Package: customizedTraining (via r-universe)

November 5, 2024

Type Package

Title Customized Training for Lasso and Elastic-Net Regularized Generalized Linear Models

Version 1.2

Date 2019-01-28

Author Scott Powers, Trevor Hastie, Robert Tibshirani

Maintainer Scott Powers <saberpowers@gmail.com>

Imports FNN, glmnet

Description Customized training is a simple technique for transductive learning, when the test covariates are known at the time of training. The method identifies a subset of the training set to serve as the training set for each of a few identified subsets in the training set. This package implements customized training for the glmnet() and cv.glmnet() functions.

License GPL-2

Repository https://saberpowers.r-universe.dev

RemoteUrl https://github.com/saberpowers/customizedtraining

RemoteRef HEAD

RemoteSha 8feae96962cb66b2d396c26327ba59ba2a9e836a

Contents

`customizedTraining-package`
customizedTraining

Description

Fit a regularized lasso model using customized training

Details

Customized training is a simple strategy for making predictions on test data when the features of the test data are available at the time of model fitting. The method clusters the data to find training points close to each test point and then fits a GLM elastic net model separately in each training cluster. In this way, customized training is a localized method for transductive learning. In contrast with local regression, however, instead of fitting a separate regression model for each test point, customized training fits only one model for each of a handful of clusters.

Use `customizedGlmnet()` to fit a `glmnet()` model with customized training. Use `cv.customizedGlmnet()` to do the same while choosing the regularization parameter and potentially the number of groups using cross-validation. The `plot()` and `predict()` methods are implemented for both `customizedGlmnet()` and `cv.customizedGlmnet()`.

Author(s)

Scott Powers, Trevor Hastie, Robert Tibshirani

Maintainer: Scott Powers <sspowers@stanford.edu>

References

Scott Powers, Trevor Hastie and Robert Tibshirani (2015) "Customized training with an application to mass specrometric imaging of gastric cancer data." Annals of Applied Statistics 9, 4:1709-1725.

See Also

`glmnet`

Examples

```
require(glmnet)


#   Simulate three groups, each with a different sparse linear model
#   producing the response, and fit customized training model (with CV) to the
#   synthetic data

# Simulation parameters:
n = m = 300
p = 100
q = 10
K = 3
sigmaC = 10
sigmaX = sigmaY = 1
set.seed(5914)

# Produce the synthetic data:
beta = matrix(0, nrow = p, ncol = K)
for (k in 1:K) beta[sample(1:p, q), k] = 1
c = matrix(rnorm(K*p, 0, sigmaC), K, p)
eta = rnorm(K)
pi = (exp(eta)+1)/sum(exp(eta)+1)
z = t(rmultinom(m + n, 1, pi))
x = crossprod(t(z), c) + matrix(rnorm((m + n)*p, 0, sigmaX), m + n, p)
y = rowSums(z*(crossprod(t(x), beta))) + rnorm(m + n, 0, sigmaY)

x.train = x[1:n, ]
y.train = y[1:n]
x.test = x[n + 1:m, ]
foldid = sample(rep(1:10, length = nrow(x.train)))

# Fit the customized training model with CV:
fit2 = cv.customizedGlmnet(x.train, y.train, x.test, Gs = 1:3,
    family = "gaussian", foldid = foldid)

# Print the optimal number of groups and value of lambda:
fit2$G.min
fit2$lambda.min

# Print the customized training model fit:
fit2

# Compute test error using the predict function:
mean((y[n + 1:m] - predict(fit2))^2)

# Plot nonzero coefficients by group:
plot(fit2)
```

---

customizedGlmnet *fit glmnet using customized training*

Description

Fit a regularized lasso model using customized training

Usage

```
customizedGlmnet(xTrain, yTrain, xTest, groupid = NULL, G = NULL,
    family = c("gaussian", "binomial", "multinomial"), dendrogram = NULL,
    dendrogramTestIndices = NULL)
```

Arguments

`xTrain`	an n-by-p matrix of training covariates
`yTrain`	a length-n vector of training responses. Numeric for family = "gaussian". Factor or character for family = "binomial" or family = "multinomial"
`xTest`	an m-by-p matrix of test covariates
`groupid`	an optional length-m vector of group memberships for the test set. If specified, customized training subsets are identified using the union of nearest neighbor sets for each test group. Either `groupid` or `G` must be specified
`G`	a positive integer indicating the number of clusters for the joint clustering of the test and training data. Ignored if `groupid` is specified. Either `groupid` or `G` must be specified
`family`	response type
`dendrogram`	optional output from `hclust` on the joint covariate data. Used by `cv.customizedGlmnet` so that clustering is not computed redundantly
`dendrogramTestIndices`	optional set of indices (corresponding to dendrogram) held out in cross-validation. Used by `cv.customizedGlmnet`

Details

Identify customized training subsets of the training data through one of two methods: (1) If groupid is specified, grouping the test data, then for each test group find the 10 nearest neighbors of each observation in the group and use the union of these nearest neighbor sets as the customized training set or (2) If G is specified, jointly cluster the test and training data using hierarchical clustering with complete linkage. Within each cluster, the training data are used as the customized training subset for the test data. Once the customized training subsets have been identified, use glmnet to fit an l1-regularized regression model to each.

Value

an object with class `customizedGlmnet`

`call`	the call that produced this object
`CTset`	a list containing the customized training subsets for each test group
`fit`	a list containing the glmnet fit for each test group
`groupid`	a length-m vector containing the group memberships of the test data

x	a list containing `train` (which is the input `xTrain`) and `test` (which is the input `xTest`). Specified in function call
y	training response vector (specified in function call)
family	response type (specified in function call)
standard	the fit of `glmnet` to the entire training set using standard training

Author(s)

Scott Powers, Trevor Hastie, Robert Tibshirani

References

Scott Powers, Trevor Hastie and Robert Tibshirani (2015) "Customized training with an application to mass specrometric imaging of gastric cancer data." Annals of Applied Statistics 9, 4:1709-1725.

See Also

`print.customizedGlmnet`, `predict.customizedGlmnet`, `plot.customizedGlmnet`, `cv.customizedGlmnet`

Examples

```
require(glmnet)

# Simulate synthetic data

n = m = 150
p = 50
q = 5
K = 3
sigmaC = 10
sigmaX = sigmaY = 1
set.seed(5914)

beta = matrix(0, nrow = p, ncol = K)
for (k in 1:K) beta[sample(1:p, q), k] = 1
c = matrix(rnorm(K*p, 0, sigmaC), K, p)
eta = rnorm(K)
pi = (exp(eta)+1)/sum(exp(eta)+1)
z = t(rmultinom(m + n, 1, pi))
x = crossprod(t(z), c) + matrix(rnorm((m + n)*p, 0, sigmaX), m + n, p)
y = rowSums(z*(crossprod(t(x), beta))) + rnorm(m + n, 0, sigmaY)

x.train = x[1:n, ]
y.train = y[1:n]
x.test = x[n + 1:m, ]
y.test = y[n + 1:m]


# Example 1: Use clustering to fit the customized training model to training
# and test data with no predefined test-set blocks
```

```
fit1 = customizedGlmnet(x.train, y.train, x.test, G = 3,
    family = "gaussian")

# Print the customized training model fit:
fit1

# Extract nonzero regression coefficients for each group:
nonzero(fit1, lambda = 10)

# Compute test error using the predict function:
mean((y.test - predict(fit1, lambda = 10))^2)

# Plot nonzero coefficients by group:
plot(fit1, lambda = 10)


# Example 2: If the test set has predefined blocks, use these blocks to define
# the customized training sets, instead of using clustering.
group.id = apply(z == 1, 1, which)[n + 1:m]

fit2 = customizedGlmnet(x.train, y.train, x.test, group.id)

# Print the customized training model fit:
fit2

# Extract nonzero regression coefficients for each group:
nonzero(fit2, lambda = 10)

# Compute test error using the predict function:
mean((y.test - predict(fit2, lambda = 10))^2)

# Plot nonzero coefficients by group:
plot(fit2, lambda = 10)
```

`cv.customizedGlmnet` *cross validation for customizedGlmnet*

Description

Does k-fold cross-validation for customizedGlmnet and returns a values for `G` and `lambda`

Usage

```
cv.customizedGlmnet(xTrain, yTrain, xTest = NULL, groupid = NULL, Gs = NULL,
    dendrogram = NULL, dendrogramCV = NULL, lambda = NULL,
    nfolds = 10, foldid = NULL, keep = FALSE,
    family = c("gaussian", "binomial", "multinomial"), verbose = FALSE)
```

Arguments

xTrain	an n-by-p matrix of training covariates
yTrain	a length-n vector of training responses. Numeric for family = "gaussian". Factor or character for family = "binomial" or family = "multinomial"
xTest	an m-by-p matrix of test covariates. May be left NULL, in which case cross validation predictions are made internally on the training set and no test predictions are returned.
groupid	an optional length-m vector of group memberships for the test set. If specified, customized training subsets are identified using the union of nearest neighbor sets for each test group, in which case cross-validation is used only to select the regularization parameter lambda, not the number of clusters G. Either groupid or Gs must be specified
Gs	a vector of positive integers indicating the numbers of clusters over which to perform cross-validation to determine the best number. Ignored if groupid is specified. Either groupid or Gs must be specified
dendrogram	optional output from hclust on the joint covariate data. Useful if method is being used several times to avoid redundancy in calculations
dendrogramCV	optional output from hclust on the training covariate data. Used as joint clustering result for cross-validation. Useful to specify in advance if method is being used several times to avoid redundancy in calculations
lambda	sequence of values to use for the regularization parameter lambda. Recomended to leave as NULL and allow glmnet to choose automatically.
nfolds	number of folds – default is 10. Ignored if foldid is specified
foldid	an optional length-n vector of fold memberships used for cross-validation
keep	Should fitted values on the training set from cross validation be included in output? Default is FALSE.
family	response type
verbose	Should progress be printed to console as folds are evaluated during cross-validation? Default is FALSE.

Value

an object of class cv.customizedGlmnet

call	the call that produced this object
G.min	unless groupid is specified, the number of clusters minimizing CV error
lambda	the sequence of values of the regularization parameter lambda considered
lambda.min	the value of the regularization parameter lambda minimizing CV error
error	a matrix containing the CV error for each G and lambda
fit	a customizedGlmnet object fit using G.min and lambda.min. Only returned if xTest is not NULL.
prediction	a length-m vector of predictions for the test set, using the tuning parameters which minimize cross-validation error. Only returned if xTest is not NULL.

`selected` a list of nonzero variables for each customized training set, using `G.min` and `lambda.min`. Only returned if `xTest` is not NULL.

`cv.fit` a array containing fitted values on the training set from cross validation. Only returned if keep is TRUE.

Author(s)

Scott Powers, Trevor Hastie, Robert Tibshirani

References

Scott Powers, Trevor Hastie and Robert Tibshirani (2015) "Customized training with an application to mass specrometric imaging of gastric cancer data." Annals of Applied Statistics 9, 4:1709-1725.

See Also

`customizedGlmnet`, `plot.cv.customizedGlmnet`, `predict.cv.customizedGlmnet`

Examples

```
require(glmnet)

# Simulate synthetic data

n = m = 150
p = 50
q = 5
K = 3
sigmaC = 10
sigmaX = sigmaY = 1
set.seed(5914)

beta = matrix(0, nrow = p, ncol = K)
for (k in 1:K) beta[sample(1:p, q), k] = 1
c = matrix(rnorm(K*p, 0, sigmaC), K, p)
eta = rnorm(K)
pi = (exp(eta)+1)/sum(exp(eta)+1)
z = t(rmultinom(m + n, 1, pi))
x = crossprod(t(z), c) + matrix(rnorm((m + n)*p, 0, sigmaX), m + n, p)
y = rowSums(z*(crossprod(t(x), beta))) + rnorm(m + n, 0, sigmaY)

x.train = x[1:n, ]
y.train = y[1:n]
x.test = x[n + 1:m, ]
y.test = y[n + 1:m]
foldid = sample(rep(1:10, length = nrow(x.train)))


# Example 1: Use clustering to fit the customized training model to training
# and test data with no predefined test-set blocks

fit1 = cv.customizedGlmnet(x.train, y.train, x.test, Gs = c(1, 2, 3, 5),
```

```
    family = "gaussian", foldid = foldid)

# Print the optimal number of groups and value of lambda:
fit1$G.min
fit1$lambda.min

# Print the customized training model fit:
fit1

# Compute test error using the predict function:
mean((y[n + 1:m] - predict(fit1))^2)

# Plot nonzero coefficients by group:
plot(fit1)


# Example 2: If the test set has predefined blocks, use these blocks to define
# the customized training sets, instead of using clustering.
foldid = apply(z == 1, 1, which)[1:n]
group.id = apply(z == 1, 1, which)[n + 1:m]

fit2 = cv.customizedGlmnet(x.train, y.train, x.test, group.id, foldid = foldid)

# Print the optimal value of lambda:
fit2$lambda.min

# Print the customized training model fit:
fit2

# Compute test error using the predict function:
mean((y[n + 1:m] - predict(fit2))^2)

# Plot nonzero coefficients by group:
plot(fit2)


# Example 3: If there is no test set, but the training set is organized into
# blocks, you can do cross validation with these blocks as the basis for the
# customized training sets.

fit3 = cv.customizedGlmnet(x.train, y.train, foldid = foldid)

# Print the optimal value of lambda:
fit3$lambda.min

# Print the customized training model fit:
fit3

# Compute test error using the predict function:
mean((y[n + 1:m] - predict(fit3))^2)

# Plot nonzero coefficients by group:
plot(fit3)
```

`nonzero` *return selected variables*

Description

`nonzero` is a generic function for returning the set of variables selected by a model

Usage

```
nonzero(object, ...)
```

Arguments

`object`	a model object for which the set of selected variables is desired
`...`	additional arguments, e.g. tuning parameters

Value

The form of the value returned by `nonzero` depends on the class of its argument. See the documentation of the particular methods for details of what is produced by that method.

See Also

`nonzero.customizedGlmnet`, `nonzero.singleton`

`nonzero.customizedGlmnet` *return selected variables from a* `customizedGlmnet` *object*

Description

Returns a list of vectors of selected variables for each separate glmnet model fit by a `customizedGlmnet` object

Usage

```
## S3 method for class 'customizedGlmnet'
nonzero(object, lambda = NULL, ...)
```

Arguments

`object`	fitted `customizedGlmnet` model object
`lambda`	value of regularization parameter to use. Must be specified
`...`	ignored

Value

a list of vectors, each vectors representing one of the `glmnet` models fit by the `customizedGlmnet` model. Each vector gives the indices of the variables selected by the model

Author(s)

Scott Powers, Trevor Hastie, Robert Tibshirani

References

Scott Powers, Trevor Hastie and Robert Tibshirani (2015) "Customized training with an application to mass specrometric imaging of gastric cancer data." Annals of Applied Statistics 9, 4:1709-1725.

See Also

`nonzero`, `customizedGlmnet`

Examples

```
require(glmnet)

# Simulate synthetic data

n = m = 150
p = 50
q = 5
K = 3
sigmaC = 10
sigmaX = sigmaY = 1
set.seed(5914)

beta = matrix(0, nrow = p, ncol = K)
for (k in 1:K) beta[sample(1:p, q), k] = 1
c = matrix(rnorm(K*p, 0, sigmaC), K, p)
eta = rnorm(K)
pi = (exp(eta)+1)/sum(exp(eta)+1)
z = t(rmultinom(m + n, 1, pi))
x = crossprod(t(z), c) + matrix(rnorm((m + n)*p, 0, sigmaX), m + n, p)
y = rowSums(z*(crossprod(t(x), beta))) + rnorm(m + n, 0, sigmaY)

x.train = x[1:n, ]
y.train = y[1:n]
x.test = x[n + 1:m, ]
y.test = y[n + 1:m]


# Example 1: Use clustering to fit the customized training model to training
# and test data with no predefined test-set blocks

fit1 = customizedGlmnet(x.train, y.train, x.test, G = 3,
    family = "gaussian")
```

```
# Extract nonzero regression coefficients for each group:
nonzero(fit1, lambda = 10)


# Example 2: If the test set has predefined blocks, use these blocks to define
# the customized training sets, instead of using clustering.
group.id = apply(z == 1, 1, which)[n + 1:m]

fit2 = customizedGlmnet(x.train, y.train, x.test, group.id)

# Extract nonzero regression coefficients for each group:
nonzero(fit2, lambda = 10)
```

`nonzero.singleton` *return selected variables from a* `singleton` *object*

Description

Returns NULL. Intended for internal use only

Usage

```
## S3 method for class 'singleton'
nonzero(object, ...)
```

Arguments

`object`	an object of class `singleton`
`...`	ignored

Value

NULL

See Also

`nonzero`

plot.customizedGlmnet *visualize variables selected in each customized training subset*

Description

Produces a plot, with a row for each customized training submodel, showing the variables selected in the subset, with variables along the horizonal axis

Usage

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

Arguments

`x`	a fitted `customizedGlmnet` object
`lambda`	regularization parameter. Required
`...`	ignored

Author(s)

Scott Powers, Trevor Hastie, Robert Tibshirani

See Also

`plot`, `customizedGlmnet`

Examples

```
require(glmnet)

# Simulate synthetic data

n = m = 150
p = 50
q = 5
K = 3
sigmaC = 10
sigmaX = sigmaY = 1
set.seed(5914)

beta = matrix(0, nrow = p, ncol = K)
for (k in 1:K) beta[sample(1:p, q), k] = 1
c = matrix(rnorm(K*p, 0, sigmaC), K, p)
eta = rnorm(K)
pi = (exp(eta)+1)/sum(exp(eta)+1)
z = t(rmultinom(m + n, 1, pi))
x = crossprod(t(z), c) + matrix(rnorm((m + n)*p, 0, sigmaX), m + n, p)
y = rowSums(z*(crossprod(t(x), beta))) + rnorm(m + n, 0, sigmaY)
```

```
x.train = x[1:n, ]
y.train = y[1:n]
x.test = x[n + 1:m, ]
y.test = y[n + 1:m]


# Example 1: Use clustering to fit the customized training model to training
# and test data with no predefined test-set blocks

fit1 = customizedGlmnet(x.train, y.train, x.test, G = 3,
    family = "gaussian")

# Plot nonzero coefficients by group:
plot(fit1, lambda = 10)


# Example 2: If the test set has predefined blocks, use these blocks to define
# the customized training sets, instead of using clustering.
group.id = apply(z == 1, 1, which)[n + 1:m]

fit2 = customizedGlmnet(x.train, y.train, x.test, group.id)

# Plot nonzero coefficients by group:
plot(fit2, lambda = 10)
```

---

plot.cv.customizedGlmnet *visualize variables selected in each customized training subset, from a cross-validated model*

Description

Produces a plot, with a row for each customized training submodel, showing the variables selected in the subset, with variables along the horizonal axis. The lambda used is the one which minimizes CV error.

Usage

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

Arguments

x	a fitted `cv.customizedGlmnet` object
...	ignored

Author(s)

Scott Powers, Trevor Hastie, Robert Tibshirani

See Also

plot, cv.customizedGlmnet

Examples

```
require(glmnet)

# Simulate synthetic data

n = m = 150
p = 50
q = 5
K = 3
sigmaC = 10
sigmaX = sigmaY = 1
set.seed(5914)

beta = matrix(0, nrow = p, ncol = K)
for (k in 1:K) beta[sample(1:p, q), k] = 1
c = matrix(rnorm(K*p, 0, sigmaC), K, p)
eta = rnorm(K)
pi = (exp(eta)+1)/sum(exp(eta)+1)
z = t(rmultinom(m + n, 1, pi))
x = crossprod(t(z), c) + matrix(rnorm((m + n)*p, 0, sigmaX), m + n, p)
y = rowSums(z*(crossprod(t(x), beta))) + rnorm(m + n, 0, sigmaY)

x.train = x[1:n, ]
y.train = y[1:n]
x.test = x[n + 1:m, ]
y.test = y[n + 1:m]
foldid = sample(rep(1:10, length = nrow(x.train)))


# Example 1: Use clustering to fit the customized training model to training
# and test data with no predefined test-set blocks

fit1 = cv.customizedGlmnet(x.train, y.train, x.test, Gs = c(1, 2, 3, 5),
    family = "gaussian", foldid = foldid)

# Print the optimal number of groups and value of lambda:
fit1$G.min
fit1$lambda.min

# Print the customized training model fit:
fit1

# Compute test error using the predict function:
mean((y[n + 1:m] - predict(fit1))^2)

# Plot nonzero coefficients by group:
plot(fit1)
```

```
# Example 2: If the test set has predefined blocks, use these blocks to define
# the customized training sets, instead of using clustering.
foldid = apply(z == 1, 1, which)[1:n]
group.id = apply(z == 1, 1, which)[n + 1:m]

fit2 = cv.customizedGlmnet(x.train, y.train, x.test, group.id, foldid = foldid)

# Print the optimal value of lambda:
fit2$lambda.min

# Print the customized training model fit:
fit2

# Compute test error using the predict function:
mean((y[n + 1:m] - predict(fit2))^2)

# Plot nonzero coefficients by group:
plot(fit2)


# Example 3: If there is no test set, but the training set is organized into
# blocks, you can do cross validation with these blocks as the basis for the
# customized training sets.

fit3 = cv.customizedGlmnet(x.train, y.train, foldid = foldid)

# Print the optimal value of lambda:
fit3$lambda.min

# Print the customized training model fit:
fit3

# Compute test error using the predict function:
mean((y[n + 1:m] - predict(fit3))^2)

# Plot nonzero coefficients by group:
plot(fit3)
```

`predict.customizedGlmnet` *make predictions from a* `customizedGlmnet` *object*

Description

Returns predictions for the test set provided at the time of fitting the `customizedGlmnet` object.

Usage

```
## S3 method for class 'customizedGlmnet'
```

```
predict(object, lambda,
  type = c('response', 'class'), ...)
```

Arguments

`object`	a fitted `customizedGlmnet` object
`lambda`	regularization parameter
`type`	Type of prediction, currently only "response" and "class" are supported. Type "response" returns fitted values for "gaussian" family and fitted probabilities for "binomial" and "multinomial" families. Type "class" applies only to "binomial" and "multinomial" families and returns the class with the highest fitted probability.
`...`	ignored

Value

a vector of predictions corresponding to the test data input to the model at the time of fitting

Author(s)

Scott Powers, Trevor Hastie, Robert Tibshirani

References

Scott Powers, Trevor Hastie and Robert Tibshirani (2015) "Customized training with an application to mass specrometric imaging of gastric cancer data." Annals of Applied Statistics 9, 4:1709-1725.

See Also

`predict`, `customizedGlmnet`

Examples

```
require(glmnet)

# Simulate synthetic data

n = m = 150
p = 50
q = 5
K = 3
sigmaC = 10
sigmaX = sigmaY = 1
set.seed(5914)

beta = matrix(0, nrow = p, ncol = K)
for (k in 1:K) beta[sample(1:p, q), k] = 1
c = matrix(rnorm(K*p, 0, sigmaC), K, p)
eta = rnorm(K)
pi = (exp(eta)+1)/sum(exp(eta)+1)
```

```
z = t(rmultinom(m + n, 1, pi))
x = crossprod(t(z), c) + matrix(rnorm((m + n)*p, 0, sigmaX), m + n, p)
y = rowSums(z*(crossprod(t(x), beta))) + rnorm(m + n, 0, sigmaY)

x.train = x[1:n, ]
y.train = y[1:n]
x.test = x[n + 1:m, ]
y.test = y[n + 1:m]


# Example 1: Use clustering to fit the customized training model to training
# and test data with no predefined test-set blocks

fit1 = customizedGlmnet(x.train, y.train, x.test, G = 3,
    family = "gaussian")

# Compute test error using the predict function:
mean((y.test - predict(fit1, lambda = 10))^2)


# Example 2: If the test set has predefined blocks, use these blocks to define
# the customized training sets, instead of using clustering.
group.id = apply(z == 1, 1, which)[n + 1:m]

fit2 = customizedGlmnet(x.train, y.train, x.test, group.id)

# Compute test error using the predict function:
mean((y.test - predict(fit2, lambda = 10))^2)
```

---

predict.cv.customizedGlmnet
make predictions from a `cv.customizedGlmnet` *object*

Description

Returns predictions for test set provided at time of fitting, using regulariztion parameter which minimizes CV error

Usage

```
## S3 method for class 'cv.customizedGlmnet'
predict(object, ...)
```

Arguments

`object`	a fitted `cv.customizedGlmnet` object
`...`	additional arguments to be passed to `predict.customizedGlmnet`

Value

a vector of predictions corresponding to the test set provided when the model was fit. The results are for the regularization parameter chosen by cross-validation

Author(s)

Scott Powers, Trevor Hastie, Robert Tibshirani

References

Scott Powers, Trevor Hastie and Robert Tibshirani (2015) "Customized training with an application to mass specrometric imaging of gastric cancer data." Annals of Applied Statistics 9, 4:1709-1725.

See Also

predict, cv.customizedGlmnet

Examples

```
require(glmnet)

# Simulate synthetic data

n = m = 150
p = 50
q = 5
K = 3
sigmaC = 10
sigmaX = sigmaY = 1
set.seed(5914)

beta = matrix(0, nrow = p, ncol = K)
for (k in 1:K) beta[sample(1:p, q), k] = 1
c = matrix(rnorm(K*p, 0, sigmaC), K, p)
eta = rnorm(K)
pi = (exp(eta)+1)/sum(exp(eta)+1)
z = t(rmultinom(m + n, 1, pi))
x = crossprod(t(z), c) + matrix(rnorm((m + n)*p, 0, sigmaX), m + n, p)
y = rowSums(z*(crossprod(t(x), beta))) + rnorm(m + n, 0, sigmaY)

x.train = x[1:n, ]
y.train = y[1:n]
x.test = x[n + 1:m, ]
y.test = y[n + 1:m]
foldid = sample(rep(1:10, length = nrow(x.train)))


# Example 1: Use clustering to fit the customized training model to training
# and test data with no predefined test-set blocks

fit1 = cv.customizedGlmnet(x.train, y.train, x.test, Gs = c(1, 2, 3, 5),
    family = "gaussian", foldid = foldid)
```

```
# Print the optimal number of groups and value of lambda:
fit1$G.min
fit1$lambda.min

# Print the customized training model fit:
fit1

# Compute test error using the predict function:
mean((y[n + 1:m] - predict(fit1))^2)

# Plot nonzero coefficients by group:
plot(fit1)


# Example 2: If the test set has predefined blocks, use these blocks to define
# the customized training sets, instead of using clustering.
foldid = apply(z == 1, 1, which)[1:n]
group.id = apply(z == 1, 1, which)[n + 1:m]

fit2 = cv.customizedGlmnet(x.train, y.train, x.test, group.id, foldid = foldid)

# Print the optimal value of lambda:
fit2$lambda.min

# Print the customized training model fit:
fit2

# Compute test error using the predict function:
mean((y[n + 1:m] - predict(fit2))^2)

# Plot nonzero coefficients by group:
plot(fit2)


# Example 3: If there is no test set, but the training set is organized into
# blocks, you can do cross validation with these blocks as the basis for the
# customized training sets.

fit3 = cv.customizedGlmnet(x.train, y.train, foldid = foldid)

# Print the optimal value of lambda:
fit3$lambda.min

# Print the customized training model fit:
fit3

# Compute test error using the predict function:
mean((y[n + 1:m] - predict(fit3))^2)

# Plot nonzero coefficients by group:
plot(fit3)
```

`predict.singleton` *make predictions from a "singleton" object*

Description

Returns the value stored in the singleton. Intended for internal use only.

Usage

```
## S3 method for class 'singleton'
predict(object, type = c('response', 'class', 'nonzero'),
  ...)
```

Arguments

`object`	an object of class `singleton`
`type`	Type of prediction to be returned, "response" or "class"
`...`	ignored

Value

The value of the singleton

Author(s)

Scott Powers, Trevor Hastie, Robert Tibshirani

See Also

`predict`

`print.customizedGlmnet` *print the summary of a fitted* `customizedGlmnet` *object*

Description

Print the numbers of training observations and test observations in each submodel of the `customizedGlmnet` fit

Usage

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

Arguments

x	fitted `customizedGlmnet` object
...	ignored

Author(s)

Scott Powers, Trevor Hastie, Robert Tibshirani

See Also

`print`, `customizedGlmnet`

Examples

```
require(glmnet)

# Simulate synthetic data

n = m = 150
p = 50
q = 5
K = 3
sigmaC = 10
sigmaX = sigmaY = 1
set.seed(5914)

beta = matrix(0, nrow = p, ncol = K)
for (k in 1:K) beta[sample(1:p, q), k] = 1
c = matrix(rnorm(K*p, 0, sigmaC), K, p)
eta = rnorm(K)
pi = (exp(eta)+1)/sum(exp(eta)+1)
z = t(rmultinom(m + n, 1, pi))
x = crossprod(t(z), c) + matrix(rnorm((m + n)*p, 0, sigmaX), m + n, p)
y = rowSums(z*(crossprod(t(x), beta))) + rnorm(m + n, 0, sigmaY)

x.train = x[1:n, ]
y.train = y[1:n]
x.test = x[n + 1:m, ]
y.test = y[n + 1:m]


# Example 1: Use clustering to fit the customized training model to training
# and test data with no predefined test-set blocks

fit1 = customizedGlmnet(x.train, y.train, x.test, G = 3,
    family = "gaussian")

# Print the customized training model fit:
fit1
```

```
# Example 2: If the test set has predefined blocks, use these blocks to define
# the customized training sets, instead of using clustering.
group.id = apply(z == 1, 1, which)[n + 1:m]

fit2 = customizedGlmnet(x.train, y.train, x.test, group.id)

# Print the customized training model fit:
fit2
```

---

print.cv.customizedGlmnet

print a "cv.customizedGlmnet" object

Description

Print the number of customized training subsets chosen by cross-validation and the number of variables selected in each training subset.

Usage

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

Arguments

x	a fitted `cv.customizedGlmnet` object
...	ignored

Author(s)

Scott Powers, Trevor Hastie, Robert Tibshirani

See Also

`print`, `cv.customizedGlmnet`

Examples

```
require(glmnet)

# Simulate synthetic data

n = m = 150
p = 50
q = 5
K = 3
sigmaC = 10
sigmaX = sigmaY = 1
set.seed(5914)
```

```
beta = matrix(0, nrow = p, ncol = K)
for (k in 1:K) beta[sample(1:p, q), k] = 1
c = matrix(rnorm(K*p, 0, sigmaC), K, p)
eta = rnorm(K)
pi = (exp(eta)+1)/sum(exp(eta)+1)
z = t(rmultinom(m + n, 1, pi))
x = crossprod(t(z), c) + matrix(rnorm((m + n)*p, 0, sigmaX), m + n, p)
y = rowSums(z*(crossprod(t(x), beta))) + rnorm(m + n, 0, sigmaY)

x.train = x[1:n, ]
y.train = y[1:n]
x.test = x[n + 1:m, ]
y.test = y[n + 1:m]
foldid = sample(rep(1:10, length = nrow(x.train)))


# Example 1: Use clustering to fit the customized training model to training
# and test data with no predefined test-set blocks

fit1 = cv.customizedGlmnet(x.train, y.train, x.test, Gs = c(1, 2, 3, 5),
    family = "gaussian", foldid = foldid)

# Print the optimal number of groups and value of lambda:
fit1$G.min
fit1$lambda.min

# Print the customized training model fit:
fit1

# Compute test error using the predict function:
mean((y[n + 1:m] - predict(fit1))^2)

# Plot nonzero coefficients by group:
plot(fit1)


# Example 2: If the test set has predefined blocks, use these blocks to define
# the customized training sets, instead of using clustering.
foldid = apply(z == 1, 1, which)[1:n]
group.id = apply(z == 1, 1, which)[n + 1:m]

fit2 = cv.customizedGlmnet(x.train, y.train, x.test, group.id, foldid = foldid)

# Print the optimal value of lambda:
fit2$lambda.min

# Print the customized training model fit:
fit2

# Compute test error using the predict function:
mean((y[n + 1:m] - predict(fit2))^2)
```

```
# Plot nonzero coefficients by group:
plot(fit2)


# Example 3: If there is no test set, but the training set is organized into
# blocks, you can do cross validation with these blocks as the basis for the
# customized training sets.

fit3 = cv.customizedGlmnet(x.train, y.train, foldid = foldid)

# Print the optimal value of lambda:
fit3$lambda.min

# Print the customized training model fit:
fit3

# Compute test error using the predict function:
mean((y[n + 1:m] - predict(fit3))^2)

# Plot nonzero coefficients by group:
plot(fit3)
```

`Vowel` *Vowel Recognition*

Description

Speaker independent recognition of the eleven steady state vowels of British English using a specified training set of lpc derived log area ratios.

Format

A data frame with 990 observations on the following 12 variables.

`y` Class label indicating vowel spoken

`subset` a factor with levels `test` `train`

`x.1` a numeric vector

`x.2` a numeric vector

`x.3` a numeric vector

`x.4` a numeric vector

`x.5` a numeric vector

`x.6` a numeric vector

`x.7` a numeric vector

`x.8` a numeric vector

`x.9` a numeric vector

`x.10` a numeric vector

Details

The speech signals were low pass filtered at 4.7kHz and then digitised to 12 bits with a 10kHz sampling rate. Twelfth order linear predictive analysis was carried out on six 512 sample Hamming windowed segments from the steady part of the vowel. The reflection coefficients were used to calculate 10 log area parameters, giving a 10 dimensional input space. For a general introduction to speech processing and an explanation of this technique see Rabiner and Schafer [RabinerSchafer78].

Each speaker thus yielded six frames of speech from eleven vowels. This gave 528 frames from the eight speakers used to train the networks and 462 frames from the seven speakers used to test the networks.

The eleven vowels, along with words demonstrating their sound, are: i (heed) I (hid) E (head) A (had) a: (hard) Y (hud) O (hod) C: (hoard) U (hood) u: (who'd) 3: (heard)

Source

https://archive.ics.uci.edu/ml/machine-learning-databases/undocumented/connectionist-bench/vowel/

References

D. H. Deterding, 1989, University of Cambridge, "Speaker Normalisation for Automatic Speech Recognition", submitted for PhD.

Examples

```
data(Vowel)
summary(Vowel)
```

Index