Package 'Rgtsvm'

June 16, 2017

Version 0.4

Date 2017-06-15

Title The e1071 compatiblility SVM package for GPU architecture based on the GTSVM software

Imports graphics, grDevices, bit64, tools, methods, stats, utils, class

Suggests SparseM, Matrix

Description The e1071 compatiblility SVM package is built on the CUDA-enable GPU architecture with the following features, including: binary classification, multiclass classification and epsilon regression; 4 kernel functions; K-fold cross validation; big matrix handle.

License GPL(>=3)

LazyLoad yes

Author Zhong Wang <zw355@cornell.edu>

Maintainer Zhong Wang <zw355@cornell.edu>

NeedsCompilation yes

Repository CRAN

R topics documented:

attach.bigmatrix *Wrapping a big matrix into a reference class*

Description

This function wraps a big matrix into a reference class in order to avoid multiple variable copying when variables are passed into deep calls in R. It seems to become pointer calling in C language.

Usage

```
attach.bigmatrix(data)
```

Arguments

data	matrix object

Value

Return a reference class with the name "BigMatrix.refer". It can be used in svm and predict calling in Rgtsvm.

See Also

load.bigmatrix

Examples

```
library(mvtnorm);
size=5000;
dimension=100;

covar.mat <- matrix(runif(dimension*dimension),nrow=dimension);
covar.mat <- t(covar.mat)
covar.mat <- round( (covar.mat + t(covar.mat))/2, 4);

zero <- rmvnorm(size, mean=c(1:dimension), sigma= covar.mat);
one  <- rmvnorm(size, mean=c(1:dimension)-5, sigma= covar.mat);
x <- rbind(zero,one);
y <- c(rep(0,nrow(zero)),rep(1,nrow(one)));

i.all <- sample(1:(2*size));
i.training <- i.all[(1:round(2*size*0.8))];
i.test <- i.all[-c(1:round(2*size*0.8))];

bigm.x <- attach.bigmatrix( data = x[ i.training,]);
model.gpu <- svm(bigm.x,y[ i.training ],type="C-classification");

y.pred <- predict(model.gpu,x[i.test,]);
cat("accuracy", sum(y.pred==y[i.test])/length(i.test),"\n");
```

load.bigmatrix *Loading a big matrix from RData or RDS file*

Description

Creating a big matrix based on the matrix variable in a RData file or a RDS file.

Usage

```
load.bigmatrix(file.data, variable = NULL)
```

Arguments

file.data	File name, RData file or RDS file.
variable	String, variable name in the RData file. If variable is NULL, the data file should be in RDS format.

Value

Return a reference class with the name "BigMatrix.refer". It can be used in svm and predict calling in Rgtsvm.

See Also

attach.bigmatrix

Examples

```
# The example can not be executed!
#
#  x0_bm <- load.bigmatrix("X0.RDS")
#  x1_bm <- load.bigmatrix("X1.Rdata", "x1")
```

load.svmlight *Load SVMlight data file into a sparse matrix.*

Description

Load SVMlight data file into a sparse matrix.

Usage

```
load.svmlight(filename, .loadbyC = TRUE)
```

Arguments

filename	string, SVM light filename.
.loadbyC	logical value, indicating whether loading data in C or R.

Details

The file must be svmlight format.(http://svmlight.joachims.org/)

Value

A sparse matrix is returned if the file is loaded or downloaded successfully.

Author(s)

Zhong Wang (R interface) <zw355@cornell.edu>

Examples

```
mat <-load.svmlight("http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/glass.scale")
str(mat);
```

---

plot.gtsvm *Scatter plot for classification models*

Description

A scatter figure is generated for the classification model trained by the svm. The figure only shows two feature columns within training data. Optionally, a filled contour is predicted and added to the class regions.

Usage

```
## S3 method for class 'gtsvm'
plot(x, data, formula,
      fill = TRUE, grid = 50, slice = list(),
      symbolPalette = palette(),
      svSymbol = "x", dataSymbol = "o", ...)
```

Arguments

x	an object of class gtsvm generated by svm.
data	data frame with the training data.
formula	formula object indicating two feature columns used as the X axis and Y axis if more than two feature columns in the training data.
fill	logical value indicating whether a contour plot for the class regions is added.
grid	integer value indicating the grid number to generating the the contour if fill = TRUE.
slice	a list of feature columns that are constant during the predicting for the grid contour if training data has more than two feature columns. all feature columns but two feature columns used in the X axis and Y axis must be held constant to generate a grid contour prediction.. In order to generate a contour, except two feature columns of X axis and Y axis, other feature columns hold constant values for predicting. Default unspecified feature columns use 0 for numeric variables and the first level for factors.

symbolPalette	Color palette indicating the color of the class data points and support vectors belong to.
svSymbol	character or integer code indicating plot character or symbol for support vectors, check the available values in pch in points function.
dataSymbol	character or integer code indicating plot character or symbol for non-support vectors, check the available values in pch in points function.
...	additional graphics parameters passed to filled.contour or plot.

Author(s)

David Meyer
<David.Meyer@R-project.org>

See Also

svm

Examples

```
## a simple example
library(MASS);
data(cats, package = "MASS")
m <- svm(Sex~., data = cats)
plot(m, cats)

## more than two variables: fix 2 dimensions
data(iris)
m2 <- svm(Species~., data = iris)
plot(m2, iris, Petal.Width ~ Petal.Length,
     slice = list(Sepal.Width = 3, Sepal.Length = 4))

## plot with custom symbols and colors
plot(m, cats, svSymbol = 1, dataSymbol = 2, symbolPalette = rainbow(4),
color.palette = terrain.colors)
```

---

plot.tune *Plot performance of tuning object*

Description

Contour or perspective plot to visualize the performace of tuning results. It is limited to two tuning parameters.

Usage

```
## S3 method for class 'tune'
plot(x, type = c("contour", "perspective"), theta = 60,
          col = "lightblue", main = NULL, xlab = NULL, ylab = NULL,
          swapxy = FALSE, transform.x = NULL, transform.y = NULL,
          transform.z = NULL, color.palette = hsv_palette(),
          nlevels = 20, ...)
```

Arguments

x	an object of class tune generated by the tune.svm
type	string indicating whether a contour plot or a perspective plot is used if two parameters shall be plotted. Ignored if only one tuning parameter in the x.
theta	numeric value indicating angle of azimuthal direction for perspective plot.
col	the color(s) of the surface facets for the perspective plot. Ignored if transparent colors are specified.
main	string indicating main title.
xlab	string or expression indicating titles for the X axes.
ylab	string or expression indicating titles for the Y axes.
swapxy	logical value indicating whether the parameter axes are swapped, only used in case of two parameters.
transform.x	function indicating how to transform the first parameter in X axis. Ignored if NULL.
transform.y	function indicating how to transform the second parameter in Y axis. Ignored if NULL.
transform.z	function indicating how to transform the error measures in Z axis in the perspective plot or color palette in the contour plot. Ignored if NULL.
color.palette	color palette used in the contour plot.
nlevels	integer value indicating number of levels used in the contour plot.
...	additional graphics parameters passed to filled.contour or persp.

Author(s)

David Meyer (based on C/C++-code by Chih-Chung Chang and Chih-Jen Lin)
<David.Meyer@R-project.org>

See Also

tune.svm

Examples

```
data(iris)
obj <- tune.svm(Species~., data = iris, sampling = "fix",
                gamma = 2^c(-8,-4,0,4), cost = 2^c(-8,-4,-2,0))
plot(obj, transform.x = log2, transform.y = log2)
plot(obj, type = "perspective", theta = 120, phi = 45)
```

predict.gtsvm *Predict method for SVMs on CUDA-enabled GPU*

Description

This function performs prediction based on a SVM model trained by svm in package *Rgtsvm*.

Usage

```
## S3 method for class 'gtsvm'
predict(object, newdata,
      decision.values = FALSE,
      probability = FALSE,
      verbose=FALSE,
      ...,
      na.action = na.omit)
```

Arguments

object	an object of class "gtsvm" returned by svm in *Rgtsvm* package.
newdata	data frame, or matrix, or sparse matrix of the test data. A vector test data must transform to a n x 1 matrix.
decision.values	logical value indicating whether the decision values of binary classfication or multiclass classification shall be returned. Only valid for classfication.
probability	logical value indicating whether class probabilities should be computed and returned. **Not supported in Rgtsvm currently**.
verbose	logical value indicating whether some alogrithm information is output into the R console, default is FALSE.
...	Unused currently.
na.action	a function to specify the action to be taken if NAs are found. The default action is na.omit, which leads to rejection of cases with missing values on any required variable. An alternative is na.fail, which causes an error if NA cases are found. (NOTE: If given, this argument must be named.

Value

A vector of predicted values or labels are returned.

If decision.values is required, the vector has a "decision.values" attribute containing a decision matrix with number of samples in rows and number of classes and in columns.

Note

The test data can be an object containing the new input data: data frame, or matrix, or a sparse matrix. The sparse matrix can be defined by the class Matrix provided by the **Matrix** package, or the class matrix.csr provided by the **SparseM** package, or the class simple_triplet_matrix provided by the **slam** package)

If the training data was scaled by the svm calling, this function shall scale the test data accordingly using scale and center of the training data.

Author(s)

Zhong Wang (R interface & eps-regression in CUDA) <zw355@cornell.edu>
David Meyer (R interface in e1071) <David.Meyer@R-project.org>
Andrew Cotter, Nathan Srebro ,Joseph Keshet (C/C++ code in CUDA)
http://ttic.uchicago.edu/~cotter/projects/gtsvm/

See Also

svm

---

svm *Training a model of SVMs on CUDA-enabled GPU*

Description

svm in the **Rgtsvm** pakcage is used to train a support vector machine by C-classfication or epsilon regression on CUDA-enabled GPU.

Usage

```
## S3 method for class 'formula'
svm(formula, data = NULL, ..., subset, na.action = na.omit, scale = TRUE);

## Default S3 method:
svm(x,
      y           = NULL,
      scale       = TRUE,
      type        = "C-classification",
      kernel      = "radial",
      degree      = 3,
      gamma       = if (is.vector(x)) 1 else 1/ncol(x),
      coef0       = 0,
      cost        = 1,
      class.weights= NULL,
      tolerance   = 0.001,
      epsilon     = 0.1,
      shrinking   = TRUE,
      cross       = 0,
      probability = FALSE,
      fitted      = TRUE,
      rough.cross = 0,
      no.change.x = TRUE,
      verbose     = FALSE,
      ...,
      subset,
      na.action = na.omit)
```

Arguments

formula	a formula object describing the training model.
data	an optional data frame containing the variables in the model. By default the variables are taken from the environment which 'svm' is called from.
x	a data matrix, or a vector, or a sparse matrix as training data.
y	a factor (for C-classification) or a numeric vector (for eps-regression) specifying the response for each row of x.
scale	logical value or a logical vector, indicating whether the feature columns are scaled. By default, both x and y variables are scaled to zero mean and unit variance. The center and scale values are returned and can be used for scaling new test data.
type	string indicating the SVM method, only C-classification or eps-regression available, default is C-classification
kernel	the kernel function used in training and predicting, four options are available (attached in details), default is radial.
degree	integer value indicating parameter value used in kernel of type polynomial, default is 3.
gamma	numerical value indicating parameter value needed for all kernels except linear, default is 1/(number of feature vector)).
coef0	numerical value indicating parameter value used in kernels of type polynomial and sigmoid, default is 0.
cost	numerical value indicating regularization term in the Lagrange formulation, which is cost of constraints violation, default is 1 .
class.weights	a named vector of weights for the different classes, used for asymmetric class sizes. Not all factor levels have to be supplied (default weight: 1). All components have to be named.
tolerance	numerical value indicating the tolerance of termination criterion for the training algorithm, default is 0.001.
epsilon	numerical value indicating epsilon in the insensitive-loss function for the eps-regression method, default is 0.1.
shrinking	logical value indicating whether to use the shrinking-heuristics, default is TRUE.
cross	integer value indicating whether a k-fold cross validation on the training data is performed to assess the quality of the model. Ignored if corss=0.
rough.cross	integer value which is less than cross, indicating how many tests are performed for cross-validation. The function will return partial tests for cross-validation rather than all repeated tests in order to reduce the running time.
fitted	logical value indicating whether the prediction should be performed and returned in the function calling, default is TRUE.
probability	logical value indicating whether the model should allow for probability predictions, not supported in **Rgtsvm**.
no.change.x	logical value indicating whther the function can change the x parameter. It would save CPU memory if this parameter is assigned to FALSE for the big matrix x.
verbose	logical value indicating whether some alogrithm information is output on R console, default is FALSE.
...	additional parameters for the low level fitting function svm.default

subset	a vector of index values specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)
na.action	a function to specify the action to be taken if NAs are found. The default action is na.omit, which leads to rejection of cases with missing values on any required variable. An alternative is na.fail, which causes an error if NA cases are found. (NOTE: If given, this argument must be named.)

Details

Rgtsvm uses a sparse matrix and regular matrix. The sparse matrix can be defined by the class Matrix provided by the **Matrix** package, or the class matrix.csr provided by the **SparseM** package, or the class simple_triplet_matrix provided by the **slam** package)

The kernel function has the following parameters, depending on the kernel type.

linear: $u'v$

polynomial: $(\gamma u'v + coef0)^{degree}$

radial basis: $e^(-\gamma|u-v|^2)$

sigmoid: $tanh(\gamma u'v + coef0)$

plot.gtsvm provides a simple visualization method for the classification model trained by svm.

Value

This function returns a trained model. It is an object of class "gtsvm" with following properties:

total.nSV	the total number of support vectors.
nSV	the number of support vectors in each group.
SV	the resulting support vectors (possibly scaled).
index	the index of the resulting support vectors in the data matrix. Note that this index refers to the preprocessed data (after the possible effect of na.omit and subset)
coefs	the corresponding coefficients multiply the training labels.
rho	the negative intercept.
fitted	the prediction values if fitted=TRUE
fitted.accuracy	The predicted accuracy for C-classification if fitted=TRUE.
fitted.MSE	the mean square error for eps-regression if fitted=TRUE.
fitted.r2	the R square for eps-regression if fitted=TRUE.
residuals	the difference between the true values and prediction for eps-regression if fitted=TRUE.
MSE	the mean square errors at each cross-validation test for eps-regression if cross>0.
tot.MSE	the mean square error in cross-validation for eps-regression if cross>0.
scorrcoeff	the R square in cross-validation for eps-regression if cross>0.
accuracies	the accuracies at each cross-validation test for C-classification if cross>0.
tot.accuracy	the total accuracies in cross-validation for C-classification if cross>0.

Author(s)

Zhong Wang (R interface & epe-regression in CUDA) <zw355@cornell.edu>
David Meyer (R interface in e1071) <David.Meyer@R-project.org>
Andrew Cotter, Nathan Srebro ,Joseph Keshet (C/C++ code in CUDA)
http://ttic.uchicago.edu/~cotter/projects/gtsvm/

References

- Andrew Cotter, Nathan Srebro, Joseph Keshet. "A GPU-Tailored Approach for Training Kernelized SVMs". 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2011.
- Chang, Chih-Chung and Lin, Chih-Jen:
LIBSVM: a library for Support Vector Machines
http://www.csie.ntu.edu.tw/~cjlin/libsvm

See Also

predict.gtsvm plot.gtsvm matrix.csr (in package **SparseM**)

Examples

```
data(iris)
attach(iris)

## classification mode
# default with factor response:
model <- svm(Species ~ ., data = iris)

# alternatively the traditional interface:
x <- subset(iris, select = -Species)
y <- Species
model <- svm(x, y)

print(model)
summary(model)

# test with train data
pred <- predict(model, x)

# Check accuracy:
table(pred, y)

# compute decision values and probabilities:
pred <- predict(model, x, decision.values = TRUE)
attr(pred, "decision.values")[1:4,]

# visualize (classes by color, SV by crosses):
plot(cmdscale(dist(iris[,-5])),
     col = as.integer(iris[,5]),
     pch = c("o","+")[1:150 %in% model$index + 1])
```

tune.control *Control parameters for the tune function*

Description

Creates an object of class `tune.control` to be used with the `tune.svm` function, containing various control parameters.

Usage

```
tune.control(random = FALSE,
      nrepeat = 1,
      repeat.aggregate = mean,
      sampling = c("cross", "fix", "bootstrap"),
      sampling.aggregate = mean,
      sampling.dispersion = sd,
      cross = 10,
      fix = 2/3,
      nboot = 10,
      boot.size = 9/10,
      best.model = TRUE,
      performances = TRUE,
      rough.cross = 0,
      error.fun = NULL)
```

Arguments

`random`	`FALSE` or an integer value, indicating whether or how many parameter combinations are drawn from the parameter space.
`nrepeat`	integer value, specifies how many replication of training and prediction shall be repeated, default is 1.
`repeat.aggregate`	function, used to aggregate the replicates results , default is `mean`.
`sampling`	string value indicating sampling scheme, three sampling methods are available, default is `cross`.
`sampling.aggregate`	function to aggregate the test results for each parameter combination, default is `mean`.
`sampling.dispersion`	function to disperse the test results for each parameter combination, default is `sd`.
`cross`	integer value, used if `sampling = "cross"`, indicating the number of partitions for cross-validation, default is 10.
`fix`	numeric value, used if `sampling = "fix"`, part of the data used for training in fixed sampling., default is 2/3
`nboot`	integer value, used if `sampling = "boot"`, indicating number of bootstrap replications.
`boot.size`	numeric value, used if `sampling = "boot"`, indicating size of the bootstrap samples.

best.model	logical value, indicating whether the best model is returned based on the the best parameter set on the complete training set).
performances	logical value, indicating whether the performance results for all parameter combinations are returned.
rough.cross	integer value if sampling = "cross", indicating how many training and testing for cross-validation are conducted, it is intended to reduce the computation time by decreasing the cross-validation jobs while maintaining the higher cross number.
error.fun	function for the error measure, It takes two arguments: a vector of true values and a vector of predicted values. If NULL, the misclassification error is used for categorical predictions and the mean squared error for numeric predictions.

Details

Three options for sampling are available:

1: sampling = "cross", a cross-times cross validation.

2: sampling = "boot", sampling with replacement on the nboot training sets of boot.size (part).

3: sampling = "fix", the dataset is split into a training set of size fix and validation set.

Value

An object of class "tune.control" containing all the above parameters (either the defaults or the user specified values).

Author(s)

David Meyer
<David.Meyer@R-project.org>

See Also

tune.svm

---

tune.svm *Parameter tuning function using grid search*

Description

tune.svm() tunes hyperparameters of statistical methods using a grid search over supplied parameter ranges. best.tune() returns the best model detected by tune.svm.

Usage

```
tune.svm(x, y = NULL, data = NULL, degree = NULL, gamma = NULL,
      coef0 = NULL, cost = NULL, nu = NULL,
      class.weights = NULL, epsilon = NULL,...)
best.tune(...)
```

Arguments

x	formula object or training matrix.
y	vector indicating the response variable only if train.x is not a formula object, otherwise ignore.
data	data frame only if the formula interface is used in train.x, otherwise ignore.
degree	a numeric vector indicating the tuning values for the parameter degree.
gamma	a numeric vector indicating the tuning values for the parameter gamma.
coef0	an numeric vector indicating the tuning values for the parameter coef0.
cost	a numeric vector indicating the tuning values for the parameter cost.
nu	a numeric vector indicating the tuning values for the parameter nu.
class.weights	a numeric vector indicating the tuning values for the parameter class.weights.
epsilon	a numeric vector indicating the tuning values for the parameter epsilon.
...	Further parameters passed to the training functions, i.e. svm, tune.

Details

To measure performance, classification error and mean squared error can be used for the classificaition and epsilon regression respectively.

Value

This function return an object of class tune, including the components:

best.parameters	a 1 x k data frame, k number of parameters.
best.performance	value, best achieved performance.
performances	if requested, a data frame of all parameter combinations along with the corresponding performance results.
train.ind	list of index vectors used for splits into training and validation sets.
best.model	if requested, the model trained on the complete training data using the best parameter combination.

Author(s)

David Meyer
<David.Meyer@R-project.org>

See Also

tune.control, plot.tune

Examples

```
data(iris)

## tune `svm' for classification with RBF-kernel (default in svm),
## using one split for training/validation set
obj <- tune.svm(Species~., data = iris, gamma = 2^(-1:1), cost = 2^(2:4))

summary(obj)
plot(obj)
```

Index