Package 'Rgtsvm'

June 10, 2017

Version 0.4

Date 2017-04-25

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

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

Suggests SparseM, Matrix

Description Training and prediction of SVM based on GTSVM are avalaible on GPU architecture.

License GPLv3

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 wrapps a big matrix into a reference class in order to avoid multiple variable coping 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(MASS);
size=5000;
dimension=100;
covar.mat <- matrix(runif(dimension*dimension),nrow=dimension);
covar.mat <- t(covar.mat)
zero <- mvrnorm(size,mu=c(1:dimension),Sigma= covar.mat);
one  <- mvrnorm(size,mu=c(1:dimension)-5,Sigma= covar.mat);
x <- rbind(zero,one);
y <- c(rep(0,nrow(zero)),rep(1,nrow(one)));

i.all <- 1:(2*size);
i.training <- sample(i.all, length(i.all)*0.8);
i.test <- i.all [! i.all

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 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)
```

Arguments

filename	SVM light filename.

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` *Plot SVM Objects*

Description

Generates a scatter plot of the input data of a `svm` fit for classification models by highlighting the classes and support vectors. Optionally, draws a filled contour plot of the class regions.

Usage

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

Arguments

`x`	An object of class `svm`
`data`	data to visualize. Should be the same used for fitting.
`formula`	formula selecting the visualized two dimensions. Only needed if more than two input variables are used.
`fill`	switch indicating whether a contour plot for the class regions should be added.
`grid`	granularity for the contour plot.
`slice`	a list of named values for the dimensions held constant (only needed if more than two variables are used). The defaults for unspecified dimensions are 0 (for numeric variables) and the first level (for factors). Factor levels can either be specified as factors or character vectors of length 1.
`symbolPalette`	Color palette used for the class the data points and support vectors belong to.
`svSymbol`	Symbol used for support vectors.
`dataSymbol`	Symbol used for data points (other than support vectors).
`...`	additional graphics parameters passed to `filled.contour` and `plot`.

Author(s)

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

See Also

`svm`

Examples

```
## a simple example
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 Tuning Object*

Description

Visualizes the results of parameter tuning.

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`
`type`	choose whether a contour plot or a perspective plot is used if two parameters are to be visualized. Ignored if only one parameter has been tuned.
`theta`	angle of azimuthal direction.
`col`	the color(s) of the surface facets. Transparent colors are ignored.
`main`	main title
`xlab, ylab`	titles for the axes. N.B. These must be character strings; expressions are not accepted. Numbers will be coerced to character strings.
`swapxy`	if `TRUE`, the parameter axes are swaped (only used in case of two parameters).
`transform.x, transform.y, transform.z`	functions to transform the parameters (`x` and `y`) and the error measures (`z`). Ignored if `NULL`.

color.palette	color palette used in contour plot.
nlevels	number of levels used in contour plot.
...	Further graphics parameters.

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

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 Support Vector Machines*

Description

This function predicts values based upon a 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	Object of class "gtsvm", created by svm in *Rgtsvm* package.
newdata	An object containing the new input data: either a matrix or a sparse matrix (object of class Matrix provided by the **Matrix** package, or of class matrix.csr provided by the **SparseM** package, or of class simple_triplet_matrix provided by the **slam** package). A vector will be transformed to a n x 1 matrix.
decision.values	Logical controlling whether the decision values of all binary classifiers computed in multiclass classification shall be computed and returned.
probability	Logical indicating whether class probabilities should be computed and returned. Only possible if the model was fitted with the probability option enabled.

verbose	logical value indicating whether some alogrithm information(default:FALSE)
...	Currently not used.
na.action	A function to specify the action to be taken if 'NA's 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 (for classification: a vector of labels, for density estimation: a logical vector). If score is TRUE, the vector gets a "decision.values" attribute containing a n x c matrix (n number of predicted values, c number of classifiers) of all c binary classifiers' decision values.

Note

If the training set was scaled by svm in *Rgtsvm*, the new data is scaled accordingly using scale and center of the training data.

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/\textasciitildecotter/proje

See Also

svm

---

svm *Training a model of Support Vector Machines by GPU*

Description

svm in *Rgtsvm* pakcage is used to train a support vector machine by the C-classfication and epsilon regression method. A formula interface is provided.

Usage

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

## Default S3 method:
svm(x,
      y           = NULL,
      scale       = TRUE,
      type        = "C-classification",
      kernel      = "radial",
      degree      = 3,
      gamma       = 0.05,
      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 symbolic description of the model to be fit.
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, a vector, or a sparse matrix (object of class Matrix provided by the **Matrix** package, or of class matrix.csr provided by the **SparseM** package, or of class simple_triplet_matrix provided by the **slam** package).
y	a response vector with one label for each row/component of x. Can be either a factor (for classification tasks) or a numeric vector (for regression).
scale	A logical vector indicating the variables to be scaled. If scale is of length 1, the value is recycled as many times as needed. Per default, data are scaled internally (both x and y variables) to zero mean and unit variance. The center and scale values are returned and used for later predictions.
type	only C-classification or eps-regression available.
kernel	the kernel used in training and predicting. You might consider changing some of the following parameters, depending on the kernel type. **linear:** $u'v$ **polynomial:** $(\gamma u'v + coef0)^{degree}$ **radial basis:** $e^(- \gamma \lvert u - v \rvert^2)$ **sigmoid:** $tanh(\gamma u'v + coef0)$
degree	parameter needed for kernel of type polynomial (default: 3)
gamma	parameter needed for all kernels except linear (default: 1/(data dimension))
coef0	parameter needed for kernels of type polynomial and sigmoid (default: 0)
cost	cost of constraints violation (default: 1)—it is the 'C'-constant of the regularization term in the Lagrange formulation.
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	tolerance of termination criterion (default: 0.001)
epsilon	epsilon in the insensitive-loss function (default: 0.1)
shrinking	option whether to use the shrinking-heuristics (default: TRUE)

cross	if a integer value k>0 is specified, a k-fold cross validation on the training data is performed to assess the quality of the model: the accuracy rate for classification and the Mean Squared Error for regression
fitted	logical indicating whether the fitted values should be computed and included in the model or not (default: TRUE)
probability	logical indicating whether the model should allow for probability predictions.
rough.cross	Number 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.
no.change.x	Logical value indicating whther tne function can change the x parameter. If x is a big matrix, it would be save emeory to use this parameter FALSE.
verbose	logical value indicating whether some alogrithm information(default:FALSE)
...	additional parameters for the low level fitting function svm.default
subset	An index vector 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 internally uses a sparse matrix and regular matrix.

If the predictor variables include factors, the formula interface must be used to get a correct model matrix.

plot.gtsvm allows a simple graphical visualization of classification models.

Value

An object of class "gtsvm" containing the fitted model, including:

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 times the training labels.

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/\textasciitildecotter/proje

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)
# (same as:)
pred <- fitted(model)

# 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
```

---

tune *Parameter Tuning of Functions Using Grid Search*

Description

This generic function tunes hyperparameters of statistical methods using a grid search over supplied parameter ranges.

Usage

```
tune.svm(method, train.x, train.y = NULL, data = list(), validation.x =
     NULL, validation.y = NULL, ranges = NULL, predict.func = predict,
     tunecontrol = tune.control(), ...)
best.tune(...)
```

Arguments

method	either the function to be tuned, or a character string naming such a function.
train.x	either a formula or a matrix of predictors.
train.y	the response variable if train.x is a predictor matrix. Ignored if train.x is a formula.
data	data, if a formula interface is used. Ignored, if predictor matrix and response are supplied directly.
validation.x	an optional validation set. Depending on whether a formula interface is used or not, the response can be included in validation.x or separately specified using validation.y.
validation.y	if no formula interface is used, the response of the (optional) validation set.
ranges	a named list of parameter vectors spanning the sampling space. The vectors will usually be created by seq.
predict.func	optional predict function, if the standard predict behavior is inadequate.
tunecontrol	object of class "tune.control", as created by the function tune.control(). If omitted, tune.control() gives the defaults.
...	Further parameters passed to the training functions.

Details

As performance measure, the classification error is used for classification, and the mean squared error for regression. It is possible to specify only one parameter combination (i.e., vectors of length 1) to obtain an error estimation of the specified type (bootstrap, cross-classification, etc.) on the given data set. For convenience, there are several tune.foo() wrappers defined, e.g., for nnet(), randomForest(), rpart(), svm(), and knn().

Cross-validation randomizes the data set before building the splits which—once created—remain constant during the training process. The splits can be recovered through the train.ind component of the returned object.

Value

For tune, an object of class tune, including the components:

best.parameters	a 1 x k data frame, k number of parameters.
best.performance	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.

best.tune() returns the best model detected by tune.

Author(s)

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

See Also

tune.control, plot.tune, tune.svm, tune.wrapper

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,
            ranges = list(gamma = 2^(-1:1), cost = 2^(2:4)),
            tunecontrol = tune.control(sampling = "fix")
           )

## alternatively:
## obj <- tune.svm(Species~., data = iris, gamma = 2^(-1:1), cost = 2^(2:4))

summary(obj)
plot(obj)

## tune `knn' using a convenience function; this time with the
## conventional interface and bootstrap sampling:
x <- iris[,-5]
y <- iris[,5]
obj2 <- tune.knn(x, y, k = 1:5, tunecontrol = tune.control(sampling = "boot"))
summary(obj2)
plot(obj2)

## tune `rpart' for regression, using 10-fold cross validation (default)
data(mtcars)
obj3 <- tune.rpart(mpg~., data = mtcars, minsplit = c(5,10,15))
summary(obj3)
plot(obj3)

## simple error estimation for lm using 10-fold cross validation
tune(lm, mpg~., data = mtcars)
```

---

tune.control *Control Parameters for the Tune Function*

Description

Creates an object of class tune.control to be used with the tune 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	if an integer value is specified, random parameter vectors are drawn from the parameter space.
nrepeat	specifies how often training shall be repeated.
repeat.aggregate	function for aggregating the repeated training results.
sampling	sampling scheme. If sampling = "cross", a cross-times cross validation is performed. If sampling = "boot", nboot training sets of size boot.size (part) are sampled (with replacement) from the supplied data. If sampling = "fix", a single split into training/validation set is used, the training set containing a fix part of the supplied data. Note that a separate validation set can be supplied via validation.x and validation.y. It is only used for sampling = "boot" and sampling = "fix"; in the latter case, fix is set to 1.
sampling.aggregate,sampling.dispersion	functions for aggregating the training results on the generated training samples (default: mean and standard deviation).
cross	number of partitions for cross-validation.
fix	part of the data used for training in fixed sampling.
nboot	number of bootstrap replications.
boot.size	size of the bootstrap samples.
best.model	if TRUE, the best model is trained and returned (the best parameter set is used for training on the complete training set).
performances	if TRUE, the performance results for all parameter combinations are returned.
rough.cross	integer.
error.fun	function returning the error measure to be minimized. 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.

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

Index