Package ‘RPEnsemble’

October 7, 2017

Version 0.4
Date 2017-10-06
Title Random Projection Ensemble Classification
Author Timothy I. Cannings and Richard J. Samworth
Maintainer Timothy I. Cannings <cannings@marshall.usc.edu>
Description Implements the methodology of "Cannings, T. I. and Samworth, R. J. (2017) Random-projection ensemble classification, J. Roy. Stat. Soc., Ser. B. (with discussion), 79, 959--1035". The random projection ensemble classifier is a general method for classification of high-dimensional data, based on careful combination of the results of applying an arbitrary base classifier to random projections of the feature vectors into a lower-dimensional space. The random projections are divided into non-overlapping blocks, and within each block the projection yielding the smallest estimate of the test error is selected. The random projection ensemble classifier then aggregates the results of applying the base classifier on the selected projections, with a data-driven voting threshold to determine the final assignment.
Depends R (>= 3.4.2), MASS, parallel
Imports class, stats
License GPL-3
NeedsCompilation no
Repository CRAN
Date/Publication 2017-10-07 14:28:08 UTC

R topics documented:

<table>
<thead>
<tr>
<th>R package</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPEnsemble-package</td>
<td>2</td>
</tr>
<tr>
<td>Other.classifier</td>
<td>3</td>
</tr>
<tr>
<td>R</td>
<td>4</td>
</tr>
<tr>
<td>RPalpha</td>
<td>5</td>
</tr>
<tr>
<td>RPChoose</td>
<td>6</td>
</tr>
<tr>
<td>RPChooseSS</td>
<td>7</td>
</tr>
<tr>
<td>RPEnsembleClass</td>
<td>9</td>
</tr>
</tbody>
</table>
Description

Implements the methodology of Cannings and Samworth (2015). The random projection ensemble classifier is a very general method for classification of high-dimensional data, based on careful combination of the results of applying an arbitrary base classifier to random projections of the feature vectors into a lower-dimensional space. The random projections are divided into non-overlapping blocks, and within each block the projection yielding the smallest estimate of the test error is selected. The random projection ensemble classifier then aggregates the results of applying the base classifier on the selected projections, with a data-driven voting threshold to determine the final assignment.

Details

RPChoose chooses the projection from a block of size B2 that minimises an estimate of the test error (see Cannings and Samworth, 2015, Section 3), and classifies the training and test sets using the base classifier on the projected data. RParallel makes many calls to RPChoose in parallel. RPAlpha chooses the best empirical value of alpha (see Cannings and Samworth, 2015, Section 5.1). RPEnsembleClass combines the results of many base classifications to classify the test set.

The method can be used with any base classifier, any test error estimate and any distribution of the random projections. This package provides code for the following options: Classifiers – linear discriminant analysis, quadratic discriminant analysis and the K-nearest neighbour classifier. Error estimates – resubstitution and leave-one-out, we also provide code for the sample-splitting method described in Cannings and Samworth (2015, Section 7) (this can be done by setting estmethod = samplesplit). Projection distribution – Haar, Gaussian or axis-aligned projections.

The package provides the option to add your own base classifier and estimation method, this can be done by editing the code in the function Other.classifier. Moreover, one could edit the RPGenerate function to generate projections from different distributions.

Author(s)

Timothy I. Cannings and Richard J. Samworth

Maintainer: Timothy I. Cannings <t.cannings@statslab.cam.ac.uk>

References

Examples

```r
# generate data from Model 1
set.seed(101)
Train <- RPModel(2, 50, 100, 0.5)
Test <- RPModel(2, 100, 100, 0.5)

# Classify the training and test set for B1 = 10 independent projections, each
# one carefully chosen from a block of size B2 = 10, using the "knn" base
# classifier and the leave-one-out test error estimate
Out <- RPParallel(XTrain = Train$x, YTrain = Train$y, XTest = Test$x, d = 2,
B1 = 10, B2 = 10, base = "knn", projmethod = "Haar", estmethod = "loo",
splitsample = FALSE, k = seq(1, 25, by = 3), clustertype = "Default")

# estimate the class 1 prior probability
phat <- sum(Train$y == 1)/50

# choose the best empirical value of the voting threshold alpha
alphahat <- RPAlpha(RP.out = Out, Y = Train$y, p1 = phat)

# combine the base classifications
Class <- RPEnsembleClass(RP.out = Out, n = 50,
n.test = 100, p1 = phat, alpha = alphahat)

# calculate the error
mean(Class != Test$y)
```

# Code for sample splitting version of the above
```r
# n.val <- 25
# s <- sample(1:50,25)
# OutSS <- RPParallel(XTrain = Train$x[-s,], YTrain = Train$y[-s],
# XVal = Train$x[s,], YVal = Train$y[s], XTest = Test$x, d = 2,
# B1 = 50, B2 = 10, base = "knn", projmethod = "Haar", estmethod = "samplesplit",
# k = seq(1,13, by = 2), clustertype = "Fork", cores = 1)
# alphahatSS <- RPAlpha(RP.out = OutSS, Y = Train$y[s], p1 = phat)
# ClassSS <- RPEnsembleClass(RP.out = OutSS, n.val = 25, n.test = 100,
# p1 = phat, samplesplit = TRUE, alpha = alphahatSS)
# mean(ClassSS != Test$y)
```

Other.classifier

The users favourite classifier

Description

User defined code to convert existing R code for classification to the correct format

Usage

Other.classifier(x, grouping, xTest, CV, ...)

Arguments

- **x**: An \( n \) by \( p \) matrix containing the training dataset.
- **grouping**: A vector of length \( n \) containing the training data classes.
- **xTest**: An \( n_{\text{test}} \) by \( p \) test dataset.
- **CV**: If TRUE perform cross-validation (or otherwise) to classify training set. If FALSE, classify test set.
- **...**: Optional arguments e.g. tuning parameters.

Details

User editable code for your choice of base classifier.

Value

- **class**: a vector of classes of the training or test set.

Author(s)

Timothy I. Cannings and Richard J. Samworth

References


```R
A rotation matrix

Description

The 100 by 100 rotation matrix used in Model 2 in Cannings and Samworth (2016).

Usage

data(R)

Format

A 100 by 100 rotation matrix

References


Examples

data(R)
head(R%*%t(R))
**RPalpha**

*Choose alpha*

**Description**

Chooses the best empirical value of the cutoff alpha, based on the leave-one-out, resubstitution or sample-split estimates of the class labels.

**Usage**

\[ \text{RPalpha}(\text{RP.out, Y, p1}) \]

**Arguments**

- **RP.out**: The result of a call to RPParallel
- **Y**: Vector of length \( n \) or \( n_{\text{val}} \) containing the training or validation dataset classes
- **p1**: (Empirical) prior probability

**Details**

See precise details in Cannings and Samworth (2015, Section 5.1).

**Value**

- **alpha**: The value of alpha that minimises the empirical error

**Author(s)**

Timothy I. Cannings and Richard J. Samworth

**References**


**See Also**

RPParallel

**Examples**

```r
Train <- RModel(1, 50, 100, 0.5)
Test <- RModel(1, 100, 100, 0.5)
Out <- RPParallel(XTrain = Train$x, YTrain = Train$y, XTest = Test$x, d = 2, B1 = 10, B2 = 10, base = "LDA", projmethod = "Haar", estmethod = "training", cores = 1)
alpha <- RPalpha(RP.out = Out, Y = Train$y, p1 = sum(Train$y == 1)/length(Train$y))
alpha
```
RPChoose  

*Chooses projection*

**Description**

Chooses the best projection from a set of size B2 based on a test error estimate, then classifies the training and test sets using the chosen projection.

**Usage**

```
RPChoose(XTrain, YTrain, XTest, d, B2 = 10, base = "LDA", k = c(3, 5),
   projmethod = "Haar", estmethod = "training", ...)
```

**Arguments**

- `XTrain`: An n by p matrix containing the training data feature vectors.
- `YTrain`: A vector of length n of the classes (either 1 or 2) of the training data.
- `XTest`: An n.test by p matrix of the test data.
- `d`: The lower dimension of the image space of the projections.
- `B2`: The block size.
- `base`: The base classifier one of "knn", "LDA", "QDA" or "other".
- `k`: The options for k if base is "knn".
- `projmethod`: Either "Haar", "Gaussian" or "axis".
- `estmethod`: Method for estimating the test errors to choose the projection: either training error "training" or leave-one-out "loo".
- `...`: Optional further arguments if base = "other".

**Details**

Randomly projects the data B2 times. Chooses the projection yielding the smallest estimate of the test error. Classifies the training set (via the same method as estmethod) and test set using the chosen projection.

**Value**

Returns a vector of length n + n.test: the first n entries are the estimated classes of the training set, the last n.test are the estimated classes of the test set.

**Note**

Resubstitution method unsuitable for the k-nearest neighbour classifier.

**Author(s)**

Timothy I. Cannings and Richard J. Samworth
RPChooseSS

References

See Also
RPPParallel, RPChooseSS, lda, qda, knn

Examples
```r
set.seed(100)
Train <- RPMModel(1, 50, 100, 0.5)
Test <- RPMModel(1, 100, 100, 0.5)
Choose.out5 <- RPChoose(XTrain = Train$x, YTrain = Train$y, XTest = Test$x,
d = 2, B2 = 5, base = "QDA", projmethod = "Haar", estmethod = "loo")
Choose.out10 <- RPChoose(XTrain = Train$x, YTrain = Train$y, XTest = Test$x,
d = 2, B2 = 10, base = "QDA", projmethod = "Haar", estmethod = "loo")
sum(Choose.out5[1:50] != Train$y)
sum(Choose.out10[1:50] != Train$y)
sum(Choose.out5[51:150] != Test$y)
sum(Choose.out10[51:150] != Test$y)
```

RPChooseSS

A sample splitting version of RPChoose

Description
Chooses the best projection based on an estimate of the test error of the classifier with training data (XTrain, YTrain), the estimation method counts the number of errors made on the validation set (XVal, YVal).

Usage
```r
RPChooseSS(XTrain, YTrain, XVal, YVal, XTest, d, B2 = 100, base = "LDA",
k = c(3, 5), projmethod = "Haar", 
...)
```

Arguments
- **XTrain**: An n by p matrix containing the training data feature vectors
- **YTrain**: A vector of length n of the classes (either 1 or 2) of the training data
- **XVal**: An n.val by p matrix containing the validation data feature vectors
- **YVal**: A vector of length n.val of the classes (either 1 or 2) of the validation data
- **XTest**: An n.test by p matrix of the test data feature vectors
- **d**: The lower dimension of the image space of the projections
- **B2**: The block size
- **base**: The base classifier one of "knn", "LDA", "QDA" or "other"
k

The options for k if base = “knn”

projmethod

Either "Haar", "Gaussian" or "axis"

...

Optional further arguments if base = "other"

Details

Maps the the data using B2 random projections. For each projection the validation set is classified using the the training set and the projection yielding the smallest number of errors over the validation set is retained. The validation set and test set are then classified using the chosen projection.

Value

Returns a vector of length n.val + n.test: the first n.val entries are the estimated classes of the validation set, the last n.test are the estimated classes of the test set.

Author(s)

Timothy I. Cannings and Richard J. Samworth

References


See Also

RParallel, RChoose, lda, qda, knn

Examples

set.seed(100)
Train <- RPMModel(1, 50, 100, 0.5)
Validate <- RPMModel(1, 50, 100, 0.5)
Test <- RPMModel(1, 100, 100, 0.5)
Choose.out5 <- RChooseSS(XTrain = Train$x, YTrain = Train$y, XVal = Validate$x, YVal = Validate$y, XTest = Test$x, d = 2, B2 = 5, base = "QDA", projmethod = "Haar")
Choose.out10 <- RChooseSS(XTrain = Train$x, YTrain = Train$y, XVal = Validate$x, YVal = Validate$y, XTest = Test$x, d = 2, B2 = 10, base = "QDA", projmethod = "Haar")
sum(Choose.out5[[1:50]] != Validate$y)
sum(Choose.out10[[1:50]] != Validate$y)
sum(Choose.out5[[51:150]] != Test$y)
sum(Choose.out10[[51:150]] != Test$y)
RPEnsembleClass

Classifies the test set using the random projection ensemble classifier

Description

Performs a biased majority vote over \( B_1 \) base classifications to assign the test set.

Usage

\[
\text{RPEnsembleClass}(\text{RP.out}, n, n.val, n.test, p, \text{samplesplit}, \alpha, \ldots)
\]

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RP.out</td>
<td>The result of a call to RPParallel</td>
</tr>
<tr>
<td>n</td>
<td>Training set sample size</td>
</tr>
<tr>
<td>n.test</td>
<td>Test set sample size</td>
</tr>
<tr>
<td>n.val</td>
<td>Validation set sample size</td>
</tr>
<tr>
<td>p</td>
<td>Prior probability estimate</td>
</tr>
<tr>
<td>samplesplit</td>
<td>TRUE if using sample-splitting method</td>
</tr>
<tr>
<td>alpha</td>
<td>The voting threshold</td>
</tr>
<tr>
<td>...</td>
<td>Optional further arguments if base = &quot;other&quot;</td>
</tr>
</tbody>
</table>

Details

An observation in the test set is assigned to class 1 if \( B_1 \alpha \) or more of the base classifications are class 1 (otherwise class 2).

Value

A vector of length \( n.test \) containing the class predictions of the test set (either 1 or 2).

Author(s)

Timothy I. Cannings and Richard J. Samworth

References


See Also

RPParallel, RPalpha, RPChoose
Examples

Train <- RPModel(1, 50, 100, 0.5)
Test <- RPModel(1, 100, 100, 0.5)
Out <- RPParallel(XTrain = Train$x, YTrain = Train$y, XTest = Test$x, d = 2, B1 = 50, B2 = 10, base = "LDA", projmethod = "Haar", estmethod = "training", clustertype = "Default")
Class <- RPEnsembleClass(RP.out = Out, n = length(Train$y), n.test = nrow(Test$x), p1 = sum(Train$y == 1)/length(Train$y), splitsample = FALSE, alpha = RPAalpha(Out, Y = Train$y, p1 = sum(Train$y == 1)/length(Train$y)))
mean(Class != Test$y)

RPGenerate  Generates random matrices

Description

Generates B2 random p by d matrices according to Haar measure, Gaussian or axis-aligned projections

Usage

RPGenerate(p = 100, d = 10, method = "Haar", B2 = 10)

Arguments

p  The original data dimension

 d  The lower dimension

method  Projection distribution, either "Haar" for Haar distributed projections, "Gaussian" for Gaussian distributed projections with i.i.d. \(N(0,1/p)\) entries, "axis" for uniformly distributed axis aligned projections, or "other" for user defined method

B2  the number of projections

Value

returns B2 p by d random matrices as a single p by d*B2 matrix

Author(s)

Timothy I. Cannings and Richard J. Samworth

References

RPModel

Examples

```r
R1 <- RPGenerate(p = 20, d = 2, "Haar", B2 = 3)
R2 <- RPGenerate(p = 20, d = 2, "Gaussian", B2 = 3)
R3 <- RPGenerate(p = 20, d = 2, "axis", B2 = 3)
colSums(R3)
rowSums(R3)
```

RPModel  

*Generate pairs \((x,y)\) from joint distribution*

Description

Generates data from the models described in Cannings and Samworth (2016)

Usage

```r
RPModel(Model.No, n, p, Pi = 1/2)
```

Arguments

- `Model.No`: Model Number
- `n`: Sample size
- `p`: Data dimension
- `Pi`: Class one prior probability

Value

- `x`: An \(n \times p\) data matrix – \(n\) observations of the \(p\)-dimensional features
- `y`: A vector of length \(n\) containing the classes (either 1 or 2)

Note

Models 1 and 2 require \(p = 100\) or \(1000\).

Author(s)

Timothy I. Cannings and Richard J. Samworth

References

Examples

```r
Data <- RPModel(Model.No = 1, 100, 100, P1 = 1/2)
table(Data$y)
colMeans(Data$x[Data$y==1,])
colMeans(Data$x[Data$y==2,])
```

---

**RPPParallel**

*Chooses a projection from each block in parallel*

**Description**

Makes \( B_1 \) calls to `RPChoose` or `RPChooseS` in parallel and returns the results as a matrix.

**Usage**

```r
RPPParallel(XTrain, YTrain, XVal, YVal, XTest, d, B1 = 500, B2 = 50,
base = "LDA", projmethod = "Gaussian", estmethod = "training", k = c(3,5,9),
clustertype = "Default", cores = 1, machines = NULL, seed = 1, ...)"
```

**Arguments**

- `XTrain` An \( n \) by \( p \) matrix containing the training data feature vectors
- `YTrain` A vector of length \( n \) containing the classes (either 1 or 2) of the training data
- `XVal` An \( n \).val by \( p \) matrix containing the validation data feature vectors
- `YVal` A vector of length \( n \).val of the classes (either 1 or 2) of the validation data
- `XTest` An \( n \).test by \( p \) matrix containing the test data feature vectors
- `d` The lower dimension of the image space of the projections
- `B1` The number of blocks
- `B2` The size of each block
- `base` The base classifier one of "knn", "LDA", "QDA" or "other"
- `k` The options for \( k \) if base is "knn"
- `projmethod` "Haar", "Gaussian" or "axis"
- `estmethod` Method for estimating the test errors to choose the projection: either training error "training", leave-one-out "loo", or sample split "samplesplit"
- `clustertype` The type of cluster: "Default" uses just one core, "Fork" uses a single machine, "Socket" uses many machines. Note "Fork" and "Socket" are not supported on windows.
- `cores` Required only if clustertype==Fork: the number of computer cores to use (note: cores > 1 not supported on Windows)
- `machines` Required only if clustertype==Socket: the names of the machines to use e.g. c("Computer1", "Computer2") (not supported on Windows)
- `seed` If not NULL, sets random seed for reproducible results
- `...` Optional further arguments if base = "other"
Details

Makes B1 calls to \texttt{RPChoose} or \texttt{RPChooseSS} in parallel.

Value

If \texttt{estmethod == \"training\" or \"loo\"}, then returns an n+n.test by B1 matrix, each row containing the result of a call to \texttt{RPChoose}. If \texttt{estmethod == \"samplesplit\"}, then returns an n.val+n.test by B1 matrix, each row containing the result of a call to \texttt{RPChooseSS}.

Author(s)

Timothy I. Cannings and Richard J. Samworth

References


See Also

\texttt{RPChoose}, \texttt{RPChooseSS}

Examples

Train <- RPModel(1, 50, 100, 0.5)
Test <- RPModel(1, 100, 100, 0.5)
Out <- RPParallel(XTrain = Train$x, YTrain = Train$y, XTest = Test$x, d = 2, B1 = 10, 
B2 = 10, base = \"LDA\", projmethod = \"Haar\", estmethod = \"training\")
colMeans(Out)
Index

Other.classifier, 2, 3

R, 4
RPalphA, 2, 5, 9
RChoose, 2, 6, 7–9, 12, 13
RChooseSs, 7, 7, 12, 13
RP(Ensemble (RP(Ensemble-package)), 2
RP(Ensemble-package), 2
RP(EnsembleClass, 2, 9
RPGenerate, 2, 10
RPModel, 11
RPParallel, 2, 5, 7–9, 12