo Li, Haoming Jiang, Han Liu, and Kathryn Roeder
Maintainers: Haoming Jiang<hjiang98@gatech.edu>;

References

P. Ravikumar, J. Lafferty, H.Liu and L. Wasserman. "Sparse Additive Models", *Journal of Royal Statistical Society: Series B*, 2009.
T. Zhao and H.Liu. "Sparse Additive Machine", *International Conference on Artificial Intelligence and Statistics*, 2012.

See Also

samQL,samHL,samLL,samEL

`plot.samEL` *Plot function for S3 class* `"samEL"`

Description

This function plots the regularization path (regularization parameter versus functional norm)

Usage

```
## S3 method for class 'samEL'
plot(x, ...)
```

Arguments

`x`	An object with S3 class `"samEL"`
`...`	System reserved (No specific usage)

Details

The horizontal axis is for the regularization parameters in log scale. The vertical axis is for the functional norm of each component.

See Also

samEL

`plot.samHL` *Plot function for S3 class* `"samHL"`

Description

This function plots the regularization path (regularization parameter versus functional norm)

Usage

```
## S3 method for class 'samHL'
plot(x, ...)
```

Arguments

`x`	An object with S3 class `"samHL"`
`...`	System reserved (No specific usage)

Details

The horizontal axis is for the regularization parameters in log scale. The vertical axis is for the functional norm of each component.

See Also

samHL

`plot.samLL` *Plot function for S3 class* `"samLL"`

Description

This function plots the regularization path (regularization parameter versus functional norm)

Usage

```
## S3 method for class 'samLL'
plot(x, ...)
```

Arguments

`x`	An object with S3 class `"samLL"`
`...`	System reserved (No specific usage)

Details

The horizontal axis is for the regularization parameters in log scale. The vertical axis is for the functional norm of each component.

See Also

samLL

plot.samQL *Plot function for S3 class "samQL"*

Description

This function plots the regularization path (regularization parameter versus functional norm)

Usage

```
## S3 method for class 'samQL'
plot(x, ...)
```

Arguments

x	An object with S3 class "samQL"
...	System reserved (No specific usage)

Details

The horizontal axis is for the regularization parameters in log scale. The vertical axis is for the functional norm of each component.

See Also

samQL

predict.samEL *Prediction function for S3 class "samEL"*

Description

Predict the labels for testing data.

Usage

```
## S3 method for class 'samEL'
predict(object, newdata, ...)
```

Arguments

object	An object with S3 class "samEL".
newdata	The testing dataset represented in a n by d matrix, where n is testing sample size and d is dimension.
...	System reserved (No specific usage)

Details

The testing dataset is rescale to the samELe range, and expanded by the samELe spline basis functions as the training data.

Value

`expectations`	Estimated expected counts also represented in a `n` by the length of `lambda` matrix, where `n` is testing sample size.

See Also

`samEL`

`predict.samHL` *Prediction function for S3 class* `"samHL"`

Description

Predict the labels for testing data.

Usage

```
## S3 method for class 'samHL'
predict(object, newdata, thol = 0, ...)
```

Arguments

`object`	An object with S3 class `"samHL"`.
`newdata`	The testing dataset represented in a `n` by `d` matrix, where `n` is testing sample size and `d` is dimension.
`thol`	The decision value threshold for prediction. The default value is 0.5
`...`	System reserved (No specific usage)

Details

The testing dataset is rescale to the samHLe range, and expanded by the samHLe spline basis functions as the training data.

Value

`values`	Predicted decision values also represented in a `n` by the length of `lambda` matrix, where `n` is testing sample size.
`labels`	Predicted labels also represented in a `n` by the length of `lambda` matrix, where `n` is testing sample size.

See Also

`samHL`

predict.samLL *Prediction function for S3 class "samLL"*

Description

Predict the labels for testing data.

Usage

```
## S3 method for class 'samLL'
predict(object, newdata, thol = 0.5, ...)
```

Arguments

object	An object with S3 class "samLL".
newdata	The testing dataset represented in a n by d matrix, where n is testing sample size and d is dimension.
thol	The decision value threshold for prediction. The default value is 0.5
...	System reserved (No specific usage)

Details

The testing dataset is rescale to the samLLe range, and expanded by the samLLe spline basis functions as the training data.

Value

probs	Estimated Posterior Probability for Prediction also represented in a n by the length of lambda matrix, where n is testing sample size.
labels	Predicted labels also represented in a n by the length of lambda matrix, where n is testing sample size.

See Also

samLL

predict.samQL *Prediction function for S3 class "samQL"*

Description

Predict the labels for testing data.

Usage

```
## S3 method for class 'samQL'
predict(object, newdata, ...)
```

Arguments

object	An object with S3 class "samQL".
newdata	The testing dataset represented in a n by d matrix, where n is testing sample size and d is dimension.
...	System reserved (No specific usage)

Details

The testing dataset is rescale to the samQLe range, and expanded by the samQLe spline basis functions as the training data.

Value

values	Predicted values also represented in a n by the length of lambda matrix, where n is testing sample size.

See Also

samQL

print.samEL *Printing function for S3 class "samEL"*

Description

Summarize the information of the object with S3 class samEL.

Usage

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

Arguments

x	An object with S3 class "samEL"
...	System reserved (No specific usage)

Details

The output includes length and d.f. of the regularization path.

See Also

samEL

---

print.samHL *Printing function for S3 class "samHL"*

Description

Summarize the information of the object with S3 class samHL.

Usage

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

Arguments

x	An object with S3 class "samHL"
...	System reserved (No specific usage)

Details

The output includes length and d.f. of the regularization path.

See Also

samHL

print.samLL *Printing function for S3 class "samLL"*

Description

Summarize the information of the object with S3 class samLL.

Usage

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

Arguments

x	An object with S3 class "samLL"
...	System reserved (No specific usage)

Details

The output includes length and d.f. of the regularization path.

See Also

samLL

print.samQL *Printing function for S3 class "samQL"*

Description

Summarize the information of the object with S3 class samQL.

Usage

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

Arguments

x	An object with S3 class "samQL"
...	System reserved (No specific usage)

Details

The output includes length and d.f. of the regularization path.

See Also

samQL

---

samEL *Training function of Sparse Additive Possion Regression*

Description

The log-linear model is learned using training data.

Usage

```
samEL(
  X,
  y,
  p = 3,
  lambda = NULL,
  nlambda = NULL,
  lambda.min.ratio = 0.25,
  thol = 1e-05,
  max.ite = 1e+05,
  regfunc = "L1"
)
```

Arguments

X	The n by d design matrix of the training set, where n is sample size and d is dimension.
y	The n-dimensional response vector of the training set, where n is sample size. Responses must be non-negative integers.
p	The number of basis spline functions. The default value is 3.
lambda	A user supplied lambda sequence. Typical usage is to have the program compute its own lambda sequence based on nlambda and lambda.min.ratio. Supplying a value of lambda overrides this. WARNING: use with care. Do not supply a single value for lambda. Supply instead a decreasing sequence of lambda values. samEL relies on its warms starts for speed, and its often faster to fit a whole path than compute a single fit.
nlambda	The number of lambda values. The default value is 20.
lambda.min.ratio	Smallest value for lambda, as a fraction of lambda.max, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). The default is 0.1.
thol	Stopping precision. The default value is 1e-5.
max.ite	The number of maximum iterations. The default value is 1e5.
regfunc	A string indicating the regularizer. The default value is "L1". You can also assign "MCP" or "SCAD" to it.

Details

We adopt various computational algorithms including the block coordinate descent, fast iterative soft-thresholding algorithm, and newton method. The computation is further accelerated by "warm-start" and "active-set" tricks.

Value

p	The number of basis spline functions used in training.
X.min	A vector with each entry corresponding to the minimum of each input variable. (Used for rescaling in testing)
X.ran	A vector with each entry corresponding to the range of each input variable. (Used for rescaling in testing)
lambda	A sequence of regularization parameter used in training.
w	The solution path matrix (d*p+1 by length of lambda) with each column corresponding to a regularization parameter. Since we use the basis expansion with the intercept, the length of each column is d*p+1.
df	The degree of freedom of the solution path (The number of non-zero component function)
knots	The p-1 by d matrix. Each column contains the knots applied to the corresponding variable.
Boundary.knots	The 2 by d matrix. Each column contains the boundary points applied to the corresponding variable.
func_norm	The functional norm matrix (d by length of lambda) with each column corresponds to a regularization parameter. Since we have d input variables, the length of each column is d.

See Also

SAM,plot.samEL,print.samEL,predict.samEL

Examples

```
## generating training data
n = 200
d = 100
X = 0.5*matrix(runif(n*d),n,d) + matrix(rep(0.5*runif(n),d),n,d)
u = exp(-2*sin(X[,1]) + X[,2]^2-1/3 + X[,3]-1/2 + exp(-X[,4])+exp(-1)-1+1)
y = rep(0,n)
for(i in 1:n) y[i] = rpois(1,u[i])

## Training
out.trn = samEL(X,y)
out.trn

## plotting solution path
plot(out.trn)
```

```
## generating testing data
nt = 1000
Xt = 0.5*matrix(runif(nt*d),nt,d) + matrix(rep(0.5*runif(nt),d),nt,d)
ut = exp(-2*sin(Xt[,1]) + Xt[,2]^2-1/3 + Xt[,3]-1/2 + exp(-Xt[,4])+exp(-1)-1+1)
yt = rep(0,nt)
for(i in 1:nt) yt[i] = rpois(1,ut[i])

## predicting response
out.tst = predict(out.trn,Xt)
```

---

samHL *Training function of Sparse Additive Machine*

Description

The classifier is learned using training data.

Usage

```
samHL(
  X,
  y,
  p = 3,
  lambda = NULL,
  nlambda = NULL,
  lambda.min.ratio = 0.4,
  thol = 1e-05,
  mu = 0.05,
  max.ite = 1e+05,
  w = NULL
)
```

Arguments

X	The n by d design matrix of the training set, where n is sample size and d is dimension.
y	The n-dimensional label vector of the training set, where n is sample size. Labels must be coded in 1 and 0.
p	The number of basis spline functions. The default value is 3.
lambda	A user supplied lambda sequence. Typical usage is to have the program compute its own lambda sequence based on nlambda and lambda.min.ratio. Supplying a value of lambda overrides this. WARNING: use with care. Do not supply a single value for lambda. Supply instead a decreasing sequence of lambda values. samHL relies on its warms starts for speed, and its often faster to fit a whole path than compute a single fit.
nlambda	The number of lambda values. The default value is 20.

`lambda.min.ratio`
: Smallest value for lambda, as a fraction of lambda.max, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). The default is 0.4.

`thol`
: Stopping precision. The default value is 1e-5.

`mu`
: Smoothing parameter used in approximate the Hinge Loss. The default value is 0.05.

`max.ite`
: The number of maximum iterations. The default value is 1e5.

`w`
: The `n`-dimensional positive vector. It is the weight of each entry in the weighted loss. The default value is 1 for all entries.

Details

We adopt various computational algorithms including the block coordinate descent, fast iterative soft-thresholding algorithm, and newton method. The computation is further accelerated by "warm-start" and "active-set" tricks.

Value

`p`
: The number of basis spline functions used in training.

`X.min`
: A vector with each entry corresponding to the minimum of each input variable. (Used for rescaling in testing)

`X.ran`
: A vector with each entry corresponding to the range of each input variable. (Used for rescaling in testing)

`lambda`
: A sequence of regularization parameter used in training.

`w`
: The solution path matrix (`d*p+1` by length of `lambda`) with each column corresponding to a regularization parameter. Since we use the basis expansion with the intercept, the length of each column is `d*p+1`.

`df`
: The degree of freedom of the solution path (The number of non-zero component function)

`knots`
: The `p-1` by `d` matrix. Each column contains the knots applied to the corresponding variable.

`Boundary.knots`
: The `2` by `d` matrix. Each column contains the boundary points applied to the corresponding variable.

`func_norm`
: The functional norm matrix (`d` by length of `lambda`) with each column corresponds to a regularization parameter. Since we have `d` input variables, the length of each column is `d`.

See Also

`SAM`,`plot.samHL`,`print.samHL`,`predict.samHL`

Examples

```
## generating training data
n = 200
d = 100
X = 0.5*matrix(runif(n*d),n,d) + matrix(rep(0.5*runif(n),d),n,d)
y = sign(((X[,1]-0.5)^2 + (X[,2]-0.5)^2)-0.06)

## flipping about 5 percent of y
y = y*sign(runif(n)-0.05)

## Training
out.trn = samHL(X,y)
out.trn

## plotting solution path
plot(out.trn)

## generating testing data
nt = 1000
Xt = 0.5*matrix(runif(nt*d),nt,d) + matrix(rep(0.5*runif(nt),d),nt,d)

yt = sign(((Xt[,1]-0.5)^2 + (Xt[,2]-0.5)^2)-0.06)

## flipping about 5 percent of y
yt = yt*sign(runif(nt)-0.05)

## predicting response
out.tst = predict(out.trn,Xt)
```

---

samLL *Training function of Sparse Additive Logistic Regression*

Description

The logistic model is learned using training data.

Usage

```
samLL(
  X,
  y,
  p = 3,
  lambda = NULL,
  nlambda = NULL,
  lambda.min.ratio = 0.1,
  thol = 1e-05,
  max.ite = 1e+05,
  regfunc = "L1"
)
```

Arguments

X	The n by d design matrix of the training set, where n is sample size and d is dimension.
y	The n-dimensional label vector of the training set, where n is sample size. Labels must be coded in 1 and 0.
p	The number of basis spline functions. The default value is 3.
lambda	A user supplied lambda sequence. Typical usage is to have the program compute its own lambda sequence based on nlambda and lambda.min.ratio. Supplying a value of lambda overrides this. WARNING: use with care. Do not supply a single value for lambda. Supply instead a decreasing sequence of lambda values. samLL relies on its warms starts for speed, and its often faster to fit a whole path than compute a single fit.
nlambda	The number of lambda values. The default value is 20.
lambda.min.ratio	Smallest value for lambda, as a fraction of lambda.max, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). The default is 0.1.
thol	Stopping precision. The default value is 1e-5.
max.ite	The number of maximum iterations. The default value is 1e5.
regfunc	A string indicating the regularizer. The default value is "L1". You can also assign "MCP" or "SCAD" to it.

Details

We adopt various computational algorithms including the block coordinate descent, fast iterative soft-thresholding algorithm, and newton method. The computation is further accelerated by "warm-start" and "active-set" tricks.

Value

p	The number of basis spline functions used in training.
X.min	A vector with each entry corresponding to the minimum of each input variable. (Used for rescaling in testing)
X.ran	A vector with each entry corresponding to the range of each input variable. (Used for rescaling in testing)
lambda	A sequence of regularization parameter used in training.
w	The solution path matrix (d*p+1 by length of lambda) with each column corresponding to a regularization parameter. Since we use the basis expansion with the intercept, the length of each column is d*p+1.
df	The degree of freedom of the solution path (The number of non-zero component function)
knots	The p-1 by d matrix. Each column contains the knots applied to the corresponding variable.
Boundary.knots	The 2 by d matrix. Each column contains the boundary points applied to the corresponding variable.

func_norm The functional norm matrix (d by length of lambda) with each column corresponds to a regularization parameter. Since we have d input variables, the length of each column is d.

See Also

SAM,plot.samLL,print.samLL,predict.samLL

Examples

```
## generating training data
n = 200
d = 100
X = 0.5*matrix(runif(n*d),n,d) + matrix(rep(0.5*runif(n),d),n,d)
y = sign(((X[,1]-0.5)^2 + (X[,2]-0.5)^2)-0.06)

## flipping about 5 percent of y
y = y*sign(runif(n)-0.05)
y = sign(y==1)

## Training
out.trn = samLL(X,y)
out.trn

## plotting solution path
plot(out.trn)

## generating testing data
nt = 1000
Xt = 0.5*matrix(runif(nt*d),nt,d) + matrix(rep(0.5*runif(nt),d),nt,d)

yt = sign(((Xt[,1]-0.5)^2 + (Xt[,2]-0.5)^2)-0.06)

## flipping about 5 percent of y
yt = yt*sign(runif(nt)-0.05)
yt = sign(yt==1)

## predicting response
out.tst = predict(out.trn,Xt)
```

---

samQL *Training function of Sparse Additive Models*

Description

The regression model is learned using training data.

Usage

```
samQL(
  X,
  y,
  p = 3,
  lambda = NULL,
  nlambda = NULL,
  lambda.min.ratio = 0.005,
  thol = 1e-05,
  max.ite = 1e+05,
  regfunc = "L1"
)
```

Arguments

X	The n by d design matrix of the training set, where n is sample size and d is dimension.
y	The n-dimensional response vector of the training set, where n is sample size.
p	The number of basis spline functions. The default value is 3.
lambda	A user supplied lambda sequence. Typical usage is to have the program compute its own lambda sequence based on nlambda and lambda.min.ratio. Supplying a value of lambda overrides this. WARNING: use with care. Do not supply a single value for lambda. Supply instead a decreasing sequence of lambda values. samQL relies on its warms starts for speed, and its often faster to fit a whole path than compute a single fit.
nlambda	The number of lambda values. The default value is 30.
lambda.min.ratio	Smallest value for lambda, as a fraction of lambda.max, the (data derived) entry value (i.e. the smallest value for which all coefficients are zero). The default is 5e-3.
thol	Stopping precision. The default value is 1e-5.
max.ite	The number of maximum iterations. The default value is 1e5.
regfunc	A string indicating the regularizer. The default value is "L1". You can also assign "MCP" or "SCAD" to it.

Details

We adopt various computational algorithms including the block coordinate descent, fast iterative soft-thresholding algorithm, and newton method. The computation is further accelerated by "warm-start" and "active-set" tricks.

Value

p	The number of basis spline functions used in training.
X.min	A vector with each entry corresponding to the minimum of each input variable. (Used for rescaling in testing)

X.ran	A vector with each entry corresponding to the range of each input variable. (Used for rescaling in testing)
lambda	A sequence of regularization parameter used in training.
w	The solution path matrix (d*p by length of lambda) with each column corresponding to a regularization parameter. Since we use the basis expansion, the length of each column is d*p+1.
intercept	The solution path of the intercept.
df	The degree of freedom of the solution path (The number of non-zero component function)
knots	The p-1 by d matrix. Each column contains the knots applied to the corresponding variable.
Boundary.knots	The 2 by d matrix. Each column contains the boundary points applied to the corresponding variable.
func_norm	The functional norm matrix (d by length of lambda) with each column corresponds to a regularization parameter. Since we have d input variables, the length of each column is d.
sse	Sums of square errors of the solution path.

See Also

SAM,plot.samQL,print.samQL,predict.samQL

Examples

```
## generating training data
n = 100
d = 500
X = 0.5*matrix(runif(n*d),n,d) + matrix(rep(0.5*runif(n),d),n,d)

## generating response
y = -2*sin(X[,1]) + X[,2]^2-1/3 + X[,3]-1/2 + exp(-X[,4])+exp(-1)-1

## Training
out.trn = samQL(X,y)
out.trn

## plotting solution path
plot(out.trn)

## generating testing data
nt = 1000
Xt = 0.5*matrix(runif(nt*d),nt,d) + matrix(rep(0.5*runif(nt),d),nt,d)

yt = -2*sin(Xt[,1]) + Xt[,2]^2-1/3 + Xt[,3]-1/2 + exp(-Xt[,4])+exp(-1)-1

## predicting response
out.tst = predict(out.trn,Xt)
```

Index