Package ‘FuzzyClass’

March 7, 2023

Title Fuzzy and Non-Fuzzy Classifiers

Version 0.1.4

Description It provides classifiers which can be used for discrete variables and for continuous variables based on the Naive Bayes and Fuzzy Naive Bayes hypothesis. Those methods were developed by researchers belong to the 'Laboratory of Technologies for Virtual Teaching and Statistics (LabTEVE)' (<http://www.de.ufpb.br/~labteve/>) and 'Laboratory of Applied Statistics to Image Processing and Geoprocessing (LEAPIG)' (<http://www.de.ufpb.br/~leapig/>) at 'Federal University of Paraiba, Brazil'. They considered some statistical distributions and their papers were published in the scientific literature, as for instance, the Gaussian classifier using fuzzy parameters, proposed by 'Moraes, Ferreira and Machado' (2021) <doi:10.1007/s40815-020-00936-4>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports caTools, doParallel, dplyr, e1071, EnvStats, foreach, MASS, parallel, purrr, Rdpack, rootSolve, stats

RdMacros Rdpack

Depends R (>= 2.10)

Suggests spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/leapigufpb/FuzzyClass

BugReports https://github.com/leapigufpb/FuzzyClass/issues

Language en-US

NeedsCompilation no

Author Jodavid Ferreira [aut, cre] (<https://orcid.org/0000-0002-2131-6464>), Ronei Moraes [ctb] (<https://orcid.org/0000-0001-8436-8950>), Arthur Ricardo [ctb]

Maintainer Jodavid Ferreira <jodavid@protonmail.com>

Repository CRAN

Date/Publication 2023-03-07 17:20:02 UTC
ExpNBFuzzyParam

Description

ExpNBFuzzyParam Fuzzy Exponential Naive Bayes Classifier with Fuzzy parameters

Usage

ExpNBFuzzyParam(
  train,
  cl,
  alphacut = 1e-04,
  metd = 2,
  alp = c(0.35, 0.7, 0.86),
  w = c(0.1, 0.3, 0.6),
  cores = 2
)

Arguments

train matrix or data frame of training set cases
cl factor of true classifications of training set
alphacut value of the alpha-cut parameter, this value is between 0 and 1.
metd Method of transforming the triangle into scalar. It is the type of data entry for the test sample, use metd 1 if you want to use the Yager technique, metd 2 if you want to use the Q technique of the uniformity test (article: Directional Statistics and Shape analysis), and metd 3 if you want to use the Thorani technique
FuzzyBetaNaiveBayes

alp
- When metd for 4, it is necessary to have alp which are alpha-cut defined

w
- When metd for 4, it is necessary to have w which are alpha-cut weights defined

cores
- how many cores of the computer do you want to use (default = 2)

Value
- A vector of classifications

References

Examples

set.seed(1) # determining a seed
data(VirtualRealityData)

# Splitting into Training and Testing
split <- caTools::sample.split(t(VirtualRealityData[,1]), SplitRatio = 0.7)
Train <- subset(VirtualRealityData, split == "TRUE")
Test <- subset(VirtualRealityData, split == "FALSE")

# ----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -4]
fit_FENB <- ExpNBFuzzyParam(
  train = Train[, -4],
  cl = Train[, 4], metd = 1, cores = 2
)
pred_FENB <- predict(fit_FENB, test)
head(pred_FENB)
head(Test[, 4])
FuzzyBinomialNaiveBayes

Arguments

- **train**: matrix or data frame of training set cases.
- **cl**: factor of true classifications of training set
- **cores**: how many cores of the computer do you want to use (default = 2)
- **fuzzy**: boolean variable to use the membership function

Value

A vector of classifications

References


Examples

```r
set.seed(1) # determining a seed
data(iris)

# Splitting into Training and Testing
split <- caTools::sample.split(t(iris[, 1]), SplitRatio = 0.7)
Train <- subset(iris, split == "TRUE")
Test <- subset(iris, split == "FALSE")

# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -5]
fit_NBT <- FuzzyBetaNaiveBayes(
  train = Train[, -5],
  cl = Train[, 5], cores = 2
)

predict(fit_NBT, test)

head(test)
head(Train[, 5])
```

---

**FuzzyBinomialNaiveBayes**

*Fuzzy Binomial Naive Bayes*

Description

FuzzyBinomialNaiveBayes Fuzzy Binomial Naive Bayes
Usage

FuzzyBinomialNaiveBayes(train, cl, cores = 2, fuzzy = TRUE)

Arguments

- **train**: matrix or data frame of training set cases.
- **cl**: factor of true classifications of training set
- **cores**: how many cores of the computer do you want to use (default = 2)
- **fuzzy**: boolean variable to use the membership function

Value

A vector of classifications

References


Examples

```r
set.seed(1) # determining a seed
class1 <- data.frame(vari1 = rbinom(100, size = 10, prob = 0.2),
                     vari2 = rbinom(100, size = 10, prob = 0.2),
                     vari3 = rbinom(100, size = 10, prob = 0.2), class = 1)
class2 <- data.frame(vari1 = rbinom(100, size = 10, prob = 0.5),
                     vari2 = rbinom(100, size = 10, prob = 0.5),
                     vari3 = rbinom(100, size = 10, prob = 0.5), class = 2)
class3 <- data.frame(vari1 = rbinom(100, size = 10, prob = 0.8),
                     vari2 = rbinom(100, size = 10, prob = 0.8),
                     vari3 = rbinom(100, size = 10, prob = 0.8), class = 3)
data <- rbind(class1, class2, class3)

# Splitting into Training and Testing
split <- caTools::sample.split(t(data[, 1]), SplitRatio = 0.7)
Train <- subset(data, split == "TRUE")
Test <- subset(data, split == "FALSE")
# ----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -4]
fit_NBT <- FuzzyBinomialNaiveBayes(
  train = Train[, -4],
  cl = Train[, 4], cores = 2
)
pred_NBT <- predict(fit_NBT, test)

head(pred_NBT)
head(Test[, 4])
```
FuzzyExponentialNaiveBayes

Fuzzy Exponential Naive Bayes

Description

FuzzyExponentialNaiveBayes Fuzzy Exponential Naive Bayes

Usage

FuzzyExponentialNaiveBayes(train, cl, cores = 2, fuzzy = TRUE)

Arguments

- **train**: matrix or data frame of training set cases.
- **cl**: factor of true classifications of training set
- **cores**: how many cores of the computer do you want to use (default = 2)
- **fuzzy**: boolean variable to use the membership function

Value

A vector of classifications

References


Examples

```r
set.seed(1) # determining a seed
data(iris)

# Splitting into Training and Testing
split <- caTools::sample.split(t(iris[, 1]), SplitRatio = 0.7)
Train <- subset(iris, split == "TRUE")
Test <- subset(iris, split == "FALSE")
# -------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -5]
fit_NBT <- FuzzyExponentialNaiveBayes(
    train = Train[, -5],
    cl = Train[, 5], cores = 2
)
```
pred_NBT <- predict(fit_NBT, test)

head(pred_NBT)
head(Test[, 5])

### FuzzyGammaNaiveBayes

**Fuzzy Gamma Naive Bayes**

**Description**

FuzzyGammaNaiveBayes Fuzzy Gamma Naive Bayes

**Usage**

FuzzyGammaNaiveBayes(train, cl, cores = 2, fuzzy = TRUE)

**Arguments**

- **train**
  - matrix or data frame of training set cases.
- **cl**
  - factor of true classifications of training set
- **cores**
  - how many cores of the computer do you want to use (default = 2)
- **fuzzy**
  - boolean variable to use the membership function

**Value**

A vector of classifications

**References**


**Examples**

```
set.seed(1) # determining a seed
data(iris)

# Splitting into Training and Testing
split <- caTools::sample.split(t(iris[, 1]), SplitRatio = 0.7)
Train <- subset(iris, split == "TRUE")
Test <- subset(iris, split == "FALSE")
# ----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -5]
fit_NBT <- FuzzyGammaNaiveBayes(
```
train = Train[, -5],
c1 = Train[, 5], cores = 2
)
pred_NBT <- predict(fit_NBT, test)
head(pred_NBT)
head(Test[, 5])

FuzzyGaussianNaiveBayes

Fuzzy Gaussian Naive Bayes Classifier Zadeh-based

Description
FuzzyGaussianNaiveBayes Fuzzy Gaussian Naive Bayes Classifier Zadeh-based

Usage
FuzzyGaussianNaiveBayes(train, c1, cores = 2, fuzzy = TRUE)

Arguments
- train: matrix or data frame of training set cases.
- c1: factor of true classifications of training set
- cores: how many cores of the computer do you want to use (default = 2)
- fuzzy: boolean variable to use the membership function

Value
A vector of classifications

References

Examples

set.seed(1) # determining a seed
data(iris)

# Splitting into Training and Testing
split <- caTools::sample.split(t(iris[, 1]), SplitRatio = 0.7)
Train <- subset(iris, split == "TRUE")
Test <- subset(iris, split == "FALSE")
# ----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.

test <- Test[, -5]
fit_GNB <- FuzzyGaussianNaiveBayes(
  train = Train[, -5],
  cl = Train[, 5], cores = 2
)

pred_GNB <- predict(fit_GNB, test)

head(pred_GNB)
head(Test[, 5])

FuzzyNaiveBayes  Fuzzy Naive Bayes

Description

FuzzyNaiveBayes Fuzzy Naive Bayes

Usage

FuzzyNaiveBayes(train, cl, fuzzy = TRUE, m = NULL, Pi = NULL)

Arguments

train matrix or data frame of training set cases
cl factor of true classifications of training set
fuzzy boolean variable to use the membership function
m is M/N, where M is the number of classes and N is the number of train lines
Pi is 1/M, where M is the number of classes

Value

A vector of classifications

References

Examples

```r
set.seed(1) # determining a seed
data(HouseVotes84)

# Splitting into Training and Testing
split <- caTools::sample.split(t(HouseVotes84[, 1]), SplitRatio = 0.7)
Train <- subset(HouseVotes84, split == "TRUE")
Test <- subset(HouseVotes84, split == "FALSE")

# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -1]
fit_FNB <- FuzzyNaiveBayes(
  train = Train[, -1],
  cl = Train[, 1]
)

pred_FNB <- predict(fit_FNB, test)

head(pred_FNB)
head(Test[, 1])
```

---

**FuzzyPoissonNaiveBayes**

*Fuzzy Poisson Naive Bayes*

Description

FuzzyPoissonNaiveBayes Fuzzy Poisson Naive Bayes

Usage

FuzzyPoissonNaiveBayes(train, cl, cores = 2, fuzzy = TRUE)

Arguments

- `train`: matrix or data frame of training set cases.
- `cl`: factor of true classifications of training set
- `cores`: how many cores of the computer do you want to use (default = 2)
- `fuzzy`: boolean variable to use the membership function

Value

A vector of classifications
References


Examples

```r
set.seed(1) # determining a seed
class1 <- data.frame(vari1 = rpois(100, lambda = 2),
                     vari2 = rpois(100, lambda = 2),
                     vari3 = rpois(100, lambda = 2), class = 1)
class2 <- data.frame(vari1 = rpois(100, lambda = 1),
                     vari2 = rpois(100, lambda = 1),
                     vari3 = rpois(100, lambda = 1), class = 2)
class3 <- data.frame(vari1 = rpois(100, lambda = 5),
                     vari2 = rpois(100, lambda = 5),
                     vari3 = rpois(100, lambda = 5), class = 3)
data <- rbind(class1, class2, class3)

# Splitting into Training and Testing
split <- caTools::sample.split(t(data[, 1]), SplitRatio = 0.7)
Train <- subset(data, split == "TRUE")
Test <- subset(data, split == "FALSE")
# ----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -4]
fit_NBT <- FuzzyPoissonNaiveBayes(
  train = Train[, -4],
  cl = Train[, 4], cores = 2
)
pred_NBT <- predict(fit_NBT, test)
head(pred_NBT)
head(Test[, 4])
```

FuzzyTrapezoidalNaiveBayes

**Fuzzy Naive Bayes Trapezoidal Classifier**

**Description**

FuzzyTrapezoidalNaiveBayes Fuzzy Naive Bayes Trapezoidal Classifier

**Usage**

FuzzyTrapezoidalNaiveBayes(train, cl, cores = 2, fuzzy = T)
Arguments

- `train`: matrix or data frame of training set cases.
- `cl`: factor of true classifications of training set
- `cores`: how many cores of the computer do you want to use (default = 2)
- `fuzzy`: boolean variable to use the membership function

Value

A vector of classifications

References


Examples

```r
set.seed(1) # determining a seed
data(iris)

# Splitting into Training and Testing
split <- caTools::sample.split(t(iris[, 1]), SplitRatio = 0.7)
Train <- subset(iris, split == "TRUE")
Test <- subset(iris, split == "FALSE")

# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.

test <- Test[, -5]
fit_NBT <- FuzzyTrapezoidalNaiveBayes(
  train = Train[, -5],
  cl = Train[, 5], cores = 2
)

pred_NBT <- predict(fit_NBT, test)
head(pred_NBT)
head(Test[, 5])
```

---

**FuzzyTriangularNaiveBayes**

*Fuzzy Naive Bayes Triangular Classifier*

**Description**

FuzzyTriangularNaiveBayes Fuzzy Naive Bayes Triangular Classifier
Usage

FuzzyTriangularNaiveBayes(train, cl, cores = 2, fuzzy = TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>train</td>
<td>matrix or data frame of training set cases.</td>
</tr>
<tr>
<td>cl</td>
<td>factor of true classifications of training set</td>
</tr>
<tr>
<td>cores</td>
<td>how many cores of the computer do you want to use (default = 2)</td>
</tr>
<tr>
<td>fuzzy</td>
<td>boolean variable to use the membership function</td>
</tr>
</tbody>
</table>

Value

A vector of classifications

References


Examples

```r
set.seed(1) # determining a seed
data(iris)

# Splitting into Training and Testing
split <- caTools::sample.split(t(iris[, 1]), SplitRatio = 0.7)
Train <- subset(iris, split == "TRUE")
Test <- subset(iris, split == "FALSE")
# ----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -5]
fit_NBT <- FuzzyTriangularNaiveBayes(
  train = Train[, -5],
  cl = Train[, 5], cores = 2
)

pred_NBT <- predict(fit_NBT, test)
head(pred_NBT)
head(Test[, 5])
```
GauNB

Fuzzy Gaussian Naive Bayes Classifier with Fuzzy parameters

Description

GauNB Fuzzy Gaussian Naive Bayes Classifier with Fuzzy parameters

Usage

GauNB

(train,
cl,
alphacut = 1e-04,
methd = 2,
alp = c(0.35, 0.7, 0.86),
w = c(0.1, 0.3, 0.6),
cores = 2)

Arguments

- **train**: matrix or data frame of training set cases.
- **cl**: factor of true classifications of training set.
- **alphacut**: value of the alpha-cut parameter, this value is between 0 and 1.
- **methd**: Method of transforming the triangle into scalar. It is the type of data entry for the test sample, use methd 1 if you want to use the Yager technique, methd 2 if you want to use the Q technique of the uniformity test (article: Directional Statistics and Shape analysis), and methd 3 if you want to use the Thorani technique.
- **alp**: When methd for 4, it is necessary to have alp which are alpha-cut defined.
- **w**: When methd for 4, it is necessary to have w which are alpha-cut weights defined.
- **cores**: how many cores of the computer do you want to use (default = 2)

Value

A vector of classifications

References

Examples

```r
set.seed(1) # determining a seed
data(iris)

# Splitting into Training and Testing
split <- caTools::sample.split(t(iris[, 1]), SplitRatio = 0.7)
Train <- subset(iris, split == "TRUE")
Test <- subset(iris, split == "FALSE")

# A matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -5]
fit_FGNB <- GauNBFuzzyParam(
  train = Train[, -5],
  cl = Train[, 5], metd = 1, cores = 2
)

pred_FGNB <- predict(fit_FGNB, test)
head(pred_FGNB)
head(Test[, 5])
```

---

### Description

This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA. The CQA lists nine different types of votes: voted for, paired for, and announced for (these three simplified to yea), voted against, paired against, and announced against (these three simplified to nay), voted present, voted present to avoid conflict of interest, and did not vote or otherwise make a position known (these three simplified to an unknown disposition).

### Usage

```r
data(HouseVotes84)
```

### Format

A data frame with 435 observations on 17 variables:

1. Class Name: 2 (democrat, republican)
2. handicapped-infants: 2 (y,n)
3. water-project-cost-sharing: 2 (y,n)
4. adoption-of-the-budget-resolution: 2 (y,n)
5. physician-fee-freeze: 2 (y,n)
6. el-salvador-aid: 2 (y,n)
7. religious-groups-in-schools: 2 (y,n)
<table>
<thead>
<tr>
<th></th>
<th>Issue Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>anti-satellite-test-ban: 2 (y,n)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>aid-to-nicaraguan-contras: 2 (y,n)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>mx-missile: 2 (y,n)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>immigration: 2 (y,n)</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>synfuels-corporation-cutback: 2 (y,n)</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>education-spending: 2 (y,n)</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>superfund-right-to-sue: 2 (y,n)</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>crime: 2 (y,n)</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>duty-free-exports: 2 (y,n)</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>export-administration-act-south-africa: 2 (y,n)</td>
<td></td>
</tr>
</tbody>
</table>

**Source**

- Donor: Jeff Schlimmer (Jeffrey.Schlimmer@a.gp.cs.cmu.edu)

These data have been taken from the UCI Repository Of Machine Learning Databases at


and were converted to R format by Friedrich Leisch.

**References**


**Examples**

```r
data(HouseVotes84)
summary(HouseVotes84)
```

---

**PoiNBFuzzyParam**  
*Fuzzy Poisson Naive Bayes Classifier with Fuzzy parameters*

**Description**

PoiNBFuzzyParam Fuzzy Poisson Naive Bayes Classifier with Fuzzy parameters
Usage

PoiNBFuzzyParam(
    train,
    cl,
    alphacut = 1e-04,
    metd = 2,
    alp = c(0.35, 0.7, 0.86),
    w = c(0.1, 0.3, 0.6),
    cores = 2
)

Arguments

train matrix or data frame of training set cases.
cl factor of true classifications of training set
alphacut value of the alpha-cut parameter, this value is between 0 and 1.
metd Method of transforming the triangle into scalar. It is the type of data entry for
the test sample, use metd 1 if you want to use the Yager technique, metd 2 if you
want to use the Q technique of the uniformity test (article: Directional Statistics
and Shape analysis), and metd 3 if you want to use the Thorani technique
alp When metd for 4, it is necessary to have alp which are alpha-cut defined
w When metd for 4, it is necessary to have w which are alpha-cut weights defined
cores how many cores of the computer do you want to use (default = 2)

Value

A vector of classifications

References

parameters using data from different statistical distributions.” In IV Baillian Congress on Fuzzy
Sistems (CBSF 2016), volume 1, 57–68.

Examples

set.seed(1) # determining a seed
class1 <- data.frame(vari1 = rpois(100,lamba = 2),
    vari2 = rpois(100,lamba = 2),
    vari3 = rpois(100,lamba = 2), class = 1)
class2 <- data.frame(vari1 = rpois(100,lamba = 1),
    vari2 = rpois(100,lamba = 1),
    vari3 = rpois(100,lamba = 1), class = 2)
class3 <- data.frame(vari1 = rpois(100,lamba = 5),
    vari2 = rpois(100,lamba = 5),
    vari3 = rpois(100,lamba = 5), class = 3)
data <- rbind(class1,class2,class3)
# Splitting into Training and Testing
split <- caTools::sample.split(t(data[, 1]), SplitRatio = 0.7)
Train <- subset(data, split == "TRUE")
Test <- subset(data, split == "FALSE")

# -----------------
# matrix or data frame of test set cases.
# A vector will be interpreted as a row vector for a single case.
test <- Test[, -4]
fit_FPoiNB <- PoiNBfuzzyParam(
  train = Train[, -4],
  cl = Train[, 4], metd = 1, cores = 2
)
pred_FPoiNB <- predict(fit_FPoiNB, test)

head(pred_FPoiNB)
head(Test[, 4])

---

**SimulatedData**

**Simulated Data**

**Description**

A dataset containing training data from Gamma Distribution

**Usage**

SimulatedData

**Format**

A dataset with 600 rows and 4 variables with 1 label.

---

**VirtualRealityData**

**Virtual Reality Simulator Data**

**Description**

A dataset containing training data from a virtual reality simulator

**Usage**

VirtualRealityData

**Format**

A dataset with 600 rows and 4 variables with 1 label.
Index

* datasets
  HouseVotes84, 15
  SimulatedData, 18
  VirtualRealityData, 18

ExpNBfuzzyParam, 2

FuzzyBetaNaiveBayes, 3
FuzzyBinomialNaiveBayes, 4
FuzzyExponentialNaiveBayes, 6
FuzzyGammaNaiveBayes, 7
FuzzyGaussianNaiveBayes, 8
FuzzyNaiveBayes, 9
FuzzyPoissonNaiveBayes, 10
FuzzyTrapezoidalNaiveBayes, 11
FuzzyTriangularNaiveBayes, 12

GauNBFuzzyParam, 14

HouseVotes84, 15

PoiNBfuzzyParam, 16

SimulatedData, 18

VirtualRealityData, 18