

Package ‘LiblineaR’

January 2, 2012

Title Linear Predictive Models Based On The Liblinear C/C++ Library.

Version 1.80-4

Author Thibault Helleputte <thelleputte@gmail.com>

Maintainer Thibault Helleputte <thelleputte@gmail.com>

Description This R package is a wrapper around the liblinear C/C++ library for machine learning. LIBLINEAR is a linear classifier for data with millions of instances and features. It supports L2-regularized classifiers (such as L2-loss linear SVM, L1-loss linear SVM, and logistic regression) as well as L1-regularized classifiers (such as L2-loss linear SVM and logistic regression). The main features of LiblineaR include multi-class classification (one-vs-the rest, and Crammer & Singer method), cross validation for model selection, probability estimates (logistic regression only) or weights for unbalanced data. The estimation of the models is particularly fast as compared to other libraries. For more information on the C/C++ liblinear library itself, refer to R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research 9(2008), 1871-1874, available at <http://www.csie.ntu.edu.tw/~cjlin/liblinear> . The two first blocks of the package version indicates which version of liblinear is currently supported. For example: 1.32-14 means that the package supports the version 1.32 of liblinear.

License GPL-2

Date 2011-04-20

LazyLoad yes

URL <http://www.thibaulthelleputte.be>

Repository CRAN

Date/Publication 2011-04-23 20:54:23

R topics documented:

LiblineaR-package	2
heuristicC	3
LiblineaR	5
predict.LiblineaR	8

Index	11
--------------	-----------

LiblineaR-package	<i>Linear Predictive Models Based On The Liblinear C/C++ Library.</i>
-------------------	---

Description

This R package is a wrapper around the liblinear C/C++ library for machine learning. It allows the estimation of predictive linear models such as L1- or L2-regularized logistic regression, L1- or L2-regularized L2-loss support vector classification, L2-regularized L1-loss support vector classification and multi-class support vector classification. The estimation of the models is particularly fast as compared to other libraries. For more information on liblinear itself, refer to R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification, *Journal of Machine Learning Research* 9(2008), 1871-1874. Software available at <http://www.csie.ntu.edu.tw/~cjlin/liblinear>. The two first blocks of the package version indicates which version of liblinear is currently supported. For example: 1.32-14 means that the package supports the version 1.32 of liblinear.

Details

Package:	LiblineaR
Type:	Package
Version:	1.7-0
Date:	2011-03-31
License:	GPL-2
OS_type:	unix

The package is a wrapper allowing R users to make use of the liblinear C/C++ library. The two main functions of this package are `LiblineaR` itself, and `predict.LiblineaR`. The first one trains a (generalized) linear model on a given dataset, and the second one uses this model to make prediction (classification) on new data. Multi-class classification is also provided.

Author(s)

Thibault Helleputte

Maintainer: Thibault Helleputte <thelleputte@gmail.com>

References

- This R Package <http://www.thibaultelleputte.be>
- Liblinear R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, *LIBLINEAR: A Library for Large Linear Classification*, Journal of Machine Learning Research 9(2008), 1871-1874. <http://www.csie.ntu.edu.tw/~cjlin/liblinear>.

heuristicC	<i>Fast Heuristics For The Estimation Of the C Constant Of A Support Vector Machine.</i>
------------	--

Description

heuristicC implements a heuristics proposed by Thorsten Joachims in order to make fast estimates of a convenient value for the C constant used by support vector machines. This implementation only works for linear support vector machines.

Usage

```
heuristicC(data)
```

Arguments

`data` a `n x p` data matrix. Each row stands for an example (sample, point) and each column stands for a dimension (feature, variable)

Value

A value for the C constant is returned, computed as follows:

$$\frac{1}{\frac{1}{n} \sum_{i=1}^n \sqrt{G[i, i]}}$$

where

$$G = \text{data} \% * \%t(\text{data})$$

Note

Classification models usually perform better if each dimension of the data is first centered and scaled. If data are scaled, it is better to compute the heuristics on the scaled data as well.

Author(s)

Thibault Helleputte
<thelleputte@gmail.com>

References

- T. Joachims
SVM light (2002)
<http://svmlight.joachims.org>

See Also

[LiblineaR](#)

Examples

```
data(iris)
attach(iris)

x=iris[,1:4]
y=factor(iris[,5])
train=sample(1:dim(iris)[1],100)

xTrain=x[train,]
xTest=x[-train,]
yTrain=y[train]
yTest=y[-train]

# Center and scale data
s=scale(xTrain,center=TRUE,scale=TRUE)

# Sparse Logistic Regression
t=6

# Tune the cost parameter of a logistic regression according to the Joachim's heuristics
co=heuristicC(s)
m=LiblineaR(data=s,labels=yTrain,type=t,cost=co,bias=TRUE,verbose=FALSE)

# Scale the test data
s2=scale(xTest,attr(s,"scaled:center"),attr(s,"scaled:scale"))

# Make prediction
p=predict(m,s2)

# Display confusion matrix
res=table(p$predictions,yTest)
print(res)

# Compute Balanced Classification Rate
BCR=mean(c(res[1,1]/sum(res[,1]),res[2,2]/sum(res[,2]),res[3,3]/sum(res[,3])))
print(BCR)
```

LiblineaR	<i>Linear Predictive Models Estimation Based On The Liblinear C/C++ Library.</i>
-----------	--

Description

LiblineaR allows the estimation of predictive linear models such as L1- or L2-regularized logistic regression, L1- or L2-regularized L2-loss support vector classification, L2-regularized L1-loss support vector classification and multi-class support vector classification. The estimation of the models is particularly fast as compared to other libraries. The implementation is based on the liblinear C/C++ library for machine learning. Multiclass classification is supported.

Usage

```
LiblineaR(data, labels, type=0, cost=1, epsilon = 0.01, bias = TRUE, wi = NULL, cross = 0, verbose = FALSE)
```

Arguments

data	a n x p data matrix. Each row stands for an example (sample, point) and each column stands for a dimension (feature, variable)
labels	a response vector for classification tasks with one label for each of the n rows of data. Can be a 1 x n matrix, a simple vector or a factor.
type	LiblineaR can produce 7 types of (generalized) linear models, by combining several types of loss functions and regularization schemes. The regularization can be L1 or L2, and the losses can be the regular L2-loss for SVM (hinge loss), L1-loss for SVM, or the logistic loss for logistic regression. The default value for type is 0. See details below. Valid options are: <ul style="list-style-type: none"> • 0 – L2-regularized logistic regression • 1 – L2-regularized L2-loss support vector classification (dual) • 2 – L2-regularized L2-loss support vector classification (primal) • 3 – L2-regularized L1-loss support vector classification (dual) • 4 – multi-class support vector classification by Crammer and Singer • 5 – L1-regularized L2-loss support vector classification • 6 – L1-regularized logistic regression • 7 – L2-regularized logistic regression (dual)
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 information on the 'C' constant in details below. A usually good baseline heuristics to tune this constant is provided by the heuristicC function of this package.
epsilon	set tolerance of termination criterion for optimization. Default to 0.01. If NULL, the liblinear defaults are used, which are: <p>if type is 0, 2, 5 or 6 epsilon=0.01</p> <p>if type is 1, 3, 4 or 7 epsilon=0.1</p>

The meaning of epsilon is as follows:

if type is 0 or 2: $|f'(w)|_2 \leq \text{epsilon} * \min(\text{pos}, \text{neg})/l * |f'(w_0)|_2$,
where f is the primal function and pos/neg are # of positive/negative data.

if type is 1, 3, 4 or 7: Dual maximal violation $\leq \text{epsilon}$

if type is 5 or 6: $|f'(w)|_\infty \leq \text{epsilon} * \min(\text{pos}, \text{neg})/l * |f'(w_0)|_\infty$,
where f is the primal function

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.
cross	if an integer value $k > 0$ is specified, a k-fold cross validation on data is performed to assess the quality of the model via a measure of the accuracy. Note that this metric might not be appropriate if classes are largely unbalanced. Default is 0.
verbose	if TRUE, information are printed. Default is FALSE.

Details

For details for the implementation of liblinear, see the README file of the original c/c++ liblinear library at <http://www.csie.ntu.edu.tw/~cjlin/liblinear>.

Value

If $\text{cross} > 0$, the average accuracy computed over cross runs of cross-validation is returned.

Otherwise, an object of class "LiblineaR" containing the fitted model is returned, including:

TypeDetail	A string describing the type of model fitted, as determined by type.
Type	An integer corresponding to type.
W	A matrix with the model weights. If bias is TRUE, W contains $p+1$ columns, the last being the bias term. The columns are named according to the names of data, if provided, or "Wx" where "x" ranges from 1 to the number of dimensions. The bias term is named "Bias". If the number of classes is 2, the matrix only has one row. If the number of classes is $k > 2$, it has k rows. Each row i corresponds then to a linear model discriminating between class i and all the other classes. If there are more than 2 classes, rows are named according to the class i which is opposed to the other classes.
Bias	TRUE or FALSE, according to the value of bias
ClassNames	A vector containing the class names

Note

Classification models usually perform better if each dimension of the data is first centered and scaled.

Author(s)

Thibault Helleputte (based on C/C++-code by Chih-Chung Chang and Chih-Jen Lin)
<thelleputte@gmail.com>

References

- For more information on liblinear itself, refer to:
R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin.
LIBLINEAR: A Library for Large Linear Classification,
Journal of Machine Learning Research 9(2008), 1871-1874.
<http://www.csie.ntu.edu.tw/~cjlin/liblinear>

See Also

[predict.LiblineaR,heuristicC](#)

Examples

```
data(iris)
attach(iris)

x=iris[,1:4]
y=factor(iris[,5])
train=sample(1:dim(iris)[1],100)

xTrain=x[train,]
xTest=x[-train,]
yTrain=y[train]
yTest=y[-train]

# Center and scale data
s=scale(xTrain,center=TRUE,scale=TRUE)

# Logistic Regression
t=0

# Find the best model with the best cost parameter via 10-fold cross-validations
tryTypes=c(0:7)
tryCosts=c(1000,100,10,1,0.1,0.01,0.001)
bestCost=NA
bestAcc=0
bestType=NA

for(ty in tryTypes){
  for(co in tryCosts){
    acc=LiblineaR(data=s,labels=yTrain,type=ty,cost=co,bias=TRUE,cross=10,verbose=FALSE)
    cat("Results for C=",co," : ",acc," accuracy.\n",sep="")
    if(acc>bestAcc){
      bestCost=co
      bestAcc=acc
      bestType=ty
    }
  }
}
```

```

}
}
}

cat("Best model type is:",bestType,"\n")
cat("Best cost is:",bestCost,"\n")
cat("Best accuracy is:",bestAcc,"\n")

# Re-train best model with best cost value.
m=LiblineaR(data=s,labels=yTrain,type=bestType,cost=bestCost,bias=TRUE,verbose=FALSE)

# Scale the test data
s2=scale(xTest,attr(s,"scaled:center"),attr(s,"scaled:scale"))

# Make prediction
pr=FALSE
if(bestType==0 | bestType==7) pr=TRUE

p=predict(m,s2,proba=pr,decisionValues=TRUE)

# Display confusion matrix
res=table(p$predictions,yTest)
print(res)

# Compute Balanced Classification Rate
BCR=mean(c(res[1,1]/sum(res[,1]),res[2,2]/sum(res[,2]),res[3,3]/sum(res[,3])))
print(BCR)

```

predict.LiblineaR *Classification Predictions With Liblinear*

Description

The function takes a model produced by the LiblineaR function and a data matrix and classifies every row of the matrix as belonging to one of the classes considered by the model.

Usage

```
## S3 method for class 'LiblineaR'
predict(object, newx, proba = FALSE, decisionValues = FALSE, ...)
```

Arguments

object	Object of class "LiblineaR", created by LiblineaR.
newx	An n x p matrix containing the new input data. A vector will be transformed to a n x 1 matrix.
proba	Logical indicating whether class probabilities should be computed and returned. Only possible if the model was fitted with type=0, type=6 or type=7, i.e. a Logistic Regression. Default is FALSE.

decisionValues Logical indicating whether model decision values should be computed and returned. Default is FALSE.
... Currently not used.

Value

By default, the returned value is a list with a single entry:

predictions A vector of predicted labels.

If proba is set to TRUE, and the fitted model is a logistic regression, an additional entry is returned:

probabilities An $n \times k$ matrix (k number of classes) of the class probabilities. The columns of this matrix are named after class labels.

If decisionValues is set to TRUE, an additional entry is returned:

decisionValues An $n \times k$ matrix (k number of classes) of the model decision values. The columns of this matrix are named after class labels.

Note

If the data on which the model has been fitted have been centered and/or scaled, it is very important to apply the same process on the new data as well, with the scale and center values of the training data.

Author(s)

Thibault Helleputte
<thelleputte@gmail.com>

References

- For more information on liblinear itself, refer to:
R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin.
LIBLINEAR: A Library for Large Linear Classification,
Journal of Machine Learning Research 9(2008), 1871-1874.
<http://www.csie.ntu.edu.tw/~cjlin/liblinear>

See Also

[LiblineaR](#)

Examples

```
data(iris)
attach(iris)

x=iris[,1:4]
y=factor(iris[,5])
train=sample(1:dim(iris)[1],100)
```

```

xTrain=x[train,]
xTest=x[-train,]
yTrain=y[train]
yTest=y[-train]

# Center and scale data
s=scale(xTrain,center=TRUE,scale=TRUE)

# Logistic Regression
t=0

# Tune the cost parameter of a logistic regression via a 10-fold cross-validation
try=c(1000,100,10,1,0.1,0.01,0.001)
res=c()
for(co in try){
acc=LiblineaR(data=s,labels=yTrain,type=t,cost=co,bias=TRUE,cross=10,verbose=FALSE)
res=c(res,acc)
cat("Results for C=",co," : ",acc," accuracy.\n",sep="")
}

# Re-train a model with best C value.
best=which.max(res)
co=try[best]
m=LiblineaR(data=s,labels=yTrain,type=t,cost=co,bias=TRUE,verbose=FALSE)

# Scale the test data
s2=scale(xTest,attr(s,"scaled:center"),attr(s,"scaled:scale"))

# Make prediction
p=predict(m,s2,proba=TRUE)

# Display confusion matrix
res=table(p$predictions,yTest)
print(res)

# Compute Balanced Classification Rate
BCR=mean(c(res[1,1]/sum(res[,1]),res[2,2]/sum(res[,2]),res[3,3]/sum(res[,3])))
print(BCR)

# Plot a ROC curve for class setosa against the rest:
## Not run: library(ROCR)
## Not run: s3=scale(x,center=TRUE,scale=TRUE)
## Not run: m=LiblineaR(data=s3,labels=y,type=0,cost=co,bias=TRUE,verbose=FALSE)
## Not run: p=predict(m,s3,proba=TRUE)
## Not run: lab=as.matrix(y)
## Not run: lab[which(lab!="versicolor")]=1
## Not run: lab[which(lab=="versicolor")]=-1
## Not run: pre=prediction(predictions=p$probabilities["versicolor"],labels=lab)
## Not run: per=performance(pre,"tpr","fpr")
## Not run: AUC=(performance(pre,"auc"))@y.values[[1]]
## Not run: subtitle=paste("AUC:",AUC)
## Not run: plot(per,main="ROC Curve: Versicolor against Setosa and Virginica",xlab="True Positive Rate",ylab="False Positive Rate")

```

Index

*Topic **classes**

LiblineaR, 5
LiblineaR-package, 2
predict.LiblineaR, 8

*Topic **classif**

heuristicC, 3
LiblineaR, 5
LiblineaR-package, 2
predict.LiblineaR, 8

*Topic **models**

LiblineaR, 5
LiblineaR-package, 2
predict.LiblineaR, 8

*Topic **multivariate**

LiblineaR, 5
LiblineaR-package, 2
predict.LiblineaR, 8

*Topic **optimize**

LiblineaR, 5
LiblineaR-package, 2
predict.LiblineaR, 8

*Topic **package**

LiblineaR-package, 2

heuristicC, 3, 7

Joachims Heuristics (heuristicC), 3

LiblineaR, 4, 5, 9

LiblineaR-package, 2

predict.LiblineaR, 7, 8