Package 'gmum.r'

April 28, 2015

Type Package

Title Package with models proposed by GMUM at Jagiellonian University

Version 1.0

Date 2014-02-25

Author Stanislaw Jastrzebski, Marcin Data, Karol Jurek, Igor Sieradzki, Konrad Talik,Piotr Kowenzowski, Mateusz Bruno-Kaminski, Maciej Zgliczynski, Michal Pletty, Wojciech Czarnecki

Maintainer Stanislaw Jastrzebski <grimghil@gmail.com>

Description More about what it does (maybe more than one line)

License MIT

Depends MASS (>= 7.3),
Matrix,
Rcpp (>= 0.11.0),
RcppArmadillo(>= 0.4.600.0),
ggplot2(>= 1.0.0),
BH,
igraph

LinkingTo Rcpp, RcppArmadillo, BH

NeedsCompilation yes

Suggests testthat,
SparseM

R topics documented:

plot.svm *plot*

Description

Plots trained svm data and models disciriminative

Usage

```
plot(svm)
```

Arguments

x	trained svm object
X	optional new data points to be predicted and plotted in one of the following formats: data.frame, data.matrix; default: NULL
mode	which plotting mode to use as string, available are: • 'normal' - default mode, plots data in cols argument and a linear decision boundry in available • 'pca' - preforms PCA decomposition and draws data in a subspace of first 2 dimensions from the PCA • 'contour' - countour plot for non-linear kernels
cols	data dimensions to be plotted as vector of length 2, default: c(1,2)
radius	radius of the plotted data points as float, default: 3
radius.max	maximum radius of data points can be plotted, when model is trained with example weights as float, default: 10

predict.svm.gmum *Predict*

Description

Returns predicted classes or distance to discriminative for provided test examples.

Usage

```
predict(svm, x)
```

Arguments

object	Trained SVM object.
x	unlabeled data, in one of the following formats: data.frame, data.matrix, SparseM::matrix.csr, Matrix::Matrix, slam::simple_triplet_matrix
decision.function	if TRUE returns SVMs decision function (distance of a point from discriminant) instead of predicted labels, default: FALSE

print.svm *print*

Description

Prints short summary of the SVM object and its parameters.

Usage

```
print(svm)
```

Arguments

object	SVM object

Format

NULL

summary.svm *summary*

Description

Prints short summary of a trained model.

Usage

```
summary(svm)
```

Arguments

svm	SVM object

SVM *SVM*

Description

Create and train SVM model object.

Usage

```
SVM(x, ...)
```

Arguments

x	training data without labels in one of the following formats: `data.frame`, `data.matrix`, `SparseM::matrix.csr`, `Matrix::Matrix`, `slam::simple_triplet_matrix`
y	labels in one of the followinf formts: `factor`, `vector`. Recommended type is `factor`
data	can be passed instead of `x, y` pair with `formula` to mark the labels column, supported formats are: `data.frame`, `data.matrix`
formula	can be passed with `data` instead of `x, y` pair, formula needs to point to lables column, for example: `target~.`.
core	Support Vector Machine library to use in traning, available are: `'libsvm'`, `'svmlight'`; default: `'libsvm'`
kernel	kernel type as string, available are: `'linear'`, `'poly'`, `'rbf'`, `'sigmoid'`; default: `'linear'` • `linear`: $x' * w$ • `poly`: $(gamma * x' * w + coef0)^{degree}$ • `rbf`: $exp(-gamma * \|x - w\|^2)$ • `sigmoid`: $tanh(gamma * x' * w + coef0)$
prep	preprocess method as string, available are: `'none'`, `'2e'`; default: `'none'`. For more information on `2eSVM` see: http://www.sciencedirect.com/science/article/pii/S0957417414004138
C	cost/complexity parameter, default: `1`
gamma	parameter for `poly`, `rbf` and `sigmoid` kernels, default: `1/n_features`
coef0	for `poly` and `sigmoid` kernels, default: `0`
degree	for `poly` kernel, default: `3`
cache_size	cache memory size in MB, default: `100`
tol	tolerance of termination criterion, default: `1e-3`
max.iter	depending on library: • libsvm: number of iterations after which the training porcess is killed (it can end earlier is desired tolerance is met), default: `1e6`

- svmlight: number of iterations after which if there is no progress traning is killed, default: `-1` (no limit)

`transductive.learning`
: option got SVM model to deduce missing labels from the dataset, default: `FALSE` NOTE: this feature is only available with svmlight library, missing labels are marked as `'TR'`, if none are found and transductive to `TRUE`, label `0` will be interpreted as missing

`transductive.posratio`
: fraction of unlabeled examples to be classified into the positive class as float from $[0, 1]$, default: the ratio of positive and negative examples in the training data

`class.weights`
: named vector with weight fir each class, default: `NULL`

`example.weights`
: vector of the same length as training data with weights for each traning example, default: `NULL` NOTE: this feature is only supported with svmlight library

`class.type`
: multiclass algorithm type as string, available are: `'one.versus.all'`, `'one.versus.one'`; default: `'one.versus.one'`

`verbosity`
: how verbose should the process be, as integer from $[1, 6]$, default: `4`

Value

SVM model object

Examples

```
# train SVM from data in x and labels in y
svm <- SVM(x, y, core="libsvm", kernel="linear", C=1)

# train SVM using a dataset with both data and lables and a formula pointing to labels
formula <- target ~ .
svm <- SVM(formula, data, core="svmlight", kernel="rbf", gamma=1e3)

# train a model with 2eSVM algorithm
data(svm_breast_cancer_dataset)
ds <- svm.breastcancer.dataset
svm.2e <- SVM(x=ds[,-1], y=ds[,1], core="libsvm", kernel="linear", prep = "2e", C=10);
# more at <link to the 2e sample>

# train SVM on a multiclass data set
data(iris)
# with "one vs rest" strategy
svm.ova <- SVM(Species ~ ., data=iris, class.type="one.versus.all", verbosity=0)
# or with "one vs one" strategy
svm.ovo <- SVM(x=iris[,1:4], y=iris[,5], class.type="one.versus.one", verbosity=0)

# we can use svmlights sample weighting feature, suppose we have weights vector
# with a weight for every sample in the traning data
weighted.svm <- SVM(formula=y~., data=df, core="svmlight", kernel="rbf", C=1.0,
                    gamma=0.5, example.weights=weights)
```

```
# svmlight alows us to determine missing labels from a dataset
# suppose we have a labels y with missing labels marked as zeros
svm.transduction <- SVM(x, y, transductive.learning=TRUE, core="svmlight")

# for more in-depth examples visit <link to samples on the website>
```

Index