Package ‘bmrm’

February 19, 2018

Type Package
Title Bundle Methods for Regularized Risk Minimization Package
Version 3.7
Date 2018-01-22
Depends R (>= 3.0.2)
Imports lpSolve, LowRankQP, matrixStats
Suggests knitr
VignetteBuilder knitr
Author Julien Prados
Maintainer Julien Prados <julien.prados@unige.ch>
Copyright 2017, University of Geneva
Description Bundle methods for minimization of convex and non-convex risk under L1 or L2 regularization. Implements the algorithm proposed by Teo et al. (JMLR 2010) as well as the extension proposed by Do and Artieres (JMLR 2012). The package comes with lot of loss functions for machine learning which make it powerful for big data analysis. The applications includes: structured prediction, linear SVM, multi-class SVM, f-beta optimization, ROC optimization, ordinal regression, quantile regression, epsilon insensitive regression, least mean square, logistic regression, least absolute deviation regression (see package examples), etc... all with L1 and L2 regularization.
License GPL-3
RoxygenNote 6.0.1
NeedsCompilation no
Repository CRAN
Date/Publication 2018-02-19 12:24:20 UTC

R topics documented:

  balanced.cv.fold .......................................................... 2
balanced.cv.fold

Split a dataset for Cross Validation taking into account class balance

Usage

balanced.cv.fold(y, num.cv = 10)

Arguments

y = the class labels of each sample of the dataset
num.cv = number of cross validation required

Value

a factor of num.cv levels that assign to each sample a test fold
**binaryClassificationLoss**

*Loss functions for binary classification*

**Description**

Loss functions for binary classification

**Usage**

- `hingeLoss(x, y, loss.weights = 1)`
- `logisticLoss(x, y, loss.weights = 1)`
- `rocLoss(x, y)`
- `fbetaLoss(x, y, beta = 1)`

**Arguments**

- `x` matrix of training instances (one instance by row)
- `y` a logical vector representing the training labels for each instance in `x`
- `loss.weights` numeric vector of loss weights to incur for each instance of `x`. Vector length should match length `y`, but values are cycled if not of identical size.
- `beta` a numeric value setting the beta parameter is the f-beta score

**Value**

a function taking one argument `w` and computing the loss value and the gradient at point `w`

**Functions**

- `hingeLoss`: Hinge Loss for Linear Support Vector Machine (SVM)
- `logisticLoss`: logistic regression
- `rocLoss`: Find linear weights maximize area under its ROC curve
- `fbetaLoss`: F-beta score loss function

**References**

Teo et al. A Scalable Modular Convex Solver for Regularized Risk Minimization. KDD 2007

**See Also**

nrbm
Examples

```r
x <- cbind(intercept=100, data.matrix(iris[1:2]))
w <- nrbm(hingeLoss(x, iris$Species=="setosa")); predict(w, x)
w <- nrbm(logisticLoss(x, iris$Species=="setosa")); predict(w, x)
w <- nrbm(rocLoss(x, iris$Species=="setosa")); predict(w, x)
w <- nrbm(fbetaLoss(x, iris$Species=="setosa")); predict(w, x)
```

---

**costMatrix**  
*Compute or check the structure of a cost matrix*

**Description**  
Compute or check the structure of a cost matrix

**Usage**  
```r
costMatrix(y, C = c("0/1", "linear"))
```

**Arguments**

- `y`  
a factor representing the labels of the instances

- `C`  
either a cost matrix to check for consistency with labels in `y`, or a character string defining the standard matrix to compute. If a character string the accepted values are "0/1" for a 0-1 cost matrix or "linear" for linear cost.

**Value**

the cost matrix object

**See Also**

- `nrbm`, `ordinalRegressionLoss`

---

**gradient**  
*Return or set gradient attribute*

**Description**

Return or set gradient attribute
Usage

gradient(x, ...)

## Default S3 method:
gradient(x, ...)

gradient(x, ...) <- value

## Default S3 replacement method:
gradient(x, ...) <- value

Arguments

x any R object
...
additional parameters
value new gradient value to set

Details

gradient attribute is used by loss/risk function to return the gradient of the function at a given point together with the function value

Value

attr(x,"gradient")

---

**hclust.fca**

*Find first common ancestor of 2 nodes in an hclust object*

Description

Find first common ancestor of 2 nodes in an hclust object

Usage

hclust.fca(hc, a, b)

Arguments

hc an hclust object
a an integer vector with the first leaf node
b an integer vector with the second leaf node (same length as a)

Value

an integer vector of the same length as a and b identifying the first common ancestors of a and b
Author(s)

Julien Prados

Examples

```r
hc <- hclust(dist(USArrests), "complete")
plot(hc)
A <- outer(seq_along(hc$order), seq_along(hc$order), hclust.fca, hc=hc)
H <- array(hc$height[A], dim(A))
image(H[hc$order, hc$order])
image(A[hc$order, hc$order])
```

is.convex

Return or set is.convex attribute

Description

Return or set is.convex attribute

Usage

```r
is.convex(x, ...)
```

## Default S3 method:
```r
is.convex(x, ...)
```

is.convex(x, ...) <- value

## Default S3 replacement method:
```r
is.convex(x, ...) <- value
```

Arguments

- `x` any R object
- `...` additional parameters
- `value` new loss value to set

Details

is.convex attribute is used by loss/risk function to determine if it is convex

Value

```r
attr(x,"is.convex")
```
iterative.hclust

Perform multiple hierarchical clustering on random subsets of a dataset

Description

Perform multiple hierarchical clustering on random subsets of a dataset

Usage

iterative.hclust(x, seeds = 1:100, mc.cores = getOption("mc.cores", 1L),
    row.rate = 0.3, col.rate = 0.1, max.cluster = 10,
    hc.method = function(x) {
        hclust(dist(prcomp(x)$x[, 1:4]), method = "complete")
    })

Arguments

- **x**: the numeric matrix containing the data to cluster (one instance per row)
- **seeds**: a vector of random seed to use.
- **mc.cores**: number of core to use for parallelization
- **row.rate**, **col.rate**: numeric value in [0,1] to specify the proportion of instance (resp. feature) to subset at each random iteration.
- **max.cluster**: upper bound on the number of expected cluster (can by +Inf).
- **hc.method**: a clustering method of arity 1, taking as input a random subset of the input matrix x and returning an hclust object

Value

A list of 3 square matrices N,H,K of size nrow(x): N is the number of time each pair of instance as been seen in the random subsets; H is the average heights where the pair of sample as been merged in the tree; K is the average number of split possible into the trees still preserving the two samples into the same cluster.

Author(s)

Julien Prados
linearRegressionLoss  

Loss functions to perform a regression

Description
Loss functions to perform a regression

Usage
lmsRegressionLoss(x, y, loss.weights = 1)
ladRegressionLoss(x, y, loss.weights = 1)
quantileRegressionLoss(x, y, q = 0.5, loss.weights = 1)
epsilonInsensitiveRegressionLoss(x, y, epsilon, loss.weights = 1)

Arguments

x       matrix of training instances (one instance by row)
y       numeric vector of values representing the training labels for each instance in x
loss.weights       numeric vector of loss weights to incur for each instance of x. Vector length should match length(y), but values are cycled if not of identical size.
q       a numeric value in the range [0-1] defining quantile value to consider
epsilon       a numeric value setting tolerance of the epsilon-regression

Value
a function taking one argument w and computing the loss value and the gradient at point w

Functions

- lmsRegressionLoss: Least Mean Square regression
- ladRegressionLoss: Least Absolute Deviation regression
- quantileRegressionLoss: Quantile Regression
- epsilonInsensitiveRegressionLoss: epsilon-insensitive regression (Vapnik et al. 1997)

References
Teo et al. Bundle Methods for Regularized Risk Minimization JMLR 2010

See Also
nrbm
Examples

```r
x <- cbind(intercept=100, data.matrix(iris[1:2]))
y <- iris[,3]
w <- nrbm(lmsRegressionLoss(x, y))
w <- nrbm(ladRegressionLoss(x, y))
w <- nrbm(quantileRegressionLoss(x, y, q=0.5))
w <- nrbm(epsilonInsensitiveRegressionLoss(x, y, epsilon=1))
```

lpSVM

**Linearly Programmed SVM**

Description

Linearly Programmed L1-loss Linear Support Vector Machine with L1 regularization

Usage

```r
svmLP(x, y, LAMBDA = 1, loss.weights = 1)
```

## S3 method for class 'svmLP'

```r
predict(object, x, ...)
```

```r
svmMulticlassLP(x, y, LAMBDA = 1, loss.weights = 1)
```

## S3 method for class 'svmMLP'

```r
predict(object, x, ...)
```

Arguments

- **x**: a numeric data matrix to predict
- **y**: a response factor for each row of x. It must be a 2 levels factor for svmLP, or a >=2 levels factor for svmMulticlassLP
- **LAMBDA**: control the regularization strength in the optimization process. This is the value used as coefficient of the regularization term.
- **loss.weights**: numeric vector of loss weights to incure for each instance of x. Vector length should match length(y), but values are cycled if not of identical size.
- **object**: an object of class svmLP or svmMLP
- **...**: unused, present to satisfy the generic predict() prototype

Details

svmLP solves a linear program implementing a linear SVM with L1 regularization and L1 loss. It solves: min_w LAMBDA*|w| + sum_i(e_i); s.t. y_i * <w.x_i> >= 1-e_i; e_i >= 0 where |w| is the L1-norm of w

svmMulticlassLP solves a linear program implementing multiclass-SVM with L1 regularization and L1 loss. It solves: min_w LAMBDA*|w| + sum_i(e_i); s.t. <w.x_i> - <w.x_j> >= 1-e_i; e_i >= 0 where |w| is the L1-norm of w
Value

the optimized weights matrix, with class svmLP
predict() return predictions for row of x, with an attribute "decision.value"
predict() return predictions for row of x, with an attribute "decision.value"

Functions

- svmlp: linear programm solving binary-SVM with L1-regularization and L1-norm
- svmMulticlassLP: linear programm solving multiclass-SVM with L1-regularization and L1-norm

Author(s)

Julien Prados

Examples

```r
x <- cbind(100, data.matrix(iris[1:4]))
y <- iris$Species
w <- svmMulticlassLP(x, y)
   table(predict(w, x), y)

w <- svmlp(x, y == "setosa")
   table(predict(w, x), y)
```

---

lvalue

Return or set lvalue attribute

Description

Return or set lvalue attribute

Usage

```r
lvalue(x, ...)

## Default S3 method:
lvalue(x, ...)

lvalue(x, ...) <- value

## Default S3 replacement method:
lvalue(x, ...) <- value
```
Arguments

x  any R object
... additional parameters
value new loss value to set

Details

lvalue attribute is used by loss/risk function to return the loss value of the function at a given point together with the function gradient

Value

attr(x,"lvalue")

mmc

Convenient wrapper function to solve max-margin clustering problem on a dataset

Description

Solve max-margin clustering problem with multiple random starting points to avoid being trapped by local minima. The random starting points are determined by randomly assigning N0 samples to each cluster and solving for multi-class SVM.

Usage

mmc(x, k = 2L, N0 = 2L, LAMBDA = 1, seeds = 1:50, 
nrbmArgsSvm = list(maxCP = 10L, MAX_ITER = 100L), nrbmArgsMmc = list(maxCP 
= 20L, MAX_ITER = 300L), mc.cores = getOption("mc.cores", 1L), ...)

Arguments

x  numeric matrix representing the dataset (one sample per row)
k  an integer specifying number of clusters to find
N0  number of instance to randomly assign per cluster when determining a random starting point. The classification dataset it defines is used to train a multi-class SVM whose solution is used as the starting point of current MMC iteration.
LAMBDA  the complexity parameter for nrbm()
seeds  the random seeds to use
nrbmArgsSvm  arguments to nrbm() when solving for multi-class SVM problem
nrbmArgsMmc  arguments to nrbm() when solving for max-margin clustering problem
mc.cores  number of core to use when running the random iterations in parallel
...  additional arguments are passed to mmcLoss()
mmcLoss

the MMC model matrix

Examples

# -- Prepare a 2D dataset to cluster with an intercept
x <- cbind(intercept=100,scale(data.matrix(iris[c(1,3)])),center=TRUE,scale=FALSE)

# -- Find max-margin clusters
y <- mmc(x,k=3,LAMBDA=0.001,minClusterSize=10,seeds=5)
table(y,iris$Species)

# -- Plot the dataset and the MMC decision boundaries
gx <- seq(min(x[,2]),max(x[,2]),length=100)
gy <- seq(min(x[,3]),max(x[,3]),length=100)
Y <- outer(gx,gy,function(a,b)(predict(y,cbind(100,a,b))))
image(gx,gy,Y,asp=1,main="MMC clustering",xlab=colnames(x)[1],ylab=colnames(x)[2])
points(x[,1],pch=19+y)

mmcLoss (Loss function for max-margin clustering)

Description

Loss function for max-margin clustering

Usage

mmcLoss(x, k = 3L, minClusterSize = 1L, groups = matrix(logical(0),
nrow(x), 0), minGroupOverlap = matrix(integer(0), k, ncol(groups)),
weight = 1/nrow(x))

Arguments

x numeric matrix representing the dataset (one sample per row)
k an integer specifying number of clusters to find
minClusterSize an integer vector specifying the minimum number of sample per cluster. Given
values are recycled if necessary to have one value per cluster.
groups a logical matrix for instance grouping (groups[i,j] TRUE when sample i belong
to group j).
minGroupOverlap an integer matrix specifying the minimum number of instance per cluster for each
group.
weight a weight vector for each instance

Value

the loss function to optimize for max margin clustering of the given dataset
**nrbm**

Convex and non-convex risk minimization with L2 regularization and limited memory

---

**Description**

Use algorithm of Do and Artieres, JMLR 2012 to find w minimizing: 
\[ f(w) = 0.5 \times \text{LAMBDA} \times \text{l2norm}(w) + \text{riskFun}(w) \]
where riskFun is either a convex or a non-convex risk function.

**Usage**

```r
nrbm(riskFun, LAMBDA = 1, MAX_ITER = 1000L, EPSILON_TOL = 0.01, w0 = 0,
     maxCP = 50L, convexRisk = is.convex(riskFun), LowRankQP.method = "LU",
     line.search = !convexRisk)
```

```r
nrbmL1(riskFun, LAMBDA = 1, MAX_ITER = 300L, EPSILON_TOL = 0.01, w0 = 0,
       maxCP = +Inf, line.search = FALSE)
```

**Arguments**

- `riskFun`: the risk function to use in the optimization (e.g.: `hingeLoss`, `softMarginVectorLoss`). The function must evaluate the loss value and its gradient for a given point vector (w). The function must return the given point vector w, with attributes "lvalue" and "gradient" set.
- `LAMBDA`: control the regularization strength in the optimization process. This is the value used as coefficient of the regularization term.
- `MAX_ITER`: the maximum number of iteration to perform. The function stop with a warning message if the number of iteration exceed this value.
- `EPSILON_TOL`: a numeric value between 0 and 1 controlling stopping criteria: the optimization end when the ratio between the optimization gap and the objective value is below this threshold.
- `w0`: initial weight vector where optimization start.
- `maxCP`: maximal number of cutting plane to use to limit memory footprint.
- `convexRisk`: a length 1 logical telling if the risk function riskFun is convex. If TRUE, use CRBM algorithm; if FALSE use NRBM algorithm from Do and Artieres, JMLR 2012.
- `LowRankQP.method`: a single character value defining the method used by LowRankQP (should be either "LU" or "CHOL").
- `line.search`: a logical, when TRUE use line search to speed up convergence.

**Value**

the optimal weight vector (w)
Functions

- `nrbm`: original L2-regularized version of `nrbm`
- `nrbml1`: L1-regularized version of `nrbm` that can only handle convex risk

References

Do and Artieres Regularized Bundle Methods for Convex and Non-Convex Risks JMLR 2012

Examples

```r
# -- Create a 2D dataset with the first 2 features of iris, with binary labels
x <- data.matrix(iris[1:2])

# -- Add a constant dimension to the dataset to learn the intercept
x <- cbind(intercept=1000,x)

# -- train scalar prediction models with maxMarginLoss and fBetaLoss
models <- list(
  svm_L1 = nrbml1(hingeLoss(x,iris$Species=="setosa"),LAMBDAD=1),
  svm_L2 = nrbm(hingeLoss(x,iris$Species=="setosa"),LAMBDAD=1),
  f1_L1 = nrbml1(fBetaLoss(x,iris$Species=="setosa"),LAMBDAD=1),
  tsvm_L2 = nrbm(tsvmLoss(x,
    ifelse(iris$Species=="versicolor",NA,iris$Species=="setosa")),
    LAMBDAD=1)
)

# -- Plot the dataset and the predictions
plot(x[,-1],pch=ifelse(iris$Species=="setosa",1,2),main="dataset & hyperplanes")
legend('bottomright',legend=names(models),col=seq_along(models),lty=1,cex=0.75,lwd=3)
for(i in seq_along(models)) {
  w <- models[[i]]
  if (w[3]!=0) abline(-w[1]*x[3],-w[2]/w[3],col=i,lwd=3)
}

# -- fit a least absolute deviation linear model on a synthetic dataset
# -- containing 196 meaningful features and 4 noisy features. Then
# -- check if the model has detected the noise
set.seed(123)
X <- matrix(rnorm(4000*200), 4000, 200)
beta <- c(rep(1,ncol(X)-4),0,0,0,0)
Y <- X%*%beta + rnorm(nrow(X))
w <- nrbm(ladRegressionLoss(x/100,Y/100),maxCP=50)
barplot(as.vector(w))
```

ontologyLoss

**Ontology Loss Function**
**ordinalRegressionLoss**

**Description**

Ontology loss function may be used when the class labels are organized has an ontology structure.

**Usage**

```r
ontologyLoss(x, y, l = 1 - table(seq_along(y), y), dag = diag(nlevels(y)))
```

**Arguments**

- `x` instance matrix, where \( x(t) \) defines the features of instance \( t \)
- `y` target vector where \( y(t) \) is an integer encoding target of \( x(t) \)
- `l` loss matrix. \( l(t, p(t)) \) must be the loss for predicting target \( p(t) \) instead of \( y(t) \) for instance \( t \). By default, the parameter is set to a 0/1 loss matrix.
- `dag` a numeric matrix defining the path in the Direct Acyclic Graph (DAG) to each class label.

**Value**

a function taking one argument \( w \) and computing the loss value and the gradient at point \( w \).

**References**


**Examples**

```r
# -- Load the data
x <- cbind(intercept=100, data.matrix(iris[1:4]))
dag <- matrix(nrow=nlevels(iris$Species), byrow=TRUE, dimnames=list(levels(iris$Species)), c(1,0,0,0,
0,1,1,0,
0,1,0,1))
w <- nrbm(ontologyLoss(x, iris$Species, dag=dag))
table(predict(w,x), iris$Species)
```

---

**ordinalRegressionLoss**  The loss function for ordinal regression

**Description**

The loss function for ordinal regression.

**Usage**

```r
ordinalRegressionLoss(x, y, C = "0/1", impl = c("loglin", "quadratic"))
```
**Arguments**

- **x**
  - matrix of training instances (one instance by row)

- **y**
  - integer vector of positive values (>=1) representing the training labels for each instance in x

- **C**
  - the cost matrix to use, C[i,j] being the cost for predicting label i instead of label j.

- **impl**
  - either the string "loglin" or "quadratic", that define the implementation to use for the computation of the loss.

**Value**

a function taking one argument w and computing the loss value and the gradient at point w

**References**

Teo et al. Bundle Methods for Regularized Risk Minimization JMLR 2010

**See Also**

nrbm

**Examples**

```r
# -- Load the data
x <- data.matrix(iris[1:4])
y <- as.integer(iris$Species)

# -- Train the model
w <- nrbm(ordinalRegressionLoss(x,y),LAMBDAL=0.001,EPSILON_TOL=0.0001)
w2 <- nrbm(ordinalRegressionLoss(x,y,impl="quadratic"),LAMBDAL=0.001,EPSILON_TOL=0.0001)

# -- plot predictions
f <- x %*% w
f2 <- x %*% w2
layout(1:2)
plot(y,f)
plot(f,f2,main="compare predictions of quadratic and loglin implementations")

# -- Compute accuracy
ij <- expand.grid(i=seq(nrow(x)),j=seq(nrow(x)))
n <- tapply(f[ij[1]] - f[ij[2]]>0,list(y[ij[1]],y[ij[2]]),sum)
N <- table(y[ij[1]],y[ij[2]])
print(n/N)
```
**predict.mmc**

Predict class of new instances according to a mmc model

---

**Usage**

```r
## S3 method for class 'mmc'
predict(object, x, ...)  
```

**Arguments**

- `object`: a mmc object
- `x`: a matrix similar to the dataset used, i.e. where rows are instances for which class must be predicted
- `...`: unused, present to satisfy the generic `predict()` prototype

**Value**

A integer vector whose length match `nrow(x)` and containing the predicted class for each of the given instances.

---

**roc.stat**

Compute statistics for ROC curve plotting

---

**Description**

Compute statistics for ROC curve plotting

**Usage**

`roc.stat(f, y)`

**Arguments**

- `f`: decision value for each instance
- `y`: a logical that specify binary labels

**Value**

A data.frame() that compute for each threshold value `f` roc curve statistics: TP, FP, TN, FN, FPR, TPR, sensitivity, specificity, precision, recall, accuracy
**Author(s)**

Julien Prados, adapted from Bob Horton code

**Examples**

```r
x <- cbind(data.matrix(iris[1:4]))
w <- nrblm(rocloss(x, iris$Species == "versicolor"), LAMBDA=0.01)
with(rocr.stat(x, w, iris$Species == "versicolor"), plot(FPR, TPR, type="l"))
with(rocr.stat(-x[,2], iris$Species == "versicolor"), lines(FPR, TPR, col="blue"))
```

---

**softMarginVectorLoss**

*Soft Margin Vector Loss function for multiclass SVM*

**Description**

Soft Margin Vector Loss function for multiclass SVM

**Usage**

```r
softMarginVectorLoss(x, y, l = 1 - table(seq_along(y), y))
```

**Arguments**

- `x`: instance matrix, where `x(t,)` defines the features of instance `t`
- `y`: target vector where `y(t)` is an integer encoding target of `x(t,)`
- `l`: loss matrix. `l(t,p(t))` must be the loss for predicting target `p(t)` instead of `y(t)` for instance `t`. By default, the parameter is set to character value "0/1" so that the loss is set to a 0/1 loss matrix.

**Value**

a function taking one argument `w` and computing the loss value and the gradient at point `w`

**References**

Teo et al. A Scalable Modular Convex Solver for Regularized Risk Minimization. KDD 2007

**Examples**

```r
# -- Build a 2D dataset from iris, and add an intercept
x <- cbind(intercept=1, data.matrix(iris[1:2]))
y <- iris$Species

# -- build the multiclass SVM model
w <- nrblm(softMarginVectorLoss(x, y))
table(predict(w, x), y)

# -- Plot the dataset, the decision boundaries, the convergence curve, and the predictions
```
tsvmLoss

Non convex loss function for transductive SVM

description

Non convex loss function for transductive SVM

Usage

tsvmLoss(x, y, loss.weights = 1)

Arguments

x
matrix of training instances (one instance by row)
y
a logical vector representing the training labels for each instance in x. NA are
allowed when labels is unknown.
loss.weights
numeric vector of loss weights to incur for each instance of x. Vector length
should match length(y), but values are cycled if not of identical size.

Value

a function taking one argument w and computing the loss value and the gradient at point w

See Also

nrbm

Examples

  x <- cbind(intercept=100, data.matrix(iris[1:2]))
y <- iris$Species == "virginica"
y[iris$Species == "setosa"] <- NA
w <- nrbm(tsvmLoss(x, y), convexRisk=FALSE)
table(predict(w, x), iris$Species)
wolfe.linesearch  Wolfe Line Search

Description
Implements Wolfe Line Search algorithm. The code is inspired from Matlab code of Do and Artiere, but not tested. The function is not used yet, but might be used later to speed up bmrn/nrbm convergence.

Usage
wolfe.linesearch(f, x0, s0, ..., a1 = 0.5, amax = 1.1, c1 = 1e-04, c2 = 0.9, maxiter = 5L, f.adjust = identity)

Arguments

f  a function to minimize. It must accept as first argument a numeric vector representing the optimization point and return a numeric value, with gradient attribute setted
x0  initial search point
s0  direction of the search from x0
...  additional parameters passed to f()
a1  first step coefficient guess
amax  max coefficient value
c1  lower bound
c2  upper bound
maxiter  maximum number of iteration for this linesearch
f.adjust  an adjustment method to adjust lvalue and gradient of f

Value
the optimal point

Author(s)
Julien Prados

References
Do and Artieres Regularized Bundle Methods for Convex and Non-Convex Risks JMLR 2012

See Also
nrbm
Index

balanced.cv.fold, 2
binaryClassificationLoss, 3
costMatrix, 4
epsilonInsensitiveRegressionLoss
  (linearRegressionLoss), 8
fbetaLoss (binaryClassificationLoss), 3
gradient, 4
gradient<- (gradient), 4
hclust.fca, 5
hingeLoss (binaryClassificationLoss), 3
is.convex, 6
is.convex<- (is.convex), 6
iterative.hclust, 7
ladRegressionLoss
  (linearRegressionLoss), 8
linearRegressionLoss, 8
lmsRegressionLoss
  (linearRegressionLoss), 8
logisticLoss
  (binaryClassificationLoss), 3
lpSVM, 9
lvalue, 10
lvalue<- (lvalue), 10
mmc, 11
mmcLoss, 12
nrbm, 13, 20
nrbml1 (nrbm), 13
ontologyLoss, 14
ordinalRegressionLoss, 15
predict.mmc, 17
predict.svmLP (lpSVM), 9
quantileRegressionLoss
  (linearRegressionLoss), 8
roc.stat, 17
rocLoss (binaryClassificationLoss), 3
softMarginVectorLoss, 18
svmLP (lpSVM), 9
svmMulticlassLP (lpSVM), 9
tsvmLoss, 19
wolfe.linesearch, 20