Package 'SVMFeatureSelectionSystem'

December 30, 2013

Type Package

Title Multiobjective feature selection for SVM and SVR.

Version 1.0

Date 2013-12-23

Author Ing. Jiri Petrlik <ipetrlik@fit.vutbr.cz>

Maintainer Ing. Jiri Petrlik <ipetrlik@fit.vutbr.cz>

Depends e1071

Description Multiobjective feature selection for SVM and SVR.

License BUT OPEN SOURCE LICENCE Version 1. (type licenseSVMFeatureSelectionsSystem())

R topics documented:

`SVMFeatureSelectionSystem-package`
Multiobjective feature selection for SVM and SVR.

Description

This package was created to solve feature selection problem for support vector machine and support vector regression. In feature selection problem we have a set of possible input variables for SVM/SVR and the goal is to choose proper subset of these variables. The package is based on multiobjective genetic algorithm called multimodal NSGAII algorithm. The package also contains tool to visualize the results.

Details

Package:	SVMFeatureSelectionSystem
Type:	Package
Version:	1.0
Date:	2013-12-23
License:	BUT OPEN SOURCE LICENCE Version 1. (type licenseSVMFeatureSelectionsSystem()).

This package was created to solve feature selection problem for support vector machine and support vector regression. Support vector machine (SVM) [1] is popular machine learning algorithm which can be used to solve classification tasks. The variant of SVM called support vector regression (SVR) [2] can be used to solve regression tasks. In feature selection problem we have a set of possible input variables for SVM/SVR and the goal is to choose proper subset of these variables. The package uses multi objective genetic algorithm called multimodal NSGAII [3] to solve this problem. In case of classification the main goal of feature selection is to find the subset of inputs for which the number of misclassified inputs will be minimal. In case of regression the main goal is to find the set of inputs for which the root mean squared error will be minimal. We choose multi objective genetic algorithm [4] because in the real world we need also to minimize the number of inputs and number of samples for which one of the input values is missing.

There are two basic functions in the package. Function featureSelectionClassificationGA is able to find proper inputs of SVM for classification task and the function featureSelectionRegressionGA is able to find proper inputs of SVR for regression task. Both functions return list which contains fitness values of obtained solutions, trained SVM/SVR for each solution and information about convergence of genetic algorithm. The convergence of multiobjective genetic algorithm can be visualized by functions plotConvergenceClassification and plotConvergenceRegression. Usually the most interesting are solutions, which lie on the Pareto front. Functions filterParetoOptimal-Classification and filterParetoOptimalRegression take the results and filter out solutions which are dominated by other obtained solutions. The package provides functions for visualization of obtained results. Functions plotParetoFrontClassification and plotParetoFrontRegression plot various compromises obtained at the end of the run of the genetic algorithm. To obtain results for new data user can use functions predictClassification and predictRegression. Functions mse, rmse, mae, rae and rse provides more information about the quality of prediction for obtained models.

Note

This software was supported by IT4Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070.

Author(s)

Ing. Jiri Petrlik <ipetrlik@fit.vutbr.cz>

References

[1] A Tutorial on Support Vector Machines for Pattern Recognition, C. Burges, Data Mining and Knowledge Discovery, Vol. 2,Issue 2, 1998

[2] Support Vector Regression, Debasish Basak, Srimanta Pal, Dipak Chandra Patranabis, Neuaral Information Processing - Letters and Reviews, Vol. 11, No. 10, 2007

[3] Deb, Kalyanmoy, Raji Reddy, A, Reliable classification of two-class cancer data using evolutionary algorithms, Biosystems, 2003

[4] Deb, Kalyanmoy, Multi-Objective Optimization using Evolutionary Algorithms, WILEY, 2009

Examples

```
# Example of classification task 1:
library(SVMFeatureSelectionSystem);
library(classifly);

features<-colnames(olives)[3:10];
predictedVariable<-colnames(olives)[2];

shuffleOlives<-olives[sample(nrow(olives)),];
dataTrain<-shuffleOlives[1:286,];
dataTest<-shuffleOlives[287:572,];

results<-featureSelectionClassificationGA(dataTrain,dataTest,predictedVariable,
  features);

# Example of classification task 2:
library(SVMFeatureSelectionSystem);
library(plsgenomics);
library(stringr);

data(Colon);
ColonDataset<-as.data.frame(Colon$X);
colnames(ColonDataset)<-str_trim(Colon$gene.names);
ColonDataset<-cbind(ColonDataset,result=Colon$Y);

ColonDataset[ColonDataset[,"result"]==1,"result"]<-"normal";
ColonDataset[ColonDataset[,"result"]==2,"result"]<-"tumor";
ColonDataset[,"result"]<-as.factor(ColonDataset[,"result"]);

ColonDataset<-ColonDataset[sample(nrow(ColonDataset)),];
trainColon<-ColonDataset[1:31,];
testColon<-ColonDataset[32:62,];

features<-colnames(trainColon)[1:2000];
predictedVariable<-colnames(trainColon)[2001];

results<-featureSelectionClassificationGA(trainColon,testColon,
  predictedVariable,features,popSize=100,generations=200);

# Example of regression task 1:
```

```
library(SVMFeatureSelectionSystem);
library(rpart);

car90$Country<-as.numeric(car90$Country);
car90$Model2<-as.numeric(car90$Model2);
car90$Reliability<-as.numeric(car90$Reliability);
car90$Rim<-as.numeric(car90$Rim);
car90$Steering<-as.numeric(car90$Steering);
car90$Tires<-as.numeric(car90$Tires);
car90$Trans1<-as.numeric(car90$Trans1);
car90$Trans2<-as.numeric(car90$Trans2);
car90$Type<-as.numeric(car90$Type);

features<-colnames(car90)[colnames(car90)!="Price"];
predictedVariable<-"Price";

shuffleCar90<-car90[sample(nrow(car90)),]
trainData<-shuffleCar90[1:80,];
testData<-shuffleCar90[81:111,];

results<-featureSelectionRegressionGA(trainData,testData,predictedVariable,
  features);

# Example of regression task 2:
library(SVMFeatureSelectionSystem);
library(mlbench);

sim<-mlbench.friedman1(100);
trainData<-as.data.frame(sim$x);
trainData<-cbind(trainData,V11=sim$y);

sim<-mlbench.friedman1(100);
testData<-as.data.frame(sim$x);
testData<-cbind(testData,V11=sim$y);

features<-colnames(trainData)[1:10];
predictedVariable<-"V11";

results<-featureSelectionRegressionGA(trainData,testData,predictedVariable,
  features);
```

`featureSelectionClassificationGA` *Search for optimal inputs (features) for SVM.*

Description

Multiobjective search for optimal inputs (features) for support vector machine.

Usage

```
featureSelectionClassificationGA(dataTrain, dataTest, predictedVariable,
  features, popSize = 40, generations = 100, ...)
```

Arguments

dataTrain	Data frame with training data for SVM.
dataTest	Data frame with testing data for validation of trained models.
predictedVariable	Column in dataTrain and dataTest which contains correct results.
features	The vector with names of columns, which contains inputs for SVM.
popSize	Size of populotaion of multimodal NSGAII algorithm.
generations	The number of generations to evolve.
...	Settings of SVM. These parameters corresponds to parameters of command svm in e1071 package.

Value

Function returns list with these components:

results	Data frame with summary of obtained results
modelList	List of obtained models
resultsList	List with values obtained by models for train set and test set
statistic	Information about convergence of the genetic algorithm

Author(s)

Ing. Jiri Petrlik

References

[1] Deb, Kalyanmoy, Raji Reddy, A, Reliable classification of two-class cancer data using evolutionary algorithms, Biosystems, 2003

Examples

```
# Example of classification task 1:

library(SVMFeatureSelectionSystem);
library(classifly);

features<-colnames(olives)[3:10];
predictedVariable<-colnames(olives)[2];

shuffleOlives<-olives[sample(nrow(olives)),];
dataTrain<-shuffleOlives[1:286,];
dataTest<-shuffleOlives[287:572,];

results<-featureSelectionClassificationGA(dataTrain,dataTest,predictedVariable,
  features);

# Example of classification task 2:

library(SVMFeatureSelectionSystem);
library(plsgenomics);
library(stringr);
```

```
data(Colon);
ColonDataset<-as.data.frame(Colon$X);
colnames(ColonDataset)<-str_trim(Colon$gene.names);
ColonDataset<-cbind(ColonDataset,result=Colon$Y);

ColonDataset[ColonDataset[,"result"]==1,"result"]<-"normal";
ColonDataset[ColonDataset[,"result"]==2,"result"]<-"tumor";
ColonDataset[,"result"]<-as.factor(ColonDataset[,"result"]);

ColonDataset<-ColonDataset[sample(nrow(ColonDataset)),];
trainColon<-ColonDataset[1:31,];
testColon<-ColonDataset[32:62,];

features<-colnames(trainColon)[1:2000];
predictedVariable<-colnames(trainColon)[2001];

results<-featureSelectionClassificationGA(trainColon,testColon,
  predictedVariable,features,popSize=100,generations=200);
```

`featureSelectionRegressionGA` *Search for optimal inputs (features) for SVR.*

Description

Multiobjective search for optimal inputs (features) for support vector regression.

Usage

```
featureSelectionRegressionGA(dataTrain, dataTest, predictedVariable, features,
  popSize = 40, generations = 100, ...)
```

Arguments

`dataTrain`	Data frame with training data for SVR.
`dataTest`	Data frame with testing data for validation of trained models.
`predictedVariable`	Column in dataTrain and dataTest which contains correct results.
`features`	The vector with names of columns, which contains inputs for SVR.
`popSize`	Size of populotaion of multimodal NSGAII algorithm.
`generations`	The number of generations to evolve.
`...`	Settings of SVR. These parameters corresponds to parameters of command svm in e1071 package.

Value

Function returns list with these components:

`results`	Data frame with summary of obtained results
`modelList`	List of obtained models
`resultsList`	List with values obtained by models for train set and test set
`statistic`	Information about convergence of the genetic algorithm

Author(s)

Ing. Jiri Petrlik

References

[1] Deb, Kalyanmoy, Raji Reddy, A, Reliable classification of two-class cancer data using evolutionary algorithms, Biosystems, 2003

Examples

```
# Example of regression task 1:
library(SVMFeatureSelectionSystem);
library(rpart);

car90$Country<-as.numeric(car90$Country);
car90$Model2<-as.numeric(car90$Model2);
car90$Reliability<-as.numeric(car90$Reliability);
car90$Rim<-as.numeric(car90$Rim);
car90$Steering<-as.numeric(car90$Steering);
car90$Tires<-as.numeric(car90$Tires);
car90$Trans1<-as.numeric(car90$Trans1);
car90$Trans2<-as.numeric(car90$Trans2);
car90$Type<-as.numeric(car90$Type);

features<-colnames(car90)[colnames(car90)!="Price"];
predictedVariable<-"Price";

shuffleCar90<-car90[sample(nrow(car90)),]
trainData<-shuffleCar90[1:80,];
testData<-shuffleCar90[81:111,];

results<-featureSelectionRegressionGA(trainData,testData,predictedVariable,
  features);

# Example of regression task 2:
library(SVMFeatureSelectionSystem);
library(mlbench);

sim<-mlbench.friedman1(100);
trainData<-as.data.frame(sim$x);
trainData<-cbind(trainData,V11=sim$y);

sim<-mlbench.friedman1(100);
testData<-as.data.frame(sim$x);
testData<-cbind(testData,V11=sim$y);

features<-colnames(trainData)[1:10];
predictedVariable<-"V11";

results<-featureSelectionRegressionGA(trainData,testData,predictedVariable,
  features);
```

`filterParetoOptimalClassification` *Filter only solutions which are on Pareto front.*

Description

This function takes the result of function featureSelectionClassificationGA and filter only solutions which are on Pareto front.

Usage

```
filterParetoOptimalClassification(results)
```

Arguments

`results`	Output of function featureSelectionClassificationGA.

Value

The output is the same as output produced by function featureSelectionClassificationGA. The only difference is such that solutions which are not on Pareto front are filtered out.

Author(s)

Ing. Jiri Petrlik

`filterParetoOptimalRegression` *Filter only solutions which are on Pareto front.*

Description

This function takes the result of function featureSelectionClassificationGA and filter only solutions which are on Pareto front.

Usage

```
filterParetoOptimalRegression(results)
```

Arguments

`results`	Output of function featureSelectionRegressionGA.

Value

The output is the same as output produced by function featureSelectionRegressionGA. The only difference is such that solutions which are not on Pareto front are filtered out.

Author(s)

Ing. Jiri Petrlik

`licenseSVMFeatureSelectionsSystem` *Print license of the package.*

Description

Print license of the package.

Usage

```
licenseSVMFeatureSelectionsSystem()
```

Author(s)

Ing. Jiri Petrlik

`mae` *Calculate mean absolute error.*

Description

Calculate mean absolute error [1].

Usage

```
mae(results, number, set = "test")
```

Arguments

`results`	Set of results obtained by function featureSelectionRegressionGA.
`number`	Number of solution in the set of results.
`set`	Informs if use training data or testing data.

Value

Mean absolute error.

Author(s)

Ing. Jiri Petrlik

References

[1] Data Mining Concepts and Techniques, Jiawei Han, Micheline Kamber, Morgan Kaufmann, 2006

`mse` *Calculate mean squared error.*

Description

Calculate mean squared error [1].

Usage

```
mse(results, number, set = "test")
```

Arguments

`results`	Set of results obtained by function featureSelectionRegressionGA.
`number`	Number of solution in the set of results.
`set`	Informs if use training data or testing data.

Value

Mean squared error.

Author(s)

Ing. Jiri Petrlik

References

[1] Data Mining Concepts and Techniques, Jiawei Han, Micheline Kamber, Morgan Kaufmann, 2006

`plotConvergenceClassification` *Plot convergence of fitness functions for featureSelectionClassificationGA.*

Description

Plot convergence of fitness functions for featureSelectionClassificationGA. The function plots three figures. The first figure is for misclassification ratio, the second is for count of features and the third is for unavailable ratio. The red line shows the minimal value of fitness in population. The black line shows the mean value of fitness in population.

Usage

```
plotConvergenceClassification(results)
```

Arguments

`results`	Output of function featureSelectionClassificationGA.

Author(s)

Ing. Jiri Petrlik

`plotConvergenceRegression`
Plot convergence of fitness functions for featureSelectionRegressionGA.

Description

Plot convergence of fitness functions for featureSelectionRegressionGA. The function plots three figures. The first figure is for root mean squared error (RMSE), the second is for count of features and the third is for unavailable ratio. The red line shows the minimal value of fitness in population. The black line shows the mean value of fitness in population.

Usage

```
plotConvergenceRegression(results)
```

Arguments

`results`	Output of function featureSelectionRegressionGA.

Author(s)

Ing. Jiri Petrlik

`plotParetoFrontClassification`
Plot Pareto fronts.

Description

Plot Pareto fronts. This function plots three Pareto fronts. The first with compromises between missclassified ratio and count of features, the second with compromises between missclassified ratio and unavalaible ratio and third with compromisses between feature count and unavalaible ratio. All is ploted on test data set.

Usage

```
plotParetoFrontClassification(results)
```

Arguments

`results`	Set of results obtained by function featureSelectionClassificationGA.

Author(s)

Ing. Jiri Petrlik

plotParetoFrontRegression

Plot Pareto fronts.

Description

Plot Pareto fronts. This function plots three Pareto fronts. The first with compromises between root mean squared error and count of features, the second with compromises between root mean squared error and unavalaible ratio and third with compromisses between feature count and unavalaible ratio. All is ploted on test data set.

Usage

```
plotParetoFrontRegression(results)
```

Arguments

results	Set of results obtained by function featureSelectionPredictionGA.

Author(s)

Ing. Jiri Petrlik

predictClassification

Use SVM model to classify new data.

Description

Perform classification on new unknown data.

Usage

```
predictClassification(results, number, data)
```

Arguments

results	Output of function featureSelectionClassificationGA. This output contains trained models.
number	Number of solution. Model of this solution is used for classification.
data	New data which will be classified.

Value

Factor with classification results.

Author(s)

Ing. Jiri Petrlik

predictRegression *Use SVR model for regression on new data.*

Description

Perform regression on new unknown data.

Usage

```
predictRegression(results, number, data)
```

Arguments

results	Output of function featureSelectionRegressionGA. This output contains trained models.
number	Number of solution. Model of this solution is used for regression.
data	New data which for which regression will be performed.

Value

Vector with regression results.

Author(s)

Ing. Jiri Petrlik

rae *Calculate relative absolute error.*

Description

Calculate relative absolute error [1].

Usage

```
rae(results, number, set = "test")
```

Arguments

results	Set of results obtained by function featureSelectionRegressionGA.
number	Number of solution in the set of results.
set	Informs if use training data or testing data.

Value

Relative absolute error.

Author(s)

Ing. Jiri Petrlik

References

[1] Data Mining Concepts and Techniques, Jiawei Han, Micheline Kamber, Morgan Kaufmann, 2006

`rmse` *Calculate root mean squared error.*

Description

Calculate root mean squared error [1].

Usage

```
rmse(results, number, set = "test")
```

Arguments

`results`	Set of results obtained by function featureSelectionRegressionGA.
`number`	Number of solution in the set of results.
`set`	Informs if use training data or testing data.

Value

Root mean squared error.

Author(s)

Ing. Jiri Petrlik

References

[1] Data Mining Concepts and Techniques, Jiawei Han, Micheline Kamber, Morgan Kaufmann, 2006

`rse` *Calculate relative squared error.*

Description

Calculate relative squared error [1].

Usage

```
rse(results, number, set = "test")
```

Arguments

`results`	Set of results obtained by function featureSelectionRegressionGA.
`number`	Number of solution in the set of results.
`set`	Informs if use training data or testing data.

Value

Calculate relative squared error.

Author(s)

Ing. Jiri Petrlik

References

[1] Data Mining Concepts and Techniques, Jiawei Han, Micheline Kamber, Morgan Kaufmann, 2006

Index