Package ‘AHM’

July 28, 2019

Type Package

Title Additive Heredity Model: Method for the Mixture-of-Mixtures Experiments

Version 1.0.1

Maintainer Sumin Shen <sumin@vt.edu>

Description An implementation of the additive heredity model for the mixture-of-mixtures experiments of Shen et al. (2019) in Technometrics <doi:10.1080/00401706.2019.1630010>. The additive heredity model considers an additive structure to inherently connect the major components with the minor components. The additive heredity model has a meaningful interpretation for the estimated model because of the hierarchical and heredity principles applied and the nonnegative garrote technique used for variable selection.

License GPL-3

Encoding UTF-8

LazyData true

Imports mixexp, plgp, devtools, dplyr, tibble, tidyr, Matrix

Depends R (>= 2.10), quadprog, glmnet

Suggests knitr, rmarkdown, partitions

VignetteBuilder knitr

BuildVignettes yes

RoxygenNote 6.0.1

NeedsCompilation no

Author Sumin Shen [aut, cre],
Lulu Kang [aut],
Xinwei Deng [aut]

Repository CRAN

Date/Publication 2019-07-28 09:00:05 UTC
The function `ahm` computes the proposed additive heredity model.

**Description**

This is one of the main functions. The function `ahm` computes the proposed additive heredity model.

**Usage**

```r
ahm(y, x, num_major = 3, dist_minor = c(2, 2, 1), type = "weak",
    alpha = 0, lambda_seq = seq(0, 5, 0.01), nfolds = NULL,
    mapping_type = c("power"), powerh = 0, rep_gcv = 100)
```

**Arguments**

- `y` numeric vector
- `x` data.frame Note the column names of the `x` should be in the order of major components, minor components, and no interactions are needed.
- `num_major` number of major components
- `dist_minor` the allocation of number of minor components nested under major components
- `type` heredity type, weak heredity is the current support type
- `alpha` 0 is for the ridge in glmnet https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html
**check_col_correlation**

- **lambda_seq**: a numeric vector for the options of lambda used in ridge regression for estimating the initials
- **nfolds**: used in cv.glmnet for initial value of parameters in the non-negative garrotte method
- **mapping_type**: the form of the coefficient function of major components in front of corresponding minor terms. Currently only support “power”
- **powerh**: the power parameter used for the power function
- **rep_gcv**: the number of choices of tuning parameter used in the GCV selection

**Value**

Return a list

**Examples**

```r
data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1","c2","c3","x11","x12","x21","x22")]
y = data_fat[,1]
out = ahm (y, x, num_major = 3, dist_minor = c(2,2,1),
          type = "weak", alpha=0, lambda_seq=seq(0,5,0.01), nfold = NULL,
          mapping_type = c("power"), powerh = h_tmp,
          rep_gcv=100)
summary(out)
```

---

**check_col_correlation**

*Check column correlations*

**Description**

Check column correlations

**Usage**

`check_col_correlation(dat)`

**Arguments**

- **dat**: data.frame

**Examples**

```r
data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1","c2","c3","x11","x12","x21","x22")]
check_col_correlation (dat=x)
```
coating  
*Photoresist-coating experiment data*

**Description**

Photoresist-coating experiment data

**Usage**

```r
data(coating)
```

**Format**

data.frame

**References**


**Examples**

```r
data(coating)
print(coating)
```

---

coef.ahm  
*Coefficient method for the fitted ahm object*

**Description**

Coefficient method for the fitted ahm object

**Usage**

```r
## S3 method for class 'ahm'
coef(object, ...)
```

**Arguments**

<table>
<thead>
<tr>
<th>object</th>
<th>ahm object</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>not used</td>
</tr>
</tbody>
</table>

---

\(^1\)https://www.tandfonline.com/doi/abs/10.1080/00401706.1998.10485481
Value

a numerical vector

Examples

data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1","c2","c3","x11","x12","x21","x22")]
y = data_fat[,1]
out = ahm(y, x, num_major = 3, dist_minor = c(2,2,1),
    type = "weak", alpha=0, lambda_seq=seq(0,5,0.01), nfold = NULL,
    mapping_type = c("power"), powerh = h_tmp,
    rep_gcv=100)
coef(out)

calc_stats_fat

Description

Coefficient method for the fitted cv.ahm object

Usage

## S3 method for class 'cv.ahm'
coef(object, metric = "mse", ...)

Arguments

object 

   cv.ahm object

metric 

   "mse" or "aicc"

... 

   not used

Value

a numerical vector

Examples

data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1","c2","c3","x11","x12","x21","x22")]
y = data_fat[,1]
powerh_path = round(seq(0.001,2,length.out =15),3)
num_major = 3; dist_minor = c(2,2,1)
res = cv.ahm (y, x, powerh_path=powerh_path, metric = "mse", num_major, dist_minor, type = "weak", alpha=0, lambda_seq=seq(0,5,0.01), nfolds=NULL, mapping_type = c("power"), rep_gcv=100)
coefficients = coef(res)

---

compute_aicc  compute AICc

Description

compute AICc

Usage

compute_aicc(rss, n, p, type = "AICc")

Arguments

rss  residual sum of squares
n  number of observation
p  number of nonzero parameters
type  character "AICc"

References

Calculating AIC “by hand” in R in Stack Overflow\(^2\)

Examples

compute_aicc (rss=10, n=30, p=6, type = "AICc")

\(^2\)https://stats.stackexchange.com/questions/87345/calculating-aic-by-hand-in-r/
cv.ahm

This is one of the main functions. It perform the cross validation on ahm models to select the optimal setting of hyper parameter h

Description

This is one of the main functions. It perform the cross validation on ahm models to select the optimal setting of hyper parameter h

Usage

cv.ahm(y, x, powerh_path = NULL, metric = c("mse", "AICc"), num_major = 3, dist_minor = c(2, 2, 1), type = "weak", alpha = 0, lambda_seq = seq(0, 5, 0.01), nfolds = NULL, mapping_type = c("power"), rep_gcv = 100)

Arguments

y numeric vector
x data.frame Note the column names of the x should be in the order of major components, minor components, and no interactions between major or minor components are needed.
powerh_path if NULL, then the default is the vector: round(seq(0.001,2,length.out =15),3)
metric "mse" or "AICc" the metric used in cross validation where the minimum is selected as the optimal
num_major number of major components
dist_minor the allocation of number of minor components nested under major components
type heredity type, weak heredity is the current support type
alpha 0 is for the ridge in glmnet https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html
lambda_seq a numeric vector for the options of lambda used in ridge regression for estimating the initials
nfolds used in cv.glmnet for initial value of parameters in the non-negative garrote method
mapping_type the form of the coefficient function of major components in front of corresponding minor terms. Currently only support "power"
rep_gcv the number of choices of tuning parameter used in the GCV selection

Value

Return a list
Examples

data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1","c2","c3","x11","x12","x21","x22")]
y = data_fat[,1]
powerh_path = round(seq(0.001,2,length.out =15),3)
num_major = 3; dist_minor = c(2,2,1)
res = cv.ahm (y, x, powerh_path=powerh_path, metric = "mse", num_major, dist_minor, type = "",
, alpha=0, lambda_seq=seq(0,5,0.01), nfolds=NULL, mapping_type = c("power"), rep_gcv=100)
object = res$metric_mse

---

design_simplex_centroid_design_3_major_component

*Design points for the simplex centroid design with 3 components*

Description

Design points for the simplex centroid design with 3 components

Usage

data(design_simplex_centroid_design_3_major_component)

Format

data.frame

Examples

data(design_simplex_centroid_design_3_major_component)
print(design_simplex_centroid_design_3_major_component)

---

enlist

*Create a list*

Description

Create a list

Usage

enlist(...)
**expand_interactions**

**Arguments**

... object to be included as elements in the list

**Examples**

```r
data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1","c2","c3","x11","x12","x21","x22")]
expand_interactions (dat=x, sel_names=c("c1", "c2", "c3"))
```

---

**find_condition_num**  
*Compute the conditional number of design matrix*

**Description**

Compute the conditional number of design matrix

**Usage**

```r
find_condition_num(x)
```
mapping_function

Arguments

x matrix to be used in svd

Examples

data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1","c2","c3","x11","x12","x21","x22")]
find_condition_num (x)

mapping_function Mapping function is a function to add the functional coefficients of major components in front of minor components terms

Description

Mapping function is a function to add the functional coefficients of major components in front of minor components terms

Usage

mapping_function(x, num_major = 3, dist_minor = c(2, 2, 1),
mapping_type = c("power"), powerh = 0)

Arguments

x data.frame Note the column names of the x should be in the order of major components, minor components, and no interactions are needed.
num_major number of major components
dist_minor the allocation of number of minor components nested under major components
mapping_type the form of the coefficient function of major components in front of corresponding minor terms. Currently only support "power"
powerh the power parameter used for the power function

Value
data frame

Examples

data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1","c2","c3","x11","x12","x21","x22")]
mapping_function(x=x, num_major=3, dist_minor=c(2,2,1), mapping_type = c("power"), powerh=0)
The mymaximin function generates the matrix of maximin design points. It uses the simplex centroid design as the base design, then in a stochastics way sample the candidate design points generated by the function partition.

Description

This method is modified based on Prof. Bobby Gramacy’s Computer Experiment lecture at Virginia Tech. Prof. Gramacy’s lecture website

Usage

mymaximin(pool, n = 50, m = 3, iter = 1e+05, Xorig = NULL)

Arguments

pool, partition the base design points provided to the function
n numeric, sample size
m numeric, 3 stands for 3 components, i.e. c1, c2, and c3
iter numeric, iterations used in the stochastic sampling
Xorig matrix, initial design points

Value

Return a matrix of the design points for the major components

Examples

# The case of unconstrained experiments
library(mixexp)
num_size = 8 # num points in the design for the major component
Xorig = as.matrix(SCD(3))
# all possible combinations sum to 1
pool_3d = partitions::compositions(1000, 3,include.zero = TRUE)/1000
res_C = mymaximin(pool=pool_3d, n=num_size, m=3, iter=1e5, Xorig=Xorig)
DesignPoints(res_C,cornerlabs = c("c3","c2","c1"),axislabs=c("c1","c2","c3"))

# The case of constrained experiments
library(mixexp)
num_size = 8 # num points in the design for the major component
# all possible combinations sum to 1
pool_3d = partitions::compositions(1000, 3,include.zero = TRUE)/1000
c1_min=0.2
c1_max=0.45

3http://bobby.gramacy.com/teaching/
predict.ahm

Description

Predict method for the fitted ahm object

Usage

## S3 method for class 'ahm'
predict(object, newx, ...)  

Arguments

- **object**: ahm object
- **newx**: Matrix of new values for x at which predictions are to be made.
- **...**: not used

Value

predicted value(s) at newx
predict.cv.ahm

Examples

data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1","c2","c3","x11","x12","x21","x22")]
y = data_fat[,1]
out = ahm (y, x, num_major = 3, dist_minor = c(2,2,1),
        type = "weak", alpha=0, lambda_seq=seq(0,5,0.01), nfold = NULL,
        mapping_type = c("power"), powerh = h_tmp,
        rep_gcv=100)
predict(out)

predict.cv.ahm

Predict method for the fitted cv.ahm object

Description

Predict method for the fitted cv.ahm object

Usage

## S3 method for class 'cv.ahm'
predict(object, newx, metric = "mse", ...)

Arguments

object          cv.ahm object
newx            Matrix of new values for x at which predictions are to be made.
metric          "mse" or "aicc"
...             not used

Value

Return a list

Examples

data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1","c2","c3","x11","x12","x21","x22")]
y = data_fat[,1]
powerh_path = round(seq(0.001,2,length.out =15),3)
num_major = 3; dist_minor = c(2,2,1)
res = cv.ahm (y, x, powerh_path=powerh_path, metric = "mse", num_major, dist_minor, type = "weak",
             alpha=0, lambda_seq=seq(0,5,0.01), nfolds=NULL, mapping_type = c("power"), rep_gcv=100)
pred = predict(res)

**prises_candidates2search**

*The candidate search points in the nonlinear optimization for the optimal value in the Pringles experiment*

**Description**

The candidate search points in the nonlinear optimization for the optimal value in the Pringles experiment

**Usage**

data(prises_candidates2search)

**Format**

matrix

**Examples**

data(prises_candidates2search)
print(prises_candidates2search)

**prises_fat**

*Pringles experiment data set with the percent of Fat as the response*

**Description**

Pringles experiment data set with the percent of Fat as the response

**Usage**

data(prises_fat)

**Format**

data.frame

**References**


⁴https://www.tandfonline.com/doi/abs/10.1198/TECH.2011.08132
**pringles_hardness**

**Examples**

```r
data(pringles_fat)
print(pringles_fat)
```

---

**pringles_hardness**  
*Pringles experiment data set with the Hardness as the response*

**Description**

Pringles experiment data set with the Hardness as the response

**Usage**

```r
data(pringles_hardness)
```

**Format**

data.frame

**References**


**Examples**

```r
data(pringles_hardness)
print(pringles_hardness)
```

---

**summary.ahm**  
*Summary method for the fitted ahm object*

**Description**

Summary method for the fitted ahm object

**Usage**

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

**Arguments**

- `object`  
  fitted ahm object

- `...`  
  not used

⁵https://www.tandfonline.com/doi/abs/10.1198/TECH.2011.08132
Examples

data("pringles_fat")
data_fat = pringles_fat
h_tmp = 1.3
x = data_fat[,c("c1","c2","c3","x11","x12","x21","x22")]
y = data_fat[,1]
out = ahm (y, x, num_major = 3, dist_minor = c(2,2,1),
        type = "weak", alpha=0, lambda_seq=seq(0,5,0.01), nfold = NULL,
        mapping_type = c("power"), powerh = h_tmp,
        rep_gcv=100)
summary(out)