Package ‘dma’

July 22, 2025

Type Package

Title Dynamic Model Averaging

Version 1.4-0

Date 2018-10-04

Author Tyler H. McCormick, Adrian Raftery, David Madigan, Sevvandi Kandanaarachchi [ctb], Hana Sevcikova [ctb]

Maintainer Hana Sevcikova <hanas@uw.edu>

Description Dynamic model averaging for binary and continuous outcomes.

Imports MASS

Suggests testthat

License GPL-2

NeedsCompilation no

Repository CRAN

Date/Publication 2018-10-05 17:30:03 UTC

Contents

`dma-package` *Dynamic model averaging*

Description

This package implements dynamic Bayesian model averaging as described for continuous outcomes in Raftery et al. (2010, Technometrics) and for binary outcomes in McCormick et al. (2011, Biometrics).

Details

Package:	dma
Type:	Package
Version:	1.4-0
Date:	2018-10-4
License:	GPL2

Author(s)

Tyler H. McCormick, Adrian Raftery, David Madigan, Sevvandi Kandanaarachchi [ctb], Hana Sevcikova [ctb]

Maintainer: Hana Sevcikova <hanas@uw.edu>

References

McCormick, T.M., Raftery, A.E., Madigan, D. and Burd, R.S. (2011) "Dynamic Logistic Regression and Dynamic Model Averaging for Binary Classification." Biometrics, 66:1162-1173.

Raftery, A.E., Karny, M., and Ettler, P. (2010). Online Prediction Under Model Uncertainty Via Dynamic Model Averaging: Application to a Cold Rolling Mill. Technometrics 52:52-66.

`dma` *Dynamic model averaging for continuous outcomes*

Description

Implemtent dynamic model averaging for continuous outcomes as described in Raftery, A.E., Karny, M., and Ettler, P. (2010). Online Prediction Under Model Uncertainty Via Dynamic Model Averaging: Application to a Cold Rolling Mill. Technometrics 52:52-66. Along with the values described below, plot() creates a plot of the posterior model probabilities over time and model-averaged fitted values and print() returns model matrix and posterior model probabilities. There are TT time points, K models, and d total covariates.

Usage

```
dma(x, y, models.which, lambda=0.99, gamma=0.99,
 eps=.001/nrow(models.which), delay=0, initialperiod=200)
```

Arguments

- `x` TTxd matrix of system inputs
- `y` TT-vector of system outputs
- `models.which` Kxd matrix, with 1 row per model and 1 col per variable indicating whether that variable is in the model (the state theta is of dim (model.dim+1); the extra 1 for the intercept)
- `lambda` parameter forgetting factor
- `gamma` flatterning parameter for model updating
- `eps` regularization parameter for regularizing posterior model model probabilities away from zero
- `delay` When y_t is controlled, only y_t-delay-1 and before are available. This is determined by the machine. Note that delay as defined here corresponds to (k-1) in the Ettler et al (2007, MixSim) paper. Thus k=25 in the paper corresponds to delay=24.
- `initialperiod` length of initial period. Performance is summarized with and without the first initialperiod samples.

Value

- `yhat.bymodel` TTxK matrix whose tk element gives yhat for yt for model k
- `yhat.ma` TT vector whose t element gives the model-averaged yhat for yt
- `pmp` TTxK matrix whose tk element is the post prob of model k at t
- `thetahat` KxTTx(nvar+1) array whose ktj element is the estimate of theta_j-1 for model k at t
- `Vtheta` KxTTx(nvar+1) array whose ktj element is the variance of theta_j-1 for model k at t
- `thetahat.ma` TTx(nvar+1) matrix whose tj element is the model-averaged estimate of theta_j-1 at t
- `Vtheta.ma` TTx(nvar+1) matrix whose tj element is the model-averaged variance of thetahat_j-1 at t
- `mse.bymodel` MSE for each model
- `mse.ma` MSE of model-averaged prediction
- `mseinitialperiod.bymodel` MSE for each model excluding the first initialperiod samples
- `mseinitialperiod.ma` MSE of model averaging excluding the first initialperiod samples
- `model.forget` forgetting factor for the model switching matrix

Author(s)

Adrian Raftery, Tyler H. McCormick

References

Raftery, A.E., Karny, M., and Ettler, P. (2010). Online Prediction Under Model Uncertainty Via Dynamic Model Averaging: Application to a Cold Rolling Mill. Technometrics 52:52-66.

Examples

```
#simulate some data to test
#first, static coefficients
coef<-c(1.8,3.4,-2,3,-2.8,3)
coefmat<-cbind(rep(coef[1],200),rep(coef[2],200),
            rep(coef[3],200),rep(coef[4],200),
            rep(coef[5],200),rep(coef[6],200))
#then, dynamic ones
coefmat<-cbind(coefmat,seq(1,2.45,length.out=nrow(coefmat)),
            seq(-.75,-2.75,length.out=nrow(coefmat)),
            c(rep(-1.5,nrow(coefmat)/2),rep(-.5,nrow(coefmat)/2)))
npar<-ncol(coefmat)-1
dat<-matrix(rnorm(200*(npar),0,1),200,(npar))
ydat<-rowSums((cbind(rep(1,nrow(dat)),dat))[1:100,]*coefmat[1:100,])
ydat<-c(ydat,rowSums((cbind(rep(1,nrow(dat)),dat)*coefmat)[-c(1:100),c(6:9)]))
mmat<-matrix(c(c(1,0,1,0,0,rep(1,(npar-7)),0,0),
            c(rep(0,(npar-4)),rep(1,4)),rep(1,npar)),3,npar,byrow=TRUE)
dma.test<-dma(dat,ydat,mmat,lambda=.99,gamma=.99,initialperiod=20)
plot(dma.test)
```

`logistic.dma` *Dynamic model averaging for binary outcomes*

Description

Implements dynamic model averaging for continuous outcomes as described in McCormick et al. (2011, Biometrics). It can be either performed for all data at once (using `logistic.dma`), or dynamically for one observation at a time (combining the remaining functions, see Example). Along with the values described below, plot() creates a plot of the posterior model probabilities over time and model-averaged fitted values (with smooth curve overlay) and print() returns model matrix and posterior model probabilities. There are K candidate models, T time points, and d total covariates (including the intercept).

Usage

```
logistic.dma(x, y, models.which, lambda = 0.99, alpha = 0.99,autotune = TRUE,
    initmodelprobs = NULL, initialsamp = NULL)

logdma.init(x, y, models.which)

logdma.predict(fit, newx)

logdma.update(fit, newx, newy, lambda = 0.99, autotune = TRUE)
```

```
logdma.average(fit, alpha = 0.99, initmodelprobs = NULL)
```

Arguments

`x`	T by (d-1) matrix of observed covariates. Note that a column of 1's is added automatically for the intercept. In `logdma.init`, this matrix contains only the training set.
`y`	T vector of binary responses. In `logdma.init`, these correspond to the training set only.
`models.which`	K by (d-1) matrix defining models. A 1 indicates a covariate is included in a particular model, a 0 if it is excluded. Model averaging is done over all modeld specified in models.which.
`lambda`	scalar forgetting factor with each model
`alpha`	scalar forgetting factor for model averaging
`autotune`	T/F indicates whether or not the automatic tuning procedure desribed in McCormick et al. should be applied. Default is true.
`initmodelprobs`	K vector of starting probabilities for model averaging. If null (default), then use 1/K for each model.
`initialsamp`	scalar indicating how many observations to use for generating initial values. If null (default), then use the first 10 percent of observations.
`newx, newy`	Subset of `x` and `y` corresponding to new observations.
`fit`	List with estimation results that are outputs of functions `logdma.init`, `logdma.update` and `logdma.average`.

Details

The function `logistic.dma` is composed of three parts, which can be also used separately: First, the model is trained with a subset of the data (function `logdma.init`), where the size of the training set is determined by `initialsamp`. Note that arguments `x` and `y` in `logdma.init` should contain the training subset only. Then, the estimation is updated with new observations (function `logdma.update`). Lastly, a dynamic model averaging is performed on the final estimates (function `logdma.average`). The updating, averaging and in addition predicting (`logdma.predict`) can be performed dynamically for one observation at a time, see Example below.

Value

Functions `logistic.dma` and `logdma.average` return an object of class `logistic.dma`. Functions `logdma.init` and `logdma.update` return a list with estimation results which is a subset of the `logistic.dma` object. It has the following components:

`x`	T by (d-1) matrix of covariates
`y`	T by 1 vector of binary responses
`models.which`	K by (d-1) matrix of candidate models
`lambda`	scalar, tuning factor within models

alpha	scalar, tuning factor for model averaging
autotune	T/F, indicator of whether or not to use autotuning algorithm
alpha.used	T vector of alpha values used
theta	K by T by d array of dynamic logistic regression estimates for each model
vartheta	K by T by d array of dynamic logistic regression variances for each model
pmp	K by T array of posterior model probabilities
yhatdma	T vector of model-averaged predictions
yhatmodel	K by T vector of fitted values for each model

Function `logdma.predict` returns a matrix with predictions corresponding to the `newx` covariates.

Author(s)

Tyler H. McCormick, David Madigan, Adrian Raftery

Sevvandi Kandanaarachchi and Hana Sevcikova implemented the "streaming" functionality, i.e. the original function was decomposed into standalone parts that can be used separately for one observation at a time.

References

McCormick, T.M., Raftery, A.E., Madigan, D. and Burd, R.S. (2011) "Dynamic Logistic Regression and Dynamic Model Averaging for Binary Classification." Biometrics, 66:1162-1173.

Examples

```
# simulate some data to test
# first, static coefficients
coef <- c(.08,-.4,-.1)
coefmat <- cbind(rep(coef[1],200),rep(coef[2],200),rep(coef[3],200))
# then, dynamic ones
coefmat <- cbind(coefmat,seq(1,.45,length.out=nrow(coefmat)),
            seq(-.75,-.15,length.out=nrow(coefmat)),
            c(rep(-1.5,nrow(coefmat)/2),rep(-.5,nrow(coefmat)/2)))
npar <- ncol(coefmat)-1

# simulate data
set.seed(1234)
dat <- matrix(rnorm(200*(npar),0,1),200,(npar))
ydat <- exp(rowSums((cbind(rep(1,nrow(dat)),dat))[1:100,]*coefmat[1:100,]))/
          (1+exp(rowSums(cbind(rep(1,nrow(dat)),dat)[1:100,]*coefmat[1:100,])))
y <- c(ydat,exp(rowSums(cbind(rep(1,nrow(dat)),dat)[-c(1:100),c(1,5,6)]*
               coefmat[-c(1:100),c(1,5,6)]))/
          (1+exp(rowSums(cbind(rep(1,nrow(dat)),dat)[-c(1:100),c(1,5,6)]*
               coefmat[-c(1:100),c(1,5,6)]))))
u <- runif (length(y))
y <- as.numeric (u < y)

# Consider three candidate models
mmat <- matrix(c(1,1,1,1,1,0,0,0,1,1,1,0,1,0,1),3,5, byrow = TRUE)
```

```
# Fit model and plot
# autotuning is turned off for this demonstration example
ldma.test <- logistic.dma(dat, y, mmat, lambda = .99, alpha = .99,
    autotune = FALSE, initialsamp = 20)
plot(ldma.test)

# Using DMA in a "streaming" mode
modl <- logdma.init(dat[1:20,], y[1:20], mmat)
yhat <- matrix(0, ncol=3, nrow=200)
for(i in 21:200){
  # if prediction is desired, use logdma.predict
  yhat[i,] <- logdma.predict(modl, dat[i,])
  # update
  modl <- logdma.update(modl, dat[i,], y[i],
                lambda = .99, autotune = FALSE)
}
# the averaging step could be also done within the loop above
ldma.stream <- logdma.average(modl, alpha = .99)
plot(ldma.stream)
```

Index