

# Package ‘penalizedSVM’

April 19, 2009

**Type** Package

**Title** Feature Selection SVM using penalty functions

**Version** 1.0

**Date** 2008-12-15

**Depends** e1071, MASS, corpcor, statmod

**Author** Natalia Becker, Wiebke Werft, Axel Benner

**Maintainer** Natalia Becker <natalia.becker@dkfz.de>

**Description** This package provides feature selection SVM using penalty functions. The smoothly clipped absolute deviation (SCAD) and ‘L1-norm’ penalties are available up to now. Other functions will be implemented in the near future.

**License** GPL (>= 2)

**LazyLoad** yes

**Repository** CRAN

**Date/Publication** 2009-01-14 18:25:26

## R topics documented:

penalizedSVM-package . . . . .	2
findgacv.scad . . . . .	2
lpsvm . . . . .	3
predict . . . . .	5
print . . . . .	7
scadsvc . . . . .	8
sim.data . . . . .	9
svm.fs . . . . .	11

<b>Index</b>	<b>14</b>
--------------	-----------

---

 penalizedSVM-package

*Feature Selection SVM using Penalty Functions*


---

## Description

Feature Selection SVM using penalty functions. The smoothly clipped absolute deviation (SCAD) and L1-norm penalties are available up to now. Other functions will be implemented in the near future.

## Details

Package:	penaltySVM
Type:	Package
Version:	1.0
Date:	2008-12-15
License:	GPL-2
LazyLoad:	yes

The main function is `svm.fs`, see the documentation file with examples

## Author(s)

Natalia Becker, Axel Benner, Wiebke Werft

Maintainer: Natalia Becker (natalia.becker@dkfz.de)

## References

Zhang, H. H., Ahn, J., Lin, X. and Park, C. (2006). *Gene selection using support vector machines with nonconvex penalty*. *Bioinformatics*, **22**, pp. 88-95.

Fung, G. and Mangasarian, O. L. (2004). *A feature selection newton method for support vector machine classification*. *Computational Optimization and Applications Journal*, **28.2**, pp. 185-202.

---

 findgacv.scad

*Calculate Generalized Approximate Cross Validation Error Estimation for SCAD SVM model*


---

## Description

calculate generalized approximate cross validation error (GACV) estimation for SCAD SVM Model

## Usage

```
findgacv.scad(y, model)
```

**Arguments**

`y` vector of class labels (only for 2 classes)  
`model` list, describing SCAD SVM model, produced by function `scadsvc`

**Value**

returns the GACV value

**Author(s)**

Natalia Becker (natalia.becker@dkfz.de)

**References**

Zhang, H. H., Ahn, J., Lin, X. and Park, C. (2006). *Gene selection using support vector machines with nonconvex penalty*. *Bioinformatics*, **22**, pp. 88-95.

Wahba G., Lin, Y. and Zhang, H. (2000). *GACV for support vector machines, or, another way to look at margin-like quantities*, in A. J. Smola, P. Bartlett, B. Schoelkopf and D. Schuurmans (eds), *Advances in Large Margin Classifiers*, MIT Press, pp. 297-309.

**See Also**

[scadsvc](#), [predict.penSVM](#), [sim.data](#)

**Examples**

```
# simulate data
train<-sim.data(n = 200, ng = 100, nsg = 10, corr=FALSE, seed=12)
print(str(train))

# train data
ff <- scadsvc(as.matrix(t(train$x)), y=train$y, lambda=0.01)
print(ff)

# estimate gacv error
(gacv<- findgacv.scad(train$y, model=ff))
```

---

lpsvm

*Fit L1-norm SVM*

---

**Description**

SVM mit variable selection (clone selection) using L1-norm penalty. ( a fast Newton algorithm NLPSVM from Fung and Mangasarian )

**Usage**

```
lpsvm(A, d, k = 5, nu = 0, output = 1, delta = 10^-3, epsi = 10^-4, my.seed = 123)
```

**Arguments**

A	n-by-d data matrix to train (n chips/patients, d clones/genes)
d	vector of class labels -1 or 1's (for n chips/patients )
k	k-fold for cv, default k=5
nu	weighted parameter, 1 - easy estimation, 0 - hard estimation, any other value - used as nu by the algorithm. default : 0
output	0 - no output, 1 - produce output, default is 0
delta	some small value, default $10^{-3}$
epsi	tuning parameter
my.seed	seed

**Details**

k: k-fold for cv, is a way to divide the data set into test and training set.  
 if k = 0: simply run the algorithm without any correctness calculation, this is the default.  
 if k = 1: run the algorithm and calculate correctness on the whole data set.  
 if k = any value less than the number of rows in the data set: divide up the data set into test and training using k-fold method.  
 if k = number of rows in the data set: use the 'leave one out' (loo) method

**Value**

a list of	
w	coefficients of the hyperplane
b	intercept of the hyperplane
xind	the index of the selected features (genes) in the data matrix.
epsi	optimal tuning parameter epsilon
iter	number of iterations
k	k-fold for cv
trainCorr	for cv: average train correctness
testCorr	for cv: average test correctness
nu	weighted parameter

**Note**

Adapted from MATLAB code <http://www.cs.wisc.edu/dmi/svm/lpsvm/>

**Author(s)**

Natalia Becker

**References**

Fung, G. and Mangasarian, O. L. (2004). *A feature selection newton method for support vector machine classification*. Computational Optimization and Applications Journal 28(2) pp. 185-202.

**See Also**

[sim.data](#)

**Examples**

```
train<-sim.data(n = 20, ng = 100, nsg = 10, corr=FALSE, seed=12)
print(str(train))

# train data
model <- lpsvm(A=t(train$x), d=train$y, k=5, nu=0,output=0, delta=10^-3, epsi=0.001, my.seed)
print(model)
```

---

predict

*Predict Method for Feature Selection SVM*

---

**Description**

This function predicts values based upon a model trained by svm. If class assignment is provided, confusion table, missclassification table, sensitivity and specificity are calculated.

**Usage**

```
## S3 method for class 'penSVM':
predict(object, newdata, newdata.labels = NULL, ...)
```

**Arguments**

object	Object of class "penSVM", created by 'svm.fs'
newdata	A matrix containing the new input data, samples in rows, features in columns
newdata.labels	optional, new data class labels
...	additional argument(s)

**Value**

returns a list of prediction values for classes

pred.class	predicted class
tab	confusion table
error	missclassification error
sensitivity	sensitivity
specificity	specificity

**Author(s)**

Natalia Becker

**See Also**[svm](#), [svm.fs](#)**Examples**

```

my.seed<- 123
train<-sim.data(n = 200, ng = 100, nsg = 10, corr=FALSE, seed=my.seed )
print(str(train))

#train standard svm
my.svm<-svm(x=t(train$x), y=train$y, kernel="linear")

# test with other data
test<- sim.data(n = 200, ng = 100, nsg = 10, seed=(my.seed+1) )

# Check accuracy standard SVM
my.pred <-ifelse( predict(my.svm, t(test$x)) >0,1,-1)
# Check accuracy:
table(my.pred, test$y)

# define set values of tuning parameter lambda1 for SCAD
lambda1.scad <- c (seq(0.01 ,0.05, .01), seq(0.1,0.5, 0.2), 1 )
# for presentation don't check all lambdas : time consuming!
fit.scad<- svm.fs(x=t(train$x),y=train$y, fs.method="scad", cross.outer= 0, lambda1.set=lambda1.scad)

# SCAD
test.error.scad<-predict(fit.scad, newdata=t(test$x),newdata.labels=test$y )
# Check accuracy SCAD SVM
print(test.error.scad$tab)

#####
# analog for 1-norm SVM
epsi.set<-vector(); for (num in (1:9)) epsi.set<-sort(c(epsi.set, c(num*10^seq(-5, -1, 1 ))))
lambda1.lnorm <- epsi.set[c(3,5)] # 2 params

# train lnorm SVM
# time consuming: for presentation only for the first 100 samples
## DON'T RUN : fit.lnorm<- svm.fs(x=t(train$x),y=train$y, fs.method="lnorm", cross.outer= 0,
fit.lnorm<- svm.fs(x=t(train$x)[1:100,],y=train$y[1:100], fs.method="lnorm", cross.outer= 0,

# L1-norm SVM
test.error.lnorm<-predict(fit.lnorm, newdata=t(test$x),newdata.labels=test$y )
# Check accuracy L1-norm SVM
print(test.error.lnorm$tab)

```

---

`print`*Print Function for FS SVM*

---

## Description

Print Function for FS SVM

## Usage

```
## S3 method for class 'scadsvm':  
print(x, ...)  
## S3 method for class 'l1norm.svm':  
print(x, ...)
```

## Arguments

<code>x</code>	model trained by scad or l1norm svm
<code>...</code>	additional argument(s)

## Author(s)

Natalia Becker

## See Also

[svm](#), [svm.fs](#)

## Examples

```
my.seed<- 123  
train<-sim.data(n = 20, ng = 100, nsg = 10, corr=FALSE, seed=my.seed )  
print(str(train))  
  
# for presentation don't check all lambdas : time consuming!  
model <- scadsvc(as.matrix(t(train$x)), y=train$y, lambda=0.05)  
print(model)
```

---

 scadsvc

*Fit SCAD SVM model*


---

**Description**

SVM with variable selection (clone selection) using SCAD penalty.

**Usage**

```
scadsvc(lambda = 0.01, x, y, a = 3.7, tol = 10(-4), class.weights = NULL)
```

**Arguments**

<code>lambda</code>	tuning parameter in SCAD function (default : 0.01)
<code>x</code>	n-by-d data matrix to train (n chips/patients, d clones/genes)
<code>y</code>	vector of class labels -1 or 1's (for n chips/patients )
<code>a</code>	tuning parameter in scad function (default: 3.7)
<code>tol</code>	the cut-off value to be taken as 0
<code>class.weights</code>	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. (default: NULL)

**Details**

Adopted from Matlab code: <http://www4.stat.ncsu.edu/~hzhang/software.html>

**Value**

a list of	
<code>w</code>	coefficients of the hyperplane
<code>b</code>	intercept of the hyperplane
<code>xind</code>	the index of the selected features (genes) in the data matrix.
<code>index</code>	the index of the resulting support vectors in the data matrix.
<code>type</code>	type of svm, from svm function
<code>lam.opt</code>	optimal lambda
<code>gacv</code>	corresponding gacv

**Author(s)**

Axel Benner

## References

Zhang, H. H., Ahn, J., Lin, X. and Park, C. (2006). *Gene selection using support vector machines with nonconvex penalty*. *Bioinformatics*, **22**, pp. 88-95.

## See Also

`findgacv.scad`, `predict.penSVM`, `sim.data`

## Examples

```
# simulate data
train<-sim.data(n = 200, ng = 100, nsg = 10, corr=FALSE, seed=12)
print(str(train))

# train data
model <- scadsvc(as.matrix(t(train$x)), y=train$y, lambda=0.01)
print(model)
```

---

sim.data

*Simulation of microarray data*

---

## Description

Simulation of 'n' samples. Each sample has 'sg' genes, only 'nsg' of them are called significant and have influence on class labels. All other '(ng - nsg)' genes are called ballanced. All gene ratios are drawn from a multivariate normal distribution. There is a possibility to create blocks of highly correlated genes.

## Usage

```
sim.data(n = 256, ng = 1000, nsg = 100,
         p.n.ratio = 0.5,
         sg.pos.factor= 1, sg.neg.factor= -1,
         # correlation info:
         corr = FALSE, corr.factor = 0.8,
         # block info:
         blocks = FALSE, n.blocks = 6, nsg.block = 1, ng.block = 5,
         seed = 12345, ...)
```

## Arguments

n	number of samples, logistic regression works well if $n > 200!$
ng	number of genes
nsg	number of significant genes
p.n.ratio	ratio between positive and negative significant genes (default 0.5)

```

sg.pos.factor      impact factor of _positive_ significant genes on the classification, default: 1
sg.neg.factor      impact factor of _negative_ significant genes on the classification, default: -1
corr               are the genes correlated to each other? (default FALSE). see Details
corr.factor        correlation factor for genes, between 0 and 1 (default 0.8)
blocks            are blocks of highly correlated genes allowed? (default FALSE)
n.blocks           number of blocks
nsg.block          number of significant genes per block
ng.block           number of genes per block
seed              seed
...              additional argument(s)

```

### Details

If no blocks (`n.blocks=0` or `blocks=FALSE`) are defined and `corr=TRUE` create covariance matrix for all genes! with decrease of correlation :  $cov(i, j) = cov(j, i) = corr.factor^{(i - j)}$

### Value

```

x          matrix of simulated data. Genes in rows and samples in columns
y          named vector of class labels
seed       seed

```

### Author(s)

Wiebke Werft, Natalia Becker

### See Also

[mvrnorm](#)

### Examples

```

my.seed<-123

# 1. simulate 20 samples, with 100 genes in each. Only the first two genes have an impact on
# All genes are assumed to be i.i.d.
train<-sim.data(n = 20, ng = 100, nsg = 3, corr=FALSE, seed=my.seed )
print(str(train))

# 2. change the proportion between positive and negative significant genes (from 0.5 to 0.8)
train<-sim.data(n = 20, ng = 100, nsg = 10, p.n.ratio = 0.8, seed=my.seed )
rownames(train$x) [1:15]
# [1] "pos1" "pos2" "pos3" "pos4" "pos5" "pos6" "pos7" "pos8"
# [2] "neg1" "neg2" "bal1" "bal2" "bal3" "bal4" "bal5"

```

```

# 3. assume to have correlation for positive significant genes, negative significant genes
train<-sim.data(n = 20, ng = 100, nsg = 10, corr=TRUE, seed=my.seed )
cor(t(train$x[1:15,]))

# 4. add 6 blocks of 5 genes each and only one significant gene per block.
# all genes in the block are correlated with constant correlation factor corr.factor=0.8
train<-sim.data(n = 20, ng = 100, nsg = 6, corr=TRUE, corr.factor=0.8,
               blocks=TRUE, n.blocks=6, nsg.block=1, ng.block=5, seed=my.seed )

print(str(train))
# first block
cor(t(train$x[1:5,]))
# second block
cor(t(train$x[6:10,]))

```

svm.fs

*Fits SVM mit variable selection using penalties.***Description**

Fits SVM with variable selection (clone selection) using penalties SCAD and L1 norm.

**Usage**

```

## Default S3 method:
svm.fs(x, y, fs.method = "l1norm", cross.outer = 0, lambda1.set, lambda2.set = NULL,
run.scad(x,y, lambda1.set=NULL, class.weights)
run.l1norm(x,y,k=5,nu=0, lambda1.set=NULL, output=1, seed=seed)

```

**Arguments**

x	input matrix with genes in columns and samples in rows!
y	vector of class labels
fs.method	feature selection method. Available 'scad' and 'l1norm'
cross.outer	fold of outer cross validation, default is 0, no cv.
lambda1.set	set of tuning parameters lambda1
lambda2.set	set of tuning parameters lambda2, not yet in use
calc.class.weights	calculate class.weights for SVM, default: FALSE
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.
k	k-fold cross validation, default: 5
nu	nu: weighted parameter

	<ul style="list-style-type: none"> <li>• 1 - easy estimation,</li> <li>• 0 - hard estimation,</li> <li>• any other value - used as nu by the algorithm, default - 0.</li> </ul>
output	0 - no output, 1 - produce output, default is 0
seed	seed
maxIter	maximal iteration, default: not used yet
...	additional argument(s)

### Details

The goodness of the model is highly correlated with the choice of tuning parameter lambda. Therefore the model is trained with different lambdas and the best model with optimal tuning parameter is used in further analyses.

The Feature Selection methods are using different techniques for finding optimal tuning parameters. By SCAD SVM Generalized approximate cross validation (gacv) error is calculated for each pre-defined tuning parameter.

By L1-norm SVM the cross validation (default 5-fold) missclassification error is calculated for each lambda. After training and cross validation, the optimal lambda with minimal missclassification error is chosen, and a final model with optimal lambda is created for the whole data set.

### Value

classes	vector of class labels as input 'y'
sample.names	sample names
class.method	feature selection method
cross.outer	outer cv
seed	seed
model	final model <ul style="list-style-type: none"> <li>• w - coefficients of the hyperplane</li> <li>• b - intercept of the hyperplane</li> <li>• xind - the index of the selected features (genes) in the data matrix.</li> <li>• index - the index of the resulting support vectors in the data matrix.</li> <li>• type - type of svm, from svm function</li> <li>• lam.opt - optimal lambda</li> <li>• gacv - corresponding gacv</li> </ul>

### Author(s)

Natalia Becker natalia.becker at dkfz.de

## References

Zhang, H. H., Ahn, J., Lin, X. and Park, C. (2006). *Gene selection using support vector machines with nonconvex penalty*. *Bioinformatics*, **22**, pp. 88-95.

Fung, G. and Mangasarian, O. L. (2004). *A feature selection newton method for support vector machine classification*. *Computational Optimization and Applications Journal*, **28(2)**, pp. 185-202.

## See Also

[predict.penSVM](#), [svm](#) (in package **e1071**)

## Examples

```
my.seed<- 123

train<-sim.data(n = 200, ng = 100, nsg = 10, corr=FALSE, seed=my.seed )
print(str(train))

# train SCAD SVM #####
# define set values of tuning parameter lambda1 for SCAD
lambda1.scad <- c (seq(0.01 ,0.05, .01), seq(0.1,0.5, 0.2), 1 )
# for presentation don't check all lambdas : time consuming!
lambda1.scad<-lambda1.scad[2:3]
#
# train SCAD SVM
fit.scad<- svm.fs(x=t(train$x),y=train$y, fs.method="scad", cross.outer= 0, lambda1.set=lambda1.scad)

# train 1NORM SVM #####
# define set values of tuning parameter lambda1 for 1norm
epsi.set<-vector(); for (num in (1:9)) epsi.set<-sort(c(epsi.set, c(num*10^seq(-5, -1, 1 ))))
# for presentation don't check all lambdas : time consuming!
lambda1.1norm <- epsi.set[c(3,5)] # 2 params

# train 1norm SVM
# time consuming: for presentation only for the first 100 samples
## DON'T RUN : fit.1norm<- svm.fs(x=t(train$x),y=train$y, fs.method="1norm", cross.outer= 0,
fit.1norm<- svm.fs(x=t(train$x) [1:100,],y=train$y[1:100], fs.method="1norm", cross.outer= 0,
```

# Index

- \*Topic **distribution**
  - sim.data, 9
- \*Topic **models**
  - findgacv.scad, 2
  - svm.fs, 11
- \*Topic **multivariate**
  - sim.data, 9
  - svm.fs, 11
- \*Topic **optimize**
  - svm.fs, 11
- \*Topic **package**
  - penalizedSVM-package, 1

findgacv.scad, 2, 8

lpsvm, 3

mvrnorm, 10

penalizedSVM

- (penalizedSVM-package), 1

penalizedSVM-package, 1

predict, 5

predict.penSVM, 3, 8, 12

print, 6

print.lnorm.svm(print), 6

print.scadsvm(print), 6

run.lnorm(svm.fs), 11

run.scad(svm.fs), 11

scadsvc, 3, 7

sim.data, 3, 4, 8, 9

svm, 6, 7, 12

svm.fs, 2, 6, 7, 11