Package 'SmartSifter'

September 14, 2016

Title Online Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms

Version 0.1.0

Date 2016-09-14

Author Lizhen Nie <nie_lizhen@yahoo.com>

Maintainer Lizhen Nie <nie_lizhen@yahoo.com>

Description Addressing the problem of outlier detection from the viewpoint of statistical learning theory. This method is proposed by Yamanishi, K., Takeuchi, J., Williams, G. et al. (2004) <DOI:10.1023/B:DAMI.0000023676.72185.7c>. It learns the probabilistic model (using a finite mixture model) through an on-line unsupervised process. After each datum is input, a score will be given with a high one indicating a high possibility of being a statistical outlier.

Depends R (>= 3.3.1)

Imports mvtnorm, rootSolve

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 5.0.1

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2016-09-14 18:50:50

R topics documented:

`delta` *delta*

Description

the delta function

Usage

```
delta(a, b)
```

Arguments

a	A number.
b	A number.

Value

The function delta(a,b), if a == b, return 1; else, return 0.

`HellingerScore` *HellingerScore*

Description

calculates the Hellinger score after inputting sample y (can be more than one)

Usage

```
HellingerScore(y, param = TRUE, smart, const, initial)
```

Arguments

y	A matrix, the new sample to be input.
param	A logical scalar, if TRUE, the model is in parametric version, otherwise, a non-parametric one.
smart	A matrix, stores all the parameters over the continuous domain.
const	A numeric vector, specifies the value of all global variables, if param = T, then const = c(N,n,d,rh,r,K,alpha); if param=FALSE, then const = c(N,n,d,rh,r,K,sigma_sqare).
initial	A numeric vector, specifies the initial value of parameters over the continuous domain, if param = T, initial = c(pi_1,mean_1,cov_1, ..., pi_K, mean_K,cov_K), if param = F, initial = c(q1,q2, ..., qK).

Value

The Hellinger score after inputting sample y.

`HellingerScoreOne` *HellingerScoreOne*

Description

calculates the Hellinger score after inputting one sample y

Usage

```
HellingerScoreOne(y, param = TRUE, smart, const, initial)
```

Arguments

y	A one-row matrix, the new sample to be input.
param	A logical scalar, if TRUE, the model is in parametric version, otherwise, a non-parametric one.
smart	A matrix, stores all the parameters over the continuous domain.
const	A numeric vector, specifies the value of all global variables, if param = T, then const = c(N,n,d,rh,r,K,alpha); if param=FALSE, then const = c(N,n,d,rh,r,K,sigma_sqare).
initial	A numeric vector, specifies the initial value of parameters over the continuous domain, if param = T, initial = c(pi_1,mean_1,cov_1, ..., pi_K, mean_K,cov_K), if param = F, initial = c(q1,q2, ..., qK).

Value

The Hellinger score after inputting one sample y.

InitializeCell *InitializeCell*

Description

initializes parameters in the continuous domain while inputting the first sample

Usage

```
InitializeCell(y, param, initial, const)
```

Arguments

y	A one-row matrix, the new sample to be input.
param	A logical scalar, if TRUE, the model is in parametric version, otherwise, a non-parametric one.
initial	A numeric vector, specifies the initial value of parameters over the continuous domain, if param = T, initial = c(pi_1,mean_1,cov_1, ..., pi_K, mean_K,cov_K), if param = F, initial = c(q1,q2, ..., qK).
const	A numeric vector, specifies the value of all global variables, if param = T, then const = c(N,n,d,rh,r,K,alpha); if param=FALSE, then const = c(N,n,d,rh,r,K,sigma_sqare).

Value

The matrix which stores updated parameters over the continuous domain.

InputOneSample *InputOneSample*

Description

updates parameters after inputting one sample

Usage

```
InputOneSample(y, param = TRUE, smart, const, initial)
```

Arguments

y	A one-row matrix, the new sample to be input.
param	A logical scalar. If TRUE, the model is in parametric version, otherwise, a non-parametric one.
smart	A matrix, stores all the parameters over the continuous domain.
const	A numeric vector, specifies the value of all global variables, if param = T, then const = c(N,n,d,rh,r,K,alpha); if param=FALSE, then const = c(N,n,d,rh,r,K,sigma_sqare).
initial	A numeric vector, specifies the initial value of parameters over the continuous domain, if param = T, initial = c(pi_1,mean_1,cov_1, ..., pi_K, mean_K,cov_K), if param = F, initial = c(q1,q2, ..., qK).

Value

The matrix which stores updated parameters over the continuous domain.

InputSample *InputSample*

Description

updates parameters after inputting sample (can be more than one)

Usage

```
InputSample(y, param = TRUE, smart, const, initial)
```

Arguments

y	A matrix, the new sample to be input.
param	A logical scalar, if TRUE, the model is in parametric version, otherwise, a non-parametric one.
smart	A matrix, stores all the parameters over the continuous domain.
const	A numeric vector, specifies the value of all global variables, if param = T, then const = c(N,n,d,rh,r,K,alpha); if param=FALSE, then const = c(N,n,d,rh,r,K,sigma_sqare).
initial	A numeric vector, specifies the initial value of parameters over the continuous domain, if param = T, initial = c(pi_1,mean_1,cov_1, ..., pi_K, mean_K,cov_K), if param = F, initial = c(q1,q2, ..., qK).

Value

The matrix which stores updated parameters over the continuous domain.

LogLoss *LogLoss*

Description

calculates the logarithmic loss after inputting sample

Usage

```
LogLoss(y, param = TRUE, smart, const, initial)
```

Arguments

y	A matrix, the new sample to be input.
param	A logical scalar, if TRUE, the model is in parametric version, otherwise, a non-parametric one.
smart	A matrix, stores all the parameters over the continuous domain.
const	A numeric vector, specifies the value of all global variables, if param = T, then const = c(N,n,d,rh,r,K,alpha); if param=FALSE, then const = c(N,n,d,rh,r,K,sigma_sqare).
initial	A numeric vector, specifies the initial value of parameters over the continuous domain, if param = T, initial = c(pi_1,mean_1,cov_1, ..., pi_K, mean_K,cov_K), if param = F, initial = c(q1,q2, ..., qK).

Value

The logarithmic loss after inputting sample y.

LogLossOne *LogLossOne*

Description

calculates the logarithmic loss after inputting one sample

Usage

```
LogLossOne(y, param, smart, const, initial)
```

Arguments

y	A one-row matrix, the new sample to be input.
param	A logical scalar, if TRUE, the model is in parametric version, otherwise, a non-parametric one.
smart	A matrix, stores all the parameters over the continuous domain.
const	A numeric vector, specifies the value of all global variables, if param = T, then const = c(N,n,d,rh,r,K,alpha); if param=FALSE, then const = c(N,n,d,rh,r,K,sigma_sqare).
initial	A numeric vector, specifies the initial value of parameters over the continuous domain, if param = T, initial = c(pi_1,mean_1,cov_1, ..., pi_K, mean_K,cov_K), if param = F, initial = c(q1,q2, ..., qK).

Value

The logarithmic loss after inputting one sample y.

Test *Test*

Description

input new samples and compute their related scores (to detect possible outliers)

Usage

```
Test(y, param = TRUE, smart, const, initial)
```

Arguments

y	A matrix, the training set.
param	A logical scalar, if TRUE, the model is in parametric version, otherwise, a non-parametric one.
smart	A matrix, stores all the parameters over the continuous domain.
const	A numeric vector, specifies the value of all global variables, if param = T, then const = c(N,n,d,rh,r,K,alpha); if param=FALSE, then const = c(N,n,d,rh,r,K,sigma_sqare).
initial	A numeric vector, specifies the initial value of parameters over the continuous domain, if param = T, initial = c(pi_1,mean_1,cov_1, ..., pi_K, mean_K,cov_K), if param = F, initial = c(q1,q2, ..., qK).

Value

List of updated parameters.

Examples

```
## The parametric version
initial=matrix(c(0.5,0,0,1,0,0,1,0.5,1,1,1,0,0,1),nrow=1)
const = c(0,1,2,0.1,0.1,2,2)
param=TRUE
y=matrix(c(1,3,1,0,1,1),nrow=2)
smart = Train(y,param,const,initial)$smart
const[1] = Train(y,param,const,initial)$N
y=matrix(c(2,1,0),nrow=1)
smart = Test(y,param,smart,const,initial)$smart
HellingerScore = Test(y,param,smart,const,initial)$HellingerScore
LogLoss = Test(y,param,smart,const,initial)$LogLoss
const[1] = Test(y,param,smart,const,initial)$N
##The nonparametric version
param=FALSE
const = c(0,1,2,0.1,0.1,2,1)
initial = matrix(c(0,0,1,1),nrow=1)
y=matrix(c(1,3,1,0,1,1),nrow=2)
smart = Train(y,param,const,initial)$smart
const[1] = Train(y,param,const,initial)$N
y=matrix(c(2,1,0),nrow=1)
smart = Test(y,param,smart,const,initial)$smart
HellingerScore = Test(y,param,smart,const,initial)$HellingerScore
LogLoss = Test(y,param,smart,const,initial)$LogLoss
const[1] = Test(y,param,smart,const,initial)$N
```

`Train` *Train*

Description

trains the parameters

Usage

```
Train(y, param = TRUE, const, initial)
```

Arguments

`y`	A matrix, the training set.
`param`	A logical scalar, if TRUE, the model is in parametric version, otherwise, a non-parametric one.
`const`	A numeric vector, specifies the value of all global variables, if param = T, then const = c(N,n,d,rh,r,K,alpha); if param=FALSE, then const = c(N,n,d,rh,r,K,sigma_sqare).
`initial`	A numeric vector, specifies the initial value of parameters over the continuous domain, if param = T, initial = c(pi_1,mean_1,cov_1, ..., pi_K, mean_K,cov_K), if param = F, initial = c(q1,q2, ..., qK).

Value

List of all parameters.

Examples

```
##parametric model test
initial=matrix(c(0.5,0,0,1,0,0,1,0.5,1,1,1,0,0,1),nrow=1)
const = c(0,1,2,0.1,0.1,2,2)
param=TRUE
y=matrix(c(1,3,1,0,1,1),nrow=2)
smart = Train(y,param,const,initial)$smart
hellingerScore = Train(y,param,const,initial)$HellingerScore
logLoss = Train(y,param,const,initial)$LogLoss
const[1] = Train(y,param,const,initial)$N

##non-parametric model test
param=FALSE
const = c(0,1,2,0.1,0.1,2,1)
initial = matrix(c(0,0,1,1),nrow=1)
y=matrix(c(1,3,1,0,1,1),nrow=2)
smart = Train(y,param,const,initial)$smart
hellingerScore = Train(y,param,const,initial)$HellingerScore
logLoss = Train(y,param,const,initial)$LogLoss
const[1] = Train(y,param,const,initial)$N
```

UpdateConst *UpdateConst*

Description

updates the vector const after inputting sample y

Usage

```
UpdateConst(y, const)
```

Arguments

y	A matrix, the new sample to be input.
const	A numeric vector, specifies the value of all global variables, if param = T, then const = c(N,n,d,rh,r,K,alpha); if param=FALSE, then const = c(N,n,d,rh,r,K,sigma_sqare).

Value

The updated vector which specifies all the constant parameters.

`WhichCell` *WhichCell*

Description

returns the index of the cell to which the new sample belongs

Usage

```
WhichCell(y, param, smart, const)
```

Arguments

`y`	A one-row matrix, the new sample to be input.
`param`	A logical scalar, if TRUE, the model is in parametric version, otherwise, a non-parametric one.
`smart`	A matrix, stores all the parameters over the continuous domain.
`const`	A numeric vector, specifies the value of all global variables, if param = T, then const = c(N,n,d,rh,r,K,alpha); if param=FALSE, then const = c(N,n,d,rh,r,K,sigma_sqare).

Value

The row index of the discrete class to which the new sample belongs.

Index