Package 'SAM'

February 19, 2026

Type Package

Title Sparse Additive Modelling

Version 1.2

Maintainer Tuo Zhao <tourzhao@gatech.edu>

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

Encoding UTF-8

NeedsCompilation yes

RoxygenNote 7.3.3

LinkingTo Rcpp, RcppEigen

Author Haoming Jiang [aut],
Yukun Ma [aut],
Han Liu [aut],
Kathryn Roeder [aut],
Xingguo Li [aut],
Tuo Zhao [aut, cre]

Repository CRAN

Date/Publication 2026-02-19 10:50:02 UTC

Contents

`SAM-package` *Sparse Additive Modelling*

Description

SAM provides sparse additive models for high-dimensional prediction tasks (regression and classification). It uses spline basis expansion and efficient optimization routines to compute full regularization paths.

Details

The package exposes four model families:

- `samQL`: quadratic-loss sparse additive regression.
- `samLL`: logistic-loss sparse additive classification.
- `samHL`: hinge-loss sparse additive classification.
- `samEL`: Poisson-loss sparse additive regression.

All models share a common spline representation and return regularization paths, allowing model selection after one fit.

Author(s)

Tuo Zhao, Xingguo Li, Haoming Jiang, Han Liu, and Kathryn Roeder
Maintainer: Tuo Zhao <tourzhao@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

Plot the regularization path (regularization parameter versus functional norm).

Usage

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

Arguments

`x`	An object with S3 class `"samEL"`
`...`	Additional arguments passed to methods; currently unused.

Details

The x-axis shows regularization parameters on a log scale. The y-axis shows the functional norm of each component function.

See Also

samEL

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

Description

Plot the regularization path (regularization parameter versus functional norm).

Usage

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

Arguments

`x`	An object with S3 class `"samHL"`
`...`	Additional arguments passed to methods; currently unused.

Details

The x-axis shows regularization parameters on a log scale. The y-axis shows the functional norm of each component function.

See Also

`samHL`

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

Description

Plot the regularization path (regularization parameter versus functional norm).

Usage

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

Arguments

`x`	An object with S3 class `"samLL"`
`...`	Additional arguments passed to methods; currently unused.

Details

The x-axis shows regularization parameters on a log scale. The y-axis shows the functional norm of each component function.

See Also

`samLL`

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

Description

Plot the regularization path (regularization parameter versus functional norm).

Usage

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

Arguments

`x`	An object with S3 class `"samQL"`
`...`	Additional arguments passed to methods; currently unused.

Details

The x-axis shows regularization parameters on a log scale. The y-axis shows the functional norm of each component function.

See Also

`samQL`

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

Description

Predict expected counts for test data.

Usage

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

Arguments

`object`	An object with S3 class `"samEL"`.
`newdata`	Numeric test matrix with `n` rows and `d` columns.
`...`	Additional arguments passed to methods; currently unused.

Details

The test matrix is rescaled using the training `X.min`/`X.ran`, truncated to `[0, 1]`, and expanded with the same spline basis used during training.

Value

`expectations`	Estimated expected counts as an `n` by `length(lambda)` matrix.
`expectation`	Alias of `expectations` kept for backward compatibility.

See Also

`samEL`

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

Description

Predict decision values and class labels for test data.

Usage

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

Arguments

`object`	An object with S3 class `"samHL"`.
`newdata`	Numeric test matrix with `n` rows and `d` columns.
`thol`	Decision-value threshold used to convert scores to labels. The default value is `0`.
`...`	Additional arguments passed to methods; currently unused.

Details

The test matrix is rescaled using the training `X.min`/`X.ran`, truncated to `[0, 1]`, and expanded with the same spline basis used during training.

Value

`values`	Predicted decision values as an `n` by `length(lambda)` matrix.
`labels`	Predicted class labels (`-1`/`1`) as an `n` by `length(lambda)` matrix.

See Also

`samHL`

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

Description

Predict class probabilities and labels for test data.

Usage

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

Arguments

object	An object with S3 class "samLL".
newdata	Numeric test matrix with n rows and d columns.
thol	Decision-value threshold used to convert probabilities to labels. The default value is 0.5.
...	Additional arguments passed to methods; currently unused.

Details

The test matrix is rescaled using the training X.min/X.ran, truncated to [0, 1], and expanded with the same spline basis used during training.

Value

probs	Estimated posterior probabilities as an n by length(lambda) matrix.
labels	Predicted class labels (0/1) as an n by length(lambda) matrix.

See Also

samLL

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

Description

Predict responses for test data.

Usage

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

Arguments

`object`	An object with S3 class `"samQL"`.
`newdata`	Numeric test matrix with `n` rows and `d` columns.
`...`	Additional arguments passed to methods; currently unused.

Details

The test matrix is rescaled using the training `X.min`/`X.ran`, truncated to `[0, 1]`, and expanded with the same spline basis used during training.

Value

`values`	Predicted responses as an `n` by `length(lambda)` matrix.

See Also

`samQL`

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

Description

Print a summary of an object of class `"samEL"`.

Usage

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

Arguments

`x`	An object with S3 class `"samEL"`
`...`	Additional arguments passed to methods; currently unused.

Details

The output includes the regularization path length and its degrees of freedom.

See Also

`samEL`

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

Description

Print a summary of an object of class `"samHL"`.

Usage

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

Arguments

`x`	An object with S3 class `"samHL"`
`...`	Additional arguments passed to methods; currently unused.

Details

The output includes the regularization path length and its degrees of freedom.

See Also

`samHL`

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

Description

Print a summary of an object of class `"samLL"`.

Usage

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

Arguments

`x`	An object with S3 class `"samLL"`
`...`	Additional arguments passed to methods; currently unused.

Details

The output includes the regularization path length and its degrees of freedom.

See Also

samLL

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

Description

Print a summary of an object of class `"samQL"`.

Usage

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

Arguments

`x`	An object with S3 class `"samQL"`
`...`	Additional arguments passed to methods; currently unused.

Details

The output includes the regularization path length and its degrees of freedom.

See Also

samQL

`samEL` *Training function of Sparse Additive Poisson Regression*

Description

Fit a sparse additive Poisson regression model on 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`	Numeric training matrix with `n` rows (samples) and `d` columns (features).
`y`	Response vector of length `n`; values must be non-negative integers.
`p`	The number of basis spline functions. The default value is 3.
`lambda`	Optional user-supplied regularization sequence. If provided, use a decreasing sequence; warm starts are used along the path and are usually much faster than fitting a single value.
`nlambda`	The number of lambda values. The default value is 20.
`lambda.min.ratio`	Smallest lambda as a fraction of `lambda.max` (the smallest value that keeps all component functions at zero). The default is `0.25`.
`thol`	Stopping tolerance. The default value is `1e-5`.
`max.ite`	Maximum number of 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

The solver combines block coordinate descent, fast iterative soft-thresholding, and Newton updates. Computation is accelerated by warm starts and active-set screening.

Value

`p`	The number of basis spline functions used in training.
`X.min`	Per-feature minimums from training data (used to rescale test data).
`X.ran`	Per-feature ranges from training data (used to rescale test data).
`lambda`	Sequence of regularization parameters used in training.
`w`	Solution path matrix with size `d*p+1` by `length(lambda)`; each column corresponds to one regularization parameter.
`df`	Degrees of freedom along the solution path (number of non-zero component functions).
`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`	Functional norm matrix (`d` by `length(lambda)`); each column corresponds to one regularization parameter.

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 Hinge-Loss Classifier*

Description

Fit a sparse additive classifier with hinge loss.

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`	Numeric training matrix with `n` rows (samples) and `d` columns (features).
`y`	Training labels of length `n`. Labels must be coded as `-1` and `1`.
`p`	The number of basis spline functions. The default value is 3.
`lambda`	Optional user-supplied regularization sequence. If provided, use a decreasing sequence; warm starts are used along the path and are usually much faster than fitting a single value.
`nlambda`	The number of lambda values. The default value is 20.
`lambda.min.ratio`	Smallest lambda as a fraction of `lambda.max` (the smallest value that keeps all component functions at zero). The default is `0.4`.
`thol`	Stopping tolerance. The default value is `1e-5`.
`mu`	Smoothing parameter used to approximate hinge loss. The default value is `0.05`.
`max.ite`	Maximum number of iterations. The default value is `1e5`.
`w`	Optional positive observation weights of length `n`. The default is `1` for all observations.

Details

The solver combines block coordinate descent, fast iterative soft-thresholding, and Newton updates. Computation is accelerated by warm starts and active-set screening.

Value

`p`	The number of basis spline functions used in training.
`X.min`	Per-feature minimums from training data (used to rescale test data).
`X.ran`	Per-feature ranges from training data (used to rescale test data).
`lambda`	Sequence of regularization parameters used in training.
`w`	Solution path matrix with size `d*p+1` by `length(lambda)`; each column corresponds to one regularization parameter.
`df`	Degrees of freedom along the solution path (number of non-zero component functions).
`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`	Functional norm matrix (`d` by `length(lambda)`); each column corresponds to one regularization parameter.

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

Fit a sparse additive logistic regression model on 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`	Numeric training matrix with `n` rows (samples) and `d` columns (features).
`y`	Binary training labels of length `n`. Labels must be coded as `0` and `1`.
`p`	The number of basis spline functions. The default value is 3.
`lambda`	Optional user-supplied regularization sequence. If provided, use a decreasing sequence; warm starts are used along the path and are usually much faster than fitting a single value.
`nlambda`	The number of lambda values. The default value is 20.
`lambda.min.ratio`	Smallest lambda as a fraction of `lambda.max` (the smallest value that keeps all component functions at zero). The default is `0.1`.
`thol`	Stopping tolerance. The default value is `1e-5`.
`max.ite`	Maximum number of 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

The solver combines block coordinate descent, fast iterative soft-thresholding, and Newton updates. Computation is accelerated by warm starts and active-set screening.

Value

`p`	The number of basis spline functions used in training.
`X.min`	Per-feature minimums from training data (used to rescale test data).
`X.ran`	Per-feature ranges from training data (used to rescale test data).
`lambda`	Sequence of regularization parameters used in training.
`w`	Solution path matrix with size `d*p+1` by `length(lambda)`; each column corresponds to one regularization parameter.
`df`	Degrees of freedom along the solution path (number of non-zero component functions).
`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`	Functional norm matrix (`d` by `length(lambda)`); each column corresponds to one regularization parameter.

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 Regression with Quadratic Loss*

Description

Fit a sparse additive regression model with quadratic loss.

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`	Numeric training matrix with `n` rows (samples) and `d` columns (features).
`y`	Numeric response vector of length `n`.
`p`	The number of basis spline functions. The default value is 3.
`lambda`	Optional user-supplied regularization sequence. If provided, use a decreasing sequence; warm starts are used along the path and are usually much faster than fitting a single value.
`nlambda`	The number of lambda values. The default value is 30.
`lambda.min.ratio`	Smallest lambda as a fraction of `lambda.max` (the smallest value that keeps all component functions at zero). The default is `5e-3`.
`thol`	Stopping tolerance. The default value is `1e-5`.
`max.ite`	Maximum number of 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

The solver combines block coordinate descent, fast iterative soft-thresholding, and Newton updates. Computation is accelerated by warm starts and active-set screening.

Value

`p`	The number of basis spline functions used in training.
`X.min`	Per-feature minimums from training data (used to rescale test data).
`X.ran`	Per-feature ranges from training data (used to rescale test data).
`lambda`	Sequence of regularization parameters used in training.
`w`	Solution path matrix with size `d*p` by `length(lambda)`; each column corresponds to one regularization parameter.
`intercept`	The solution path of the intercept.
`df`	Degrees of freedom along the solution path (number of non-zero component functions).
`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`	Functional norm matrix (`d` by `length(lambda)`); each column corresponds to one regularization parameter.
`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