Package 'bmrm'

October 12, 2022

Type Package

Title Bundle Methods for Regularized Risk Minimization Package

Version 4.1

Date 2019-04-03

Depends R (>= 3.0.2)

Imports methods, lpSolve, LowRankQP, matrixStats, Rcpp

Suggests knitr

VignetteBuilder knitr

Author Julien Prados

Maintainer Julien Prados <julien.prados@unige.ch>

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 6.1.0

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-04-03 14:00:03 UTC

R topics documented:

`balanced.cv.fold` *Split a dataset for Cross Validation taking into account class balance*

Description

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

---

balanced.loss.weights *Compute loss.weights so that total losses of each class is balanced*

Description

Compute loss.weights so that total losses of each class is balanced

Usage

```
balanced.loss.weights(y)
```

Arguments

y	a object coerced to factor that represent the class labels of each sample of the dataset

Value

a numeric vector of the same length as y

---

bhattacharyya.coefficient *Compute Bhattacharyya coefficient needed for Hellinger distance*

Description

Compute Bhattacharyya coefficient needed for Hellinger distance

Usage

```
bhattacharyya.coefficient(x)
```

Arguments

x	a numeric matrix

Value

a square matrix containing Bhattacharyya coefficient for each pair of row in x

Author(s)

Julien Prados

`binaryClassificationLoss` *Loss functions for binary classification*

Description

Loss functions for binary classification

Usage

```
logisticLoss(x, y, loss.weights = 1)

rocLoss(x, y)

fbetaLoss(x, y, beta = 1)

hingeLoss(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
`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.
`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

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

References

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

See Also

nrbm

Examples

```
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

```
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 paramters
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 identifing the first common ancestors of a and b

Author(s)

Julien Prados

Examples

```
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])
```

---

hellinger.dist *Compute Hellinger distance*

Description

Compute Hellinger distance

Usage

```
hellinger.dist(x)
```

Arguments

x	a numeric matrix

Value

an object of class "dist" with Hellinger distance of each pair of row in x

Author(s)

Julien Prados

is.convex *Return or set is.convex attribute*

Description

Return or set is.convex attribute

Usage

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

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

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

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

Arguments

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

Details

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

Value

attr(x,"is.convex")

iterative.hclust *Perform multiple hierachical clustering on random subsets of a dataset*

Description

Perform multiple hierachical clustering on random subsets of a dataset

Usage

```
iterative.hclust(x, seeds = 1:100, row.rate = 0.3, col.rate = 0.1,
  max.cluster = 10L, ret.height = FALSE, hc.method = function(x, PCs
  = 1:6, ...) {     hclust(dist(prcomp(x, rank. = max(PCs))$x[, PCs, drop =
  FALSE]), ...) }, ...)
```

Arguments

x	the numeric matrix containing the data to cluster (one instance per row)
seeds	a vector of random seed to use.
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).
ret.height	a logical to specify whether the average merging height should be returned.
hc.method	a clustering method of arity 1, taking as input a random subset of the input matrix x and returning an hclust object
...	additional arguments are passed to the hc.method

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 corresponding sum of heights for the pairs; K is the sum of the number of split possible that still preserve 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 incure 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

```
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

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

## S3 method for class 'svmLP'
predict(object, x, ...)

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

## S3 method for class 'svmMLP'
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

```
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

```
lvalue(x, ...)

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

lvalue(x, ...) <- value

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

Arguments

`x`	any R object
`...`	additional paramters
`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 trap 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()

Value

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 reclycled 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 specifyng 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

multivariateHingeLoss *The loss function for multivariate hinge loss*

Description

The loss function for multivariate hinge loss

Usage

```
multivariateHingeLoss(x, y, loss.weights = 1)
```

Arguments

x	matrix of training instances (one instance by row)
y	logical matrix of targets: y(t,) is the vector of binary labels for x(t,)
loss.weights	numeric vector of loss weights to incure for each instance of x. Vector length should match nrow(x), 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[1:4]))
y <- model.matrix(~iris$Species+0)>0
w <- nrbm(multivariateHingeLoss(x,y),LAMBDA=1)
table(y,predict(w,x)>0,col(y))
table(
  do.call(paste0,as.data.frame(y+0)),
  do.call(paste0,as.data.frame((predict(w,x)>0)+0))
)
```

---

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*LAMBDA*l2norm(w) + riskFun(w) where riskFun is either a convex or a non-convex risk function.

Usage

```
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)

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 controling stoping 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	mximal 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

```
# -- 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$Species=="setosa"),LAMBDA=1),
  svm_L2 = nrbm(hingeLoss(x,iris$Species=="setosa"),LAMBDA=1),
```

```
  f1_L1 = nrbmL1(fbetaLoss(x,iris$Species=="setosa"),LAMBDA=1),
  tsvm_L2 = nrbm(hingeLoss(x,
                ifelse(iris$Species=="versicolor",NA,iris$Species=="setosa")),
                LAMBDA=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]*1000/w[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*

Description

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

Usage

```
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

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

Examples

```
 # -- 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

```
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

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

# -- Train the model
w <- nrbm(ordinalRegressionLoss(x,y),LAMBDA=0.001,EPSILON_TOL=0.0001)
w2 <- nrbm(ordinalRegressionLoss(x,y,impl="quadratic"),LAMBDA=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$i] - f[ij$j]>0,list(y[ij$i],y[ij$j]),sum)
N <- table(y[ij$i],y[ij$j])
print(n/N)
```

`predict.mmc` *Predict class of new instances according to a mmc model*

Description

Predict class of new instances according to a mmc model

Usage

```
## 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.

preferenceLoss *The loss function for Preference loss*

Description

The loss function for Preference loss

Usage

```
preferenceLoss(x, P)
```

Arguments

x	matrix of training instances (one instance by row)
P	a data.frame with 3 fields (i,j,cost) that specify the cost for prefering sample j over sample i.

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

```
x <- data.matrix(iris[1:4])
P <- expand.grid(i=which(iris$Species=="virginica"),j=which(iris$Species!="virginica"))
w <- nrbm(preferenceLoss(x,P),LAMBDA=0.001,EPSILON_TOL=0.0001)
```

print.roc.stat *Generic method overlad to print object of class roc.stat*

Description

Generic method overlad to print object of class roc.stat

Usage

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

Arguments

x	a roc.stat object return by the function roc.stat
...	additional parameters

---

rank.linear.weights *Rank linear weight of a linear model*

Description

Rank linear weight of a linear model

Usage

```
rank.linear.weights(w)
```

Arguments

w	a numeric vector of linear weights

Value

a data.frame with a rank for each feature as well as z-score, p-value, and false discovery rate.

---

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

```
x <- cbind(data.matrix(iris[1:4]))
w <- nrbmL1(rocLoss(x,iris$Species=="versicolor"),LAMBDA=0.01)
plot(roc.stat(x %*% w,iris$Species=="versicolor"))
lines(roc.stat(-x[,2],iris$Species=="versicolor"),col="blue")
```

---

rowmean *Columun means of a matrix based on a grouping variable*

Description

Similar to rowsum, but for mean values.

Usage

```
rowmean(x, group, ...)
```

Arguments

x	a matrix
group	a factor with one element per row of x
...	additional arguments are passed to rowsum()

Value

a matrix containing the means, with one row per level of group.

---

softMarginVectorLoss *Soft Margin Vector Loss function for multiclass SVM*

Description

Soft Margin Vector Loss function for multiclass SVM

Usage

```
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,). If it contains NAs, the return function is a non-convex loss for transductive multiclass-SVM.
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

```
# -- Build a 2D dataset from iris, and add an intercept
x <- cbind(intercept=100,data.matrix(iris[c(1,2)]))
y <- iris$Species

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

# -- Plot the dataset, the decision boundaries, the convergence curve, and the predictions
gx <- seq(min(x[,2]),max(x[,2]),length=200) # positions of the probes on x-axis
gy <- seq(min(x[,3]),max(x[,3]),length=200) # positions of the probes on y-axis
Y <- outer(gx,gy,function(a,b) {predict(w,cbind(100,a,b))})
image(gx,gy,unclass(Y),asp=1,main="dataset & decision boundaries",
      xlab=colnames(x)[2],ylab=colnames(x)[3])
points(x[,-1],pch=19+as.integer(y))
```

---

softmaxLoss *softmax Loss Function*

Description

softmax loss function may be used to predict probability distributions

Usage

```
softmaxLoss(x, y, loss.weights = 1)
```

Arguments

x	instance matrix, where x(t,) defines the features of instance t
y	target matrix where y(t,) is a probability distribution that should sum to 1
loss.weights	numeric vector of loss weights to incure for each instance of x. Vector length should match nrow(y), but values are recycled if not of identical size.

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

Examples

```
# -- Load the data
x <- cbind(intercept=100,data.matrix(iris[1:4]))
y <- model.matrix(~iris$Species+0)
w <- nrbm(softmaxLoss(x,y))
P <- predict(w,x)
table(max.col(P),iris$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 bmrm/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

Examples

```
fun <- function(w) {
  gradient(w) <- w
  lvalue(w) <- 0.5*sum(w*w)
  w
}
wolfe.linesearch(fun,fun(c(5,5)),c(-1,-1))
wolfe.linesearch(fun,fun(c(5,5)),c(1,1))
```

Index