Package 'regtools'

November 7, 2016

Version 1.0.1

Title Regression Tools

Author Norm Matloff

Maintainer Norm Matloff <matloff@cs.ucdavis.edu>

Depends FNN,mvtnorm,dummies,car

Suggests knitr, rmarkdown

VignetteBuilder knitr

License GPL (>= 2)

Description Tools for linear, nonlinear and nonparametric regression and classification. Parametric fit assessment using nonparametric methods. One vs. All and All vs. All multiclass classification. Nonparametric regression for general dimension, locally-linear option. Nonlinear regression with Eickert-White method for dealing with heteroscedasticity, k-NN for general dimension and general descriptive functions.

NeedsCompilation no

Repository CRAN

Date/Publication 2016-11-07 21:07:44

R topics documented:

`avalogtrn,avalogpred,ovalogtrn,ovalogpred,knntrn,predict.ovaknn`
Classification with More Than 2 Classes

Description

One vs. All, All vs. All tools for multiclass classification, parametric and nonparametric.

Usage

```
ovalogtrn(m,trnxy,truepriors=NULL)
ovalogpred(coefmat,predx)
avalogtrn(m,trnxy)
avalogpred(m,coefmat,predx)
knntrn(y,xdata,m,k,truepriors=NULL)
## S3 method for class 'ovaknn'
predict(object,...)
classadjust(econdprobs,wrongratio,trueratio)
```

Arguments

`trnxy`	Data matrix, one data point per row, Y in the last column.
`object`	Needed for consistency with generic.
`...`	Needed for consistency with generic.
`y`	Vector of response variable data in the training set, with codes 0,1,...m-1.
`xdata`	X and associated neighbor indices. Output of `preprocessx`.
`coefmat`	Output from `ovalogtrn`.
`k`	Number of nearest neighbors.
`predx`	One data point to be predicted.
`m`	Number of classes in multiclass setting.
`econdprobs`	Estimated conditional class probabilities, given the predictors.
`wrongratio`	Incorrect, data-provenanced, p/(1-p), with p being the unconditional probability of a certain class.
`trueratio`	Same as `wrongratio`, but with the correct value.
`truepriors`	True unconditional class probabilities, typically obtained externally.

Details

These functions do classification in the multiclass setting, using the One vs. All method. In addition to logit, the k-Nearest Neighbor method is available. For this, `preprocessx` must first be called. In the logit case, All vs. All is also offered.

The functions `ovalogtrn`, `avalogtrn` and `knntrn` are used on the training set, and then fed into the prediction functions, `ovalogpred`, `avalogpred` and `predict.ovaknn`. The arguments for the latter are the output of `knntrn` and a matrix of prediction points (internally referred to as `predpts` in the code), one per row.

Value

The prediction functions, `ovalogpred`, `avalogpred` and `predict.ovaknn`, return the predicted classes for the points in `predx` or `predpts`.

The functions `ovalogtrn` and `avalogtrn` return the estimated logit coefficent vectors, one per column. There are `m` of them in the former case, `mm-1/2` in the latter, in which case the order of the R function `combin` is used.

The function `knntrn` returns a copy of the `xdata` input, but with an extra component added. The latter is the matrix of estimated regression function values; the element in row i, column j, is the probability that Y = j given that X = row i in the X data.

Author(s)

Norm Matloff

Examples

```
## Not run:
# toy example, kNN
set.seed <- 9999
x <- runif(25)
y <- sample(0:2,25,replace=TRUE)
xd <- preprocessx(x,2)
kout <- knntrn(y,xd,m=3,k=2)
kout$regest  # row 1:  0.5,0.5,0.0
predict(kout,matrix(c(0.25,0.55,0.88),ncol=1))  # 2,0,0

# sim data, kNN
set.seed <- 9999
n <- 1500
# within-grp cov matrix
cv <- rbind(c(1,0.2),c(0.2,1))
xy <- NULL
for (i in 1:3)
   xy <- rbind(xy,rmvnorm(n,mean=rep(i*2.0,2),sigma=cv))
y <- rep(0:2,each=n)
xy <- cbind(xy,y)
xdata <- preprocessx(xy[,-3],20)
oo <- knntrn(y,xdata,m=3,k=20)
predout <- predict(oo,xy[,-3])
mean(predout$predy == y)  # about 0.87

\dontrun{
library(mlbench)
data(Vehicle)
xdata <- preprocessx(Vehicle[,-19],25)
kout <- knntrn(Vehicle$Class,xdata,k=25)
predict(kout,as.matrix(Vehicle[1,-19])) # predicted Y is 3

# UCI Letter Recognition data
data(LetterRecognition)
```

```
# prep data
lr <- LetterRecognition
# code Y values
lr[,1] <- as.numeric(lr[,1]) - 1
# training and test sets
lrtrn <- lr[1:14000,]
lrtest <- lr[14001:20000,]
# kNN
xdata <- preprocessx(lrtrn[,-1],50)
# without setting priors
trnout <- knntrn(lrtrn[,1],xdata,26,50)
ypred <- predict(trnout,as.matrix(lrtest[,-1]))
# how well did it work?
mean(ypred$predy == lrtest[,1])  # 0.86
# logit
ologout <- ovalogtrn(26,lr[,c(2:17,1)])
ypred <- ovalogpred(ologout,lr[,-1])
mean(ypred == lr[,1])  # only 0.73
# try quadratic terms
for (i in 2:17)
   lr <- cbind(lr,lr[,i]^2)
ologout1 <- ovalogtrn(26,lr[,c(2:33,1)])
ypred <- ovalogpred(ologout1,lr[,-1])
mean(ypred == lr[,1])  # increased to 0.81
}


## End(Not run)
```

---

knnest,meany,vary,loclin,predict.knn,preprocessx,kmin,parvsnonparplot,nonparvsxplot,l1,l2

Nonparametric Regression and Classification

Description

Full set of tools for k-NN regression and classification, including both for direct usage and as tools for assessing the fit of parametric models.

Usage

```
knnest(y,xdata,k,nearf=meany)
preprocessx(x,kmax,xval=FALSE)
meany(predpt,nearxy)
vary(predpt,nearxy)
loclin(predpt,nearxy)
## S3 method for class 'knn'
predict(object,...)
kmin(y,xdata,lossftn=l2,nk=5,nearf=meany)
```

```
parvsnonparplot(lmout,knnout,cex=1.0)
nonparvsxplot(knnout,lmout=NULL)
nonparvarplot(knnout)
l2(y,muhat)
l1(y,muhat)
```

Arguments

y	Response variable data in the training set. Vector or matrix, the latter case for vector-valued response, e.g. multiclass classification.
x	X data, predictors, one row per data point, in the training set.
...	Needed for consistency with generic. See Details below for 'arguments.
xdata	X and associated neighbor indices. Output of preprocessx.
k	Number of nearest neighbors
object	Output of knnest.
predpt	One point on which to predict, as a vector.
nearxy	A set of X neighbors of a point.
nearf	Function to apply to the nearest neighbors of a point.
kmax	Maximal number of nearest neighbors to find.
xval	Cross-validation flag. If TRUE, then the set of nearest neighbors of a point will not include the point itself.
lossftn	Loss function to be used in cross-validation determination of "best" k.
nk	Number of values of k to try in cross-validation.
lmout	Output of lm.
knnout	Output of knnest.
cex	R parameter to control dot size in plot.
muhat	Vector of estimated regression function values.

Details

The knnest function does k-nearest neighbor regression function estimation, in any dimension, i.e. any number of predictor variables, and any number of response variables. This of course includes classification problems case; a scalar Y = 0,1 would represent two classes, with the regression function reducing to the conditional probability of class 1, given the predictors.

The preprocessx function does the prep work. For each row in x, the code finds the kmax closest rows to that row. By separating this computation from knnest, one can save a lot of overall computing time. If for instance one wants to try the number of nearest neighbors k at 25, 50 and 100, one can call preprocessx with kmax equal to 100, then reuse the results; in calling knnest for several values of k, we do not need to call preprocessx again.

In addition to averaging the neighbor Y values, one can specify other types of smoothing by proper specification of the nearf function. For instance, one specifies local linear smoothing by setting nearf to loclin, one could check heteroscedasticity in linear regression models by setting nearf to vary to estimate conditional variance.

The X, i.e. predictor, data will be scaled by the code, so as to put all predictor variables on an equal footing. The scaling parameters will be recorded, and then applied later in prediction.

One can choose the number of nearest neighbors k through the kmin function.

The function predict.knn uses the output of knnest to do estimation or prediction on new points. Since the output of knnest is of class "knn", one invokes this function with the simpler predict. The second argument is the set of new points, supplied as a matrix, called predpts within the code. A "1-NN" method is used here: Given a new point u whose "Y" value we wish to predict, the code finds the single closest row in the training set, and returns the previously-estimated regression function value at that row.

The functions ovaknntrn and ovaknnpred are multiclass wrappers for knnest and knnpred. Here y is coded 0,1,...,m-1 for the m classes.

The tools here can be useful for fit assessment of parametric models. The parvsnonparplot function plots fitted values of parameteric model vs. kNN fitted, nonparvsxplot k-NN fitted values against each predictor, one by one.

The functions l2 and l1 are used to define L2 and L1 loss.

Value

The return value of preprocessx is an R list. Its x component is the scaled x matrix, with the scaling factors being recorded in the scaling component. The idxs component contains the indices of the nearest neighbors of each point in the predictor data, stored in a matrix with nrow(x) rows and k columns. Row i contains the indices of the nearest rows in x to row i of x. The first of these indices is for the closest point, then for the second-closest, and so on. If cross-validation is requrested (xval = TRUE, then any point will not be considered a neighbor of itself.

The knnest function returns an expanded version of xdata, with the expansion consisting of a new component regest, the estimated regression function values at the training set points.

The function predict.knn returns the predicted Y values at predpts. It is called simply via predict.

The function kmin returns an R list, with the component meanerrs containing the cross-validated mean loss function values and ks containing the corresponding values of k; plot.knn then plots the former against the latter.

Author(s)

Norm Matloff

Examples

```
set.seed(9999)
x <- matrix(sample(1:100,30),ncol=3)
xd <- preprocessx(x[,1],2,TRUE)  # just 1 predictor
ko <- knnest(x[,2],xd,2)  # Y is x[,2]
ko$regest # 1st element = 74.5
predict(ko,matrix(76))  # 47.5
ko <- knnest(x[,-1],xd,2)  # Y bivar
ko$regest # 1st row = (74.5,31.5)
predict(ko,matrix(76))  # 47.5, 65.0
```

```
xe <- matrix(rnorm(30000),ncol=3)
xe[,-3] <- xe[,-3] + 2
# xe is 2 predictors and epsilon
y <- xe %*% c(1,0.5,0.2)  # Y
x <- xe[,-3]  # X
xdata <- preprocessx(x,500)  # k as high as 500
zout <- knnest(y,xdata,200)
predict(zout,matrix(c(1,1),nrow=1))  # about 1.55

## Not run:
data(prgeng)
pe <- prgeng
# dummies for MS, PhD
pe$ms <- as.integer(pe$educ == 14)
pe$phd <- as.integer(pe$educ == 16)
# computer occupations only
pecs <- pe[pe$occ >= 100 & pe$occ <= 109,]
# for simplicity, let's choose a few predictors
pecs1 <- pecs[,c(1,7,9,12,13,8)]
# will predict wage income from age, gender etc.
# prepare nearest-neighbor data, k up to 50
xdata <- preprocessx(pecs1[,1:5],50)
zout <- knnest(pecs1[,6],xdata,50)  # k = 50
# find the est. mean income for 42-year-old women, 52 weeks worked, with
# a Master's
predict(zout,matrix(c(42,2,52,0,0),nrow=1))  # 62106
# try k = 25; don't need to call preprocessx() again
zout <- knnest(pecs1[,6],xdata,25)
predict(zout,matrix(c(42,2,52,0,0),nrow=1))  # 69104
# quite a difference; what k values are good?
kmin(pecs1[,6],xdata) # at least 50
# what about a man?
zout <- knnest(pecs1[,6],xdata,50)
predict(zout,matrix(c(42,1,52,0,0),nrow=1))  # 78588
# form training and test sets, fit on the former and predict on the
# latter
fullidxs <- 1:nrow(pecs1)
train <- sample(fullidxs,10000)
xdata <- preprocessx(pecs1[train,1:5],50)
trainout <- knnest(pecs1[train,6],xdata,50)
testout <- predict(trainout,as.matrix(pecs1[-train,-6]))
# find mean abs. prediction error (about $25K)
mean(abs(pecs1[-train,6] - testout))
# examples of fit assessment
# look for nonlinear relations between Y and each X
nonparvsxplot(zout)  # keep hitting Enter for next plot
# there seem to be quadratic relations with age and wkswrkd, so add quad
# terms and run lm()
pecs2 <- pecs1
pecs2$age2 <- pecs1$age^2
pecs2$wks2 <- pecs1$wkswrkd^2
lmout2 <- lm(wageinc ~ .,data=pecs2)
```

```
# check parametric fit by comparing to kNN
parvsnonparplot(lmout2,zout)
# linear model line somewhat faint, due to large n;
# parametric model seems to overpredict at high end;
# to deal with faintness, reduce size of points
parvsnonparplot(lmout2,zout,cex=0.1)
# assess homogeneity of conditional variance
nonparvarplot(zout)
# hockey stick!

## End(Not run)

# Y vector-valued (3 classes)
# 3 clusters, equal wts, coded 0,1,2
n <- 1500
# within-grp cov matrix
cv <- rbind(c(1,0.2),c(0.2,1))
xy <- NULL
for (i in 1:3)
   xy <- rbind(xy,rmvnorm(n,mean=rep(i*2.0,2),sigma=cv))
y <- rep(0:2,each=n)
xy <- cbind(xy,dummy(y))
xdata <- preprocessx(xy[,-(3:5)],20) # X is xy[,1:2], k <= 20
ko <- knnest(xy[,3:5],xdata,20)
# find predicted Y for each data pt
mx <- apply(as.matrix(ko$regest),1,which.max) - 1
# overall correct classification rate
mean(mx == y)  # should be about 0.87
```

`lmac,makeNA,coef.lmac,vcov.lmac,pcac,loglinac,tbltofakedf`
Available Cases Method for Missing Data

Description

Various estimators that handle missing data via the Available Cases Method

Usage

```
lmac(xy,nboot=0)
makeNA(m,probna)
## S3 method for class 'lmac'
coef(object,...)
## S3 method for class 'lmac'
vcov(object,...)
pcac(indata,scale=FALSE)
loglinac(x,margin)
tbltofakedf(tbl)
```

Arguments

xy	Matrix or data frame, X values in the first columns, Y in the last column.
indata	Matrix or data frame.
x	Matrix or data frame, one column per variable.
nboot	If positive, number of bootstrap samples to take.
probna	Probability that an element will be NA.
scale	If TRUE, call cor instead of cov.
	Matrix or data frame, one column per variable.
tbl	An R table.
m	Number of synthetic NAs to insert.
object	Output from lmac.
...	Needed for consistency with generic function. Not used.
margin	A list of vectors specifying the model, as in loglin.

Details

The Available Cases (AC) approach applies to statistical methods that depend only on products of k of the variables, so that cases having non-NA values for those k variables can be used, as opposed to using only cases that are fully intact in all variables, the Complete Cases (CC) approach. In the case of linear regression, for instance, the estimated coefficients depend only on covariances between the variables (both predictors and response). This approach assumes thst the cases with missing values have the same distribution as the intact cases.

The lmac function forms OLS estimates as with lm, but applying AC, in contrast to lm, which uses the CC method.

The pcac function is an AC substitute for prcomp. The data is centered, corresponding to a fixed value of center = TRUE in prcomp. It is also scaled if scale is TRUE, corresponding scale. = TRUE in prcomp;. Due to AC, there is a small chance of negative eigenvalues, in which case stop will be called.

The loglinac function is an AC substitute for loglin. The latter takes tables as input, but loglinac takes the raw data. If you have just the table, use tbltofakedf to regenerate a usable data frame.

The makeNA function is used to insert random NA values into data, for testing purposes.

Value

For lmac, an object of class 'lmac', with components

- coefficients, as with lm; accessible directly or by calling coef, as with lm
- fitted.values, as with lm
- residuals, as with lm
- r2, (unadjusted) R-squared
- cov, for nboot >0 the estimated covariance matrix of the vector of estimated regression coefficients; accessible directly or by calling vcov, as with lm

For pcac, an R list, with components

- sdev, as with prcomp
- rotation, as with prcomp

For loglinac, an R list, with components

- param, estimated coefficients, as in loglin
- fit, estimated expected call counts, as in loglin

Author(s)

Norm Matloff

Examples

```
n <- 25000
w <- matrix(rnorm(2*n),ncol=2)  # x and epsilon
x <- w[,1]
y <- x + w[,2]
# insert some missing values
nmiss <- round(0.1*n)
x[sample(1:n,nmiss)] <- NA
nmiss <- round(0.2*n)
y[sample(1:n,nmiss)] <- NA
acout <- lmac(cbind(x,y))
coef(acout)  # should be near pop. values 0 and 1
```

---

ltrfreqs *Letter Frequencies*

Description

This is data consists of capital letter frequencies obtained at http://www.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.h tml

Usage

```
data(ltrfreqs); ltrfreqs
```

mm *Method of Moments, Including Possible Regression Terms*

Description

Method of Moments computation for almost any statistical problem that has derivatives with respect to theta. Capable of handling models that include parametric regression terms, but not need be a regression problem. (This is not *Generalized* Method of Moments; see the package **gmm** for the latter.)

Usage

```
mm(m,g,x,init=rep(0.5,length(m)),eps=0.0001,maxiters=1000)
```

Arguments

m	Vector of sample moments, "left-hand sides" of moment equations.
g	Function of parameter estimates, forming the "right-hand sides." This is a multivariate-valued function, of dimensionality equal to that of m.
init	Vector of initial guesses for parameter estimates. If components are named, these will be used as labels in the output.
eps	Convergence criterion.
maxiters	Maximum number of iterations.
x	Input data.

Details

Standard Newton-Raphson methods are used to solve for the parameter estimates, with numericDeriv being used to find the approximate derivatives.

Value

R list consisting of components tht, the vector of parameter estimates, and numiters, the number of iterations performed.

Author(s)

Norm Matloff

Examples

```
x <- rgamma(1000,2)
m <- c(mean(x),var(x))
g <- function(x,theta) {  # from theoretical properties of gamma distr.
   g1 <-  theta[1] / theta[2]
   g2 <-  theta[1] / theta[2]^2
   c(g1,g2)
```

```
}
# should output about 2 and 1
mm(m,g,x)

## Not run:
library(mfp)
data(bodyfat)
# model as a beta distribution
g <- function(x,theta) {
   t1 <- theta[1]
   t2 <- theta[2]
   t12 <- t1 + t2
   meanb <- t1 / t12
   m1 <- meanb
   m2 <- t1*t2 / (t12^2 * (t12+1))
   c(m1,m2)
}
x <- bodyfat$brozek/100
m <- c(mean(x),var(x))
# about 4.65 and 19.89
mm(m,g,x)

## End(Not run)
```

`nlshc` *Heteroscedastic Nonlinear Regression*

Description

Extension of `nls` to the heteroscedastic case.

Usage

```
nlshc(nlsout,type='hc3')
```

Arguments

`nlsout`	Object of type 'nls'.
`type`	Eickert-White algorithm to use. See documentation for **nls**.

Details

Calls `nls` but then forms a different estimated covariance matrix for the estimated regression coefficients, applying the Eickert-White technique to handle heteroscedasticity. This then gives valid statistical inference in that setting.

Some users may prefer to use `nlsLM` of the package **minpack.lm** instead of `nls`. This is fine, as both functions return objects of class 'nls'.

Value

Estimated covariance matrix

Author(s)

Norm Matloff

References

Zeileis A (2006), Object-Oriented Computation of Sandwich Estimators. *Journal of Statistical Software*, **16**(9), 1–16, http://www.jstatsoft.org/v16/i09/.

Examples

```
# simulate data from a setting in which mean Y is
# 1 / (b1 * X1 + b2 * X2)
n <- 250
b <- 1:2
x <- matrix(rexp(2*n),ncol=2)
meany <- 1 / (x %*% b)  # reg ftn
y <- meany + (runif(n) - 0.5) * meany  # heterosced epsilon
xy <- cbind(x,y)
xy <- data.frame(xy)
# see nls() docs
nlout <- nls(X3 ~ 1 / (b1*X1+b2*X2),
   data=xy,start=list(b1 = 1,b2=1))
nlshc(nlout)
```

---

prgeng *Silicon Valley programmers and engineers*

Description

This data set is adapted from the 2000 Census (5% sample, person records). It is restricted to programmers and engineers in the Silicon Valley area.

The variable codes, e.g. occupational codes, are available from the Census Bureau, at http://www.census.gov/prod/cen2000/doc/pums.pdf. (Short code lists are given in the record layout, but longer ones are in the appendix Code Lists.)

The variables are:

- age, with a U(0,1) variate added for jitter
- cit, citizenship; 1-4 code various categories of citizens; 5 means noncitizen (including permanent residents
- educ: 01-09 code no college; 10-12 means some college; 13 is a bachelor's degree, 14 a master's, 15 a professiona deal and 16 is a doctorate
- occ, occupation

- birth, place of birth
- wageinc, wage income
- wkswrkd, number of weeks worked
- yrentry, year of entry to the U.S. (0 for natives)
- powpuma, location of work
- gender, 1 for male, 2 for female

Usage

```
data(prgeng); prgeng
```

---

ridgelm,plot.rlm *Ridge Regression*

Description

Similar to lm.ridge in MASS packaged included with R, but with a different kind of scaling and a little nicer plotting.

Usage

```
ridgelm(xy,lambda = seq(0.01,1,0.01),mapback=TRUE)
## S3 method for class 'rlm'
plot(x,y,...)
```

Arguments

xy	Data, response variable in the last column.
lambda	Vector of desired values for the ridge parameter.
mapback	If TRUE, the scaling that had been applied to the original data will be map back to the original scale, so that the estimated regression coefficients are now on the scale of the original data.
x	Object of type 'rlm', output of ridgelm.
y	Needed for consistency with the generic. Not used.
...	Needed for consistency with the generic. Not used.

Details

Centers and scales the predictors X, and centers the response variable Y. Computes X'X and then solves [(X'X)/n + lambda I]b = X'Y/n for b. The 1/n factors are important, making the diagonal elements of (X'X)/n all 1s and thus facilitating choices for the lambdas in a manner independent of the data.

Calling plot on the output of ridgelm dispatches to plot.rlm, thus diplaying the ridge traces.

Value

The function `ridgelm` returns an object of class 'rlm', with components `bhats`, the estimated beta vectors, one column per lambda value, and `lambda`, a copy of the input.

Author(s)

Norm Matloff

Index