Package 'SwarmSVM'

January 25, 2019

Title Ensemble Learning Algorithms Based on Support Vector Machines

Version 0.1-5

Date 2019-01-25

Author Tong He <hetong007@gmail.com>, Aydin Demircioglu <aydin.demircioglu@ini.ruhr-uni-bochum.de>

Maintainer Tong He <hetong007@gmail.com>

Description Three ensemble learning algorithms based on support vector machines. They all train support vector machines on subset of data and combine the result.

Depends R (>= 3.2.0)

Imports e1071, LiblineaR, Matrix, SparseM, kernlab, methods, checkmate (>= 1.6.0), BBmisc

License GPL-2

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-01-25 21:40:03 UTC

R topics documented:

`alphasvm` *Support Vector Machines taking initial alpha values*

Description

alphasvm is used to train a support vector machine. It can be used to carry out general regression and classification (of nu and epsilon-type), as well as density-estimation. A formula interface is provided.

Print alphasvm object

Summary alphasvm object

Print summary.alphasvm object

Usage

```
alphasvm(x, ...)

## S3 method for class 'formula'
alphasvm(formula, data = NULL, ..., subset,
  na.action = stats::na.omit, scale = FALSE)

## Default S3 method:
alphasvm(x, y = NULL, scale = FALSE, type = NULL,
  kernel = "radial", degree = 3, gamma = if (is.vector(x)) 1 else
  1/ncol(x), coef0 = 0, cost = 1, nu = 0.5, class.weights = NULL,
  cachesize = 40, tolerance = 0.001, epsilon = 0.1, shrinking = TRUE,
  cross = 0, probability = FALSE, fitted = TRUE, alpha = NULL,
  mute = TRUE, nclass = NULL, ..., subset, na.action = stats::na.omit)

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

## S3 method for class 'alphasvm'
summary(object, ...)
```

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

Arguments

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).
...	additional parameters for the low level fitting function svm.default
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.
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 stats::na.omit, which leads to rejection of cases with missing values on any required variable. An alternative is stats::na.fail, which causes an error if NA cases are found. (NOTE: If given, this argument must be named.)
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.
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).
type	svm can be used as a classification machine. The default setting for type is C-classification, but may be set to nu-classification as well.
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.
nu	parameter needed for nu-classification
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.
cachesize	cache memory in MB (default 40)
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
probability	logical indicating whether the model should allow for probability predictions.
fitted	logical indicating whether the fitted values should be computed and included in the model or not (default: TRUE)
alpha	Initial values for the coefficients (default: NULL). A numerical vector for binary classification or a nx(k-1) matrix for a k-class-classification problem.
mute	a logical value indicating whether to print training information from svm.
nclass	the number of classes in total.
object	An object of class alphasvm

Details

For multiclass-classification with k levels, k>2, libsvm uses the 'one-against-one'-approach, in which k(k-1)/2 binary classifiers are trained; the appropriate class is found by a voting scheme.

libsvm internally uses a sparse data representation, which is also high-level supported by the package **SparseM**.

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

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

The probability model for classification fits a logistic distribution using maximum likelihood to the decision values of all binary classifiers, and computes the a-posteriori class probabilities for the multi-class problem using quadratic optimization. The probabilistic regression model assumes (zero-mean) laplace-distributed errors for the predictions, and estimates the scale parameter using maximum likelihood.

Author(s)

Tong He (based on package e1071 by David Meyer and C/C++ code by Cho-Jui Hsieh in Divide-and-Conquer kernel SVM (DC-SVM))

References

- Chang, Chih-Chung and Lin, Chih-Jen:
LIBSVM: a library for Support Vector Machines
http://www.csie.ntu.edu.tw/~cjlin/libsvm
- Exact formulations of models, algorithms, etc. can be found in the document:
Chang, Chih-Chung and Lin, Chih-Jen:
LIBSVM: a library for Support Vector Machines
http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.ps.gz

- More implementation details and speed benchmarks can be found on: Rong-En Fan and Pai-Hsune Chen and Chih-Jen Lin:
  Working Set Selection Using the Second Order Information for Training SVM
  http://www.csie.ntu.edu.tw/~cjlin/papers/quadworkset.pdf

Examples

```
data(svmguide1)
svmguide1.t = svmguide1[[2]]
svmguide1 = svmguide1[[1]]

model = alphasvm(x = svmguide1[,-1], y = svmguide1[,1], scale = TRUE)
preds = predict(model, svmguide1.t[,-1])
table(preds, svmguide1.t[,1])

data(iris)
attach(iris)

# default with factor response:
model = alphasvm(Species ~ ., data = iris)

# get new alpha
new.alpha = matrix(0, nrow(iris),2)
new.alpha[model$index,] = model$coefs

model2 = alphasvm(Species ~ ., data = iris, alpha = new.alpha)
preds = predict(model2, as.matrix(iris[,-5]))
table(preds, iris[,5])
```

`cluster.fun.kkmeans` *Wrapper function for kernal kmeans*

Description

Wrapper function for kernal kmeans

Usage

```
cluster.fun.kkmeans(x, centers, ...)
```

Arguments

`x`	the input data
`centers`	the number of centers
`...`	other parameters passing to `kernlab::kkmeans`

`cluster.fun.mlpack.old`

Kmeans Clustering from RcppMLPACK

Description

The Kmeans algorithm from `RcppMLPACK`.

Usage

```
cluster.fun.mlpack.old(x, centers, ...)
```

Arguments

`x`	The input data for the clustering algorithm.
`centers`	A number indicating the number of clustering centers.
`...`	arguments for future use.

`cluster.predict.kkmeans`

Predict function for kernel kmeans

Description

Predict function for kernel kmeans

Usage

```
cluster.predict.kkmeans(x, cluster.object)
```

Arguments

`x`	The data to make prediction
`cluster.object`	The result object from `kernlab::kkmeans`

clusterSVM *Clustered Support Vector Machine*

Description

Implementation of Gu, Quanquan, and Jiawei Han. "Clustered support vector machines."

Usage

```
clusterSVM(x, y, centers = NULL, cluster.object = NULL, lambda = 1,
  sparse = TRUE, valid.x = NULL, valid.y = NULL, valid.metric = NULL,
  type = 1, cost = 1, epsilon = NULL, bias = TRUE, wi = NULL,
  verbose = 1, seed = NULL, cluster.method = "kmeans",
  cluster.fun = NULL, cluster.predict = NULL, ...)
```

Arguments

x	the nxp training data matrix. Could be a matrix or a sparse matrix object.
y	a response vector for prediction tasks with one value for each of the n rows of x. For classification, the values correspond to class labels and can be a 1xn matrix, a simple vector or a factor.
centers	an integer indicating the number of centers in clustering.
cluster.object	an object generated from cluster.fun, and can be passed to cluster.predict
lambda	the weight for the global l2-norm
sparse	indicating whether the transformation results in a sparse matrix or not
valid.x	the mxp validation data matrix.
valid.y	if provided, it will be used to calculate the validation score with valid.metric
valid.metric	the metric function for the validation result. By default it is the accuracy for classification. Customized metric is acceptable.
type	the type of the mission for LiblineaR.
cost	cost of constraints violation (default: 1). Rules the trade-off between regularization and correct classification on data. It can be seen as the inverse of a regularization constant. See details in LiblineaR.
epsilon	set tolerance of termination criterion for optimization. If NULL, the LIBLINEAR defaults are used, which are:
bias	if bias is TRUE (default), instances of data becomes [data; 1].
wi	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 according to the corresponding class label.
verbose	if set to 0, no information is printed. If set to 1 (default), the running time and validation score (if applicable) will be printed. If set to 2, the running time ,validation score (if applicable) and the LiblineaR information will be printed.

seed	the random seed. Set it to NULL to randomize the model.
cluster.method	The clusterign algorithm to use. Possible choices are • "kmeans" Algorithm from stats::kmeans • "mlKmeans" Algorithm from RcppMLPACK::mlKmeans • "kernkmeans" Algorithm from kernlab::kkmeans If cluster.fun and cluster.predict are provided, cluster.method doesn't work anymore.
cluster.fun	The function to train cluster labels for the data based on given number of centers. Customized function is acceptable, as long as the resulting list contains two fields named as cluster and centers.
cluster.predict	The function to predict cluster labels for the data based on trained object. Customized function is acceptable, as long as the resulting list contains two fields named as cluster and centers.
...	additional parameters passing to cluster.fun.

Value

- svm the svm object from LiblineaR
- lambda the parameter used.
- sparse whether the data is sparsely transformed
- label the clustering label for training data
- centers the clustering centers from teh training dataset
- cluster.fun the function used for clustering
- cluster.object the object either
- cluster.predict the function used for prediction on new data based on the object
- valid.pred the validation prediction
- valid.score the validation score
- valid.metric the validation metric
- time a list object recording the time consumption for each steps.

Examples

```
data(svmguide1)
svmguide1.t = svmguide1[[2]]
svmguide1 = svmguide1[[1]]

csvm.obj = clusterSVM(x = svmguide1[,-1], y = svmguide1[,1], lambda = 1,
                      centers = 8, seed = 512, verbose = 0,
                      valid.x = svmguide1.t[,-1],valid.y = svmguide1.t[,1])
csvm.pred = csvm.obj$valid.pred

# Or predict from the data
pred = predict(csvm.obj, svmguide1.t[,-1])
```

csvmTransform *Data Transformation function for Clustered Support Vector Machine*

Description

Transform a data matrix according to the kmeans clustering result based on Gu, Quanquan, and Jiawei Han. "Clustered support vector machines."

Usage

```
csvmTransform(x, lambda, cluster.label, sparse = TRUE)
```

Arguments

x	The data matrix, could be a matrix or dgCMatrix object.
lambda	The parameter from the algorithm
cluster.label	The clustering label starting from 1. Its length must equal to the number of rows in x
sparse	Logical argument indicating whether the output should be a sparse matrix or not

dcSVM *Divide-and-Conquer kernel SVM (DC-SVM)*

Description

Implementation of Divide-and-Conquer kernel SVM (DC-SVM) by Cho-Jui Hsieh, Si Si, and Inderjit S. Dhillon

Usage

```
dcSVM(x, y, k = 4, m, kernel = 3, max.levels, early = 0,
  final.training = FALSE, pre.scale = FALSE, seed = NULL,
  verbose = TRUE, valid.x = NULL, valid.y = NULL, valid.metric = NULL,
  cluster.method = "kmeans", cluster.fun = NULL, cluster.predict = NULL,
  ...)
```

Arguments

x	the nxp training data matrix. Could be a matrix or a sparse matrix object.
y	a response vector for prediction tasks with one value for each of the n rows of x. For classification, the values correspond to class labels and can be a 1xn matrix, a simple vector or a factor.
k	the number of sub-problems divided
m	the number of sample for kernel kmeans

kernel	the kernel type: 1 for linear, 2 for polynomial, 3 for gaussian
max.levels	the maximum number of level
early	whether use early prediction
final.training	whether train the svm over the entire data again. usually not needed.
pre.scale	either a logical value indicating whether to scale the data or not, or an integer vector specifying the columns. We don't scale data in SVM seperately.
seed	the random seed. Set it to NULL to randomize the model.
verbose	a logical value indicating whether to print information of training.
valid.x	the mxp validation data matrix.
valid.y	if provided, it will be used to calculate the validation score with valid.metric
valid.metric	the metric function for the validation result. By default it is the accuracy for classification. Customized metric is acceptable.
cluster.method	The clusterign algorithm to use. Possible choices are • "kmeans" Algorithm from stats::kmeans • "mlKmeans" Algorithm from RcppMLPACK::mlKmeans • "kernkmeans" Algorithm from kernlab::kkmeans If cluster.fun and cluster.predict are provided, cluster.method doesn't work anymore.
cluster.fun	The function to train cluster labels for the data based on given number of centers. Customized function is acceptable, as long as the resulting list contains two fields named as cluster and centers.
cluster.predict	The function to predict cluster labels for the data based on trained object. Customized function is acceptable, as long as the resulting list contains two fields named as cluster and centers.
...	other parameters passed to e1071::svm

Value

- svm a list of svm models if using early prediction, or an svm object otherwise.
- early whether using the early prediction strategy or not
- cluster.tree a matrix containing clustering labels in each level
- cluster.fun the clustering training function
- cluster.predict the clustering predicting function
- scale a list containing scaling information
- valid.pred the validation prediction
- valid.score the validation score
- valid.metric the validation metric
- time a list object recording the time consumption for each steps.

Examples

```
data(svmguide1)
svmguide1.t = as.matrix(svmguide1[[2]])
svmguide1 = as.matrix(svmguide1[[1]])
dcsvm.model = dcSVM(x = svmguide1[,-1], y = svmguide1[,1],
                    k = 4, max.levels = 4, seed = 0, cost = 32, gamma = 2,
                    kernel = 3,early = 0, m = 800,
                    valid.x = svmguide1.t[,-1], valid.y = svmguide1.t[,1])
preds = dcsvm.model$valid.pred
table(preds, svmguide1.t[,1])
dcsvm.model$valid.score
```

---

eucliDist *Euclidean Distance calculation*

Description

Euclidean Distance calculation

Usage

```
eucliDist(x, centers)
```

Arguments

x	the data matrix
centers	the matrix of centers

---

gater *Gater function for mixture SVMs*

Description

Gater function for mixture SVMs

Usage

```
gater(x, y, S, hidden, learningrate = 0.01, threshold = 0.01,
  stepmax = 100, verbose = verbose, ...)
```

Arguments

x	the nxp training data matrix. Could be a matrix or a sparse matrix object.
y	a response vector for prediction tasks with one value for each of the n rows of x. For classification, the values correspond to class labels and can be a 1xn matrix, a simple vector or a factor. For regression, the values correspond to the values to predict, and can be a 1xn matrix or a simple vector.
S	the prediction matrix from experts
hidden	the number of neurons in the hidden layer
learningrate	the learningrate for the back propagation
threshold	neural network stops training once all gradient is below the threshold
stepmax	the maximum iteration of the neural network training process
verbose	a logical value indicating whether to print information of training.
...	other parameters passing to neuralnet

---

gaterSVM *Mixture SVMs with gater function*

Description

Implementation of Collobert, R., Bengio, S., and Bengio, Y. "A parallel mixture of SVMs for very large scale problems. Neural computation".

Usage

```
gaterSVM(x, y, m, c = 1, max.iter, hidden = 5, learningrate = 0.01,
  threshold = 0.01, stepmax = 100, seed = NULL, valid.x = NULL,
  valid.y = NULL, valid.metric = NULL, verbose = FALSE, ...)
```

Arguments

x	the nxp training data matrix. Could be a matrix or an object that can be transformed into a matrix object.
y	a response vector for prediction tasks with one value for each of the n rows of x. For classification, the values correspond to class labels and can be a 1xn matrix, a simple vector or a factor. For regression, the values correspond to the values to predict, and can be a 1xn matrix or a simple vector.
m	the number of experts
c	a positive constant controlling the upper bound of the number of samples in each subset.
max.iter	the number of iterations
hidden	the number of neurons on the hidden layer
learningrate	the learningrate for the back propagation

threshold	neural network stops training once all gradient is below the threshold
stepmax	the maximum iteration of the neural network training process
seed	the random seed. Set it to NULL to randomize the model.
valid.x	the mxp validation data matrix.
valid.y	if provided, it will be used to calculate the validation score with valid.metric
valid.metric	the metric function for the validation result. By default it is the accuracy for classification. Customized metric is acceptable.
verbose	a logical value indicating whether to print information of training.
...	other parameters passing to neuralnet

Value

- expert a list of svm experts
- gater the trained neural network model
- valid.pred the validation prediction
- valid.score the validation score
- valid.metric the validation metric
- time a list object recording the time consumption for each steps.

Examples

```
data(svmguide1)
svmguide1.t = as.matrix(svmguide1[[2]])
svmguide1 = as.matrix(svmguide1[[1]])
gaterSVM.model = gaterSVM(x = svmguide1[,-1], y = svmguide1[,1], hidden = 10, seed = 0,
                     m = 10, max.iter = 1, learningrate = 0.01, threshold = 1, stepmax = 100,
                     valid.x = svmguide1.t[,-1], valid.y = svmguide1.t[,1], verbose = FALSE)
table(gaterSVM.model$valid.pred,svmguide1.t[,1])
gaterSVM.model$valid.score
```

---

kmeans.predict *Euclidean Distance based clustering prediction*

Description

Euclidean Distance based clustering prediction

Usage

```
kmeans.predict(x, cluster.object)
```

Arguments

x	the data matrix
cluster.object	the matrix of centers

plot.alphasvm *Plot alphasvm object*

Description

Plot alphasvm object

Usage

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

Arguments

x	An object of class alphasvm
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 graphics::filled.contour and plot.

predict.alphasvm *Prediction function for an alphasvm object*

Description

Prediction function for an alphasvm object

Usage

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

Arguments

object	the object trained from alphasvm
newdata	the test data set
decision.values	a logical variable indicating whether to output the decision values
probability	a logical variable indicating whether to output the classfication probability
...	currently not used
na.action	A function to specify the action to be taken if 'NA's are found. The default action is stats::na.omit, which leads to rejection of cases with missing values on any required variable. An alternative is stats::na.fail, which causes an error if NA cases are found. (NOTE: If given, this argument must be named.)

---

predict.clusterSVM *Predictions with Clustered Support Vector Machines*

Description

The function applies a model (classification) produced by the clusterSVM function to every row of a data matrix and returns the model predictions.

Usage

```
## S3 method for class 'clusterSVM'
predict(object, newdata = NULL, cluster.predict = NULL,
  ...)
```

Arguments

object	Object of class "clusterSVM", created by clusterSVM.
newdata	An n x p matrix containing the new input data. Could be a matrix or a sparse matrix object.
cluster.predict	a function predict new labels on newdata.
...	other parameters passing to predict.LiblineaR

predict.dcSVM *Predictions with Divide-Conquer Support Vector Machines*

Description

The function applies a model produced by the dcSVM function to every row of a data matrix and returns the model predictions.

Usage

```
## S3 method for class 'dcSVM'
predict(object, newdata, ...)
```

Arguments

object	Object of class "dcSVM", created by dcSVM.
newdata	An n x p matrix containing the new input data. Could be a matrix or a sparse matrix object.
...	other parameters passing to predict.svm

predict.gater *Predictions for Gater function*

Description

The function applies a model produced by the gaterSVM function to every row of a data matrix and returns the model predictions.

Usage

```
## S3 method for class 'gater'
predict(object, newdata, ...)
```

Arguments

object	Object of class "gater", created by gater.
newdata	An n x p matrix containing the new input data. Could be a matrix or a sparse matrix object.
...	parameters for future usage.

predict.gaterSVM *Prediction for Gater SVM*

Description

The function applies a model produced by the gaterSVM function to every row of a data matrix and returns the model predictions.

Usage

```
## S3 method for class 'gaterSVM'
predict(object, newdata, ...)
```

Arguments

object	An object of class gaterSVM
newdata	newdata An n x p matrix containing the new input data. Could be a matrix or a sparse matrix object.
...	parameters for future usage.

svmguide1 *svmguide1*

Description

An astroparticle application from Jan Conrad of Uppsala University, Sweden.

Usage

```
data(svmguide1)
```

Format

A list of two data objects svmguide1 and svmguide1.t. The first column is the target variable.

SwarmSVM *SwarmSVM: A Package for several Ensemble Support Vector Machine Models*

Description

The SwarmSVM package contains three differenct Ensemble SVM models, they all train SVM models on subset of data and combine the results together. The three models are from the following three papers:

Details

1. Gu, Q., & Han, J. (2013). Clustered support vector machines. In proceedings of the sixteenth international conference on artificial intelligence and statistics (pp. 307-315). 2. Hsieh, C. J., Si, S., & Dhillon, I. S. (2013). A divide-and-conquer solver for kernel support vector machines. arXiv preprint arXiv:1311.0914. 3. Collobert, R., Bengio, S., & Bengio, Y. (2002). A parallel mixture of SVMs for very large scale problems. Neural computation, 14(5), 1105-1114.

SwarmSVM functions

clusterSVM dcSVM gaterSVM

---

write.alphasvm *Write alphasvm object*

Description

Write alphasvm object

Usage

```
## S3 method for class 'alphasvm'
write(object, svm.file = "Rdata.svm",
  scale.file = "Rdata.scale", yscale.file = "Rdata.yscale")
```

Arguments

object	Object of class "alphasvm", created by alphasvm.
svm.file	filename to export the alphasvm object to.
scale.file	filename to export the scaling data of the explanatory variables to.
yscale.file	filename to export the scaling data of the dependent variable to, if any.

Index