Package 'MEGB'

January 29, 2025

Type Package

Title Gradient Boosting for Longitudinal Data

Version 0.1

Author Oyebayo Ridwan Olaniran [aut, cre],
Saidat Fehintola Olaniran [aut]

Maintainer Oyebayo Ridwan Olaniran <olaniran.or@unilorin.edu.ng>

Description Gradient boosting is a powerful statistical learning method known for its ability to model
complex relationships between predictors and outcomes while performing inherent variable se-
lection.
However, traditional gradient boosting methods lack flexibility in handling longitudi-
nal data where
within-subject correlations play a critical role. In this package, we propose a novel approach
Mixed Effect Gradient Boosting ('MEGB'), designed specifically for high-
dimensional longitudinal data.
'MEGB' incorporates a flexible semi-
parametric model that embeds random effects within the gradient boosting
framework, allowing it to account for within-
individual covariance over time. Additionally, the method
efficiently handles scenarios where the number of predictors greatly exceeds the number of ob-
servations
(p>>n) making it particularly suitable for genomics data and other large-scale biomedical studies.

License GPL-2

Encoding UTF-8

Imports stats, gbm, MASS, latex2exp

NeedsCompilation no

RoxygenNote 7.3.2.9000

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Repository CRAN

Date/Publication 2025-01-29 17:00:02 UTC

Contents

MEGB   *Mixed Effect Gradient Boosting (MEGB) Algorithm*

Description

MEGB is an adaptation of the gradient boosting regression method to longitudinal data similar to the
Mixed Effect Random Forest (MERF) developed by Hajjem et. al. (2014) <doi:10.1080/00949655.2012.741599>
which was implemented by Capitaine et. al. (2020) <doi:10.1177/0962280220946080>. The algo-
rithm estimates the parameters of a semi-parametric mixed-effects model:

$$Y_i(t) = f(X_i(t)) + Z_i(t)\beta_i + \epsilon_i$$

with $Y_i(t)$ the output at time t for the ith individual; $X_i(t)$ the input predictors (fixed effects) at
time t for the ith individual; $Z_i(t)$ are the random effects at time t for the ith individual; ϵ_i is the
residual error.

Usage

```
MEGB(
  X,
  Y,
  id,
  Z,
  iter = 100,
  ntree = 500,
  time,
  shrinkage = 0.05,
  interaction.depth = 1,
  n.minobsinnode = 5,
  cv.folds = 0,
  delta = 0.001,
  verbose = TRUE
)
```

Arguments

X	[matrix]: A N x p matrix containing the p predictors of the fixed effects, column codes for a predictor.
Y	[vector]: A vector containing the output trajectories.
id	[vector]: Is the vector of the identifiers for the different trajectories.

Z	[matrix]: A N x q matrix containing the q predictor of the random effects.
iter	[numeric]: Maximal number of iterations of the algorithm. The default is set to `iter=100`
ntree	[numeric]: Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times. The default value is `ntree=500`.
time	[vector]: Is the vector of the measurement times associated with the trajectories in `Y`,`Z` and `X`.
shrinkage	[numeric]: a shrinkage parameter applied to each tree in the expansion. Also known as the learning rate or step-size reduction. The default value is set to 0.05.
interaction.depth	
	[numeric]: The maximum depth of variable interactions: 1 builds an additive model, 2 builds a model with up to two-way interactions, etc. The default value is set to 1.
n.minobsinnode	[numeric]: minimum number of observations (not total weights) in the terminal nodes of the trees. The default value is set to 5.
cv.folds	[numeric]: Number of cross-validation folds to perform. If cv.folds>1 then gbm, in addition to the usual fit, will perform a cross-validation and calculate an estimate of generalization error returned in cv_error. The default value is set to 0.
delta	[numeric]: The algorithm stops when the difference in log likelihood between two iterations is smaller than `delta`. The default value is set to 0.001
verbose	[boolean]: If TRUE, MEGB will print out number of iterations to achieve convergence. Default is TRUE.

Value

A fitted MEGB model which is a list of the following elements:

- `forest`: GBMFit obtained at the last iteration.

- `random_effects` : Predictions of random effects for different trajectories.

- `id_btilde`: Identifiers of individuals associated with the predictions `random_effects`.

- `var_random_effects`: Estimation of the variance covariance matrix of random effects.

- `sigma`: Estimation of the residual variance parameter.

- `time`: The vector of the measurement times associated with the trajectories in `Y`,`Z` and `X`.

- `LL`: Log-likelihood of the different iterations.

- `id`: Vector of the identifiers for the different trajectories.

- `OOB`: OOB error of the fitted random forest at each iteration.

Examples

```
set.seed(1)
data <-simLong(n = 20,p = 6,rel_p = 6,time_points = 10,rho_W = 0.6, rho_Z=0.6,
              random_sd_intercept = sqrt(0.5),
              random_sd_slope = sqrt(3),
           noise_sd = 0.5,linear=TRUE)  # Generate the data composed by n=20 individuals.
# Train a MEGB model on the generated data. Should take ~ 7 seconds
megb <-   MEGB(X=as.matrix(data[,-1:-5]),Y=as.matrix(data$Y),
Z=as.matrix(data[,4:5]),id=data$id,time=data$time,ntree=500,cv.folds=0,verbose=TRUE)
megb$forest # is the fitted gradient boosting (GBMFit) (obtained at the last iteration).
megb$random_effects # are the predicted random effects for each individual.
plot(megb$LL,type="o",col=2) # evolution of the log-likelihood.
megb$OOB # OOB error at each iteration.
```

---

predict.MEGB   *Predict with longitudinal trees and random forests.*

Description

Predict with longitudinal trees and random forests.

Usage

```
## S3 method for class 'MEGB'
predict(object, X, Z, id, time, ntree, ...)
```

Arguments

object	: a longituRF output of (S)MERF; (S)REEMforest; (S)MERT or (S)REEMtree function.
X	[matrix]: matrix of the fixed effects for the new observations to be predicted.
Z	[matrix]: matrix of the random effects for the new observations to be predicted.
id	[vector]: vector of the identifiers of the new observations to be predicted.
time	[vector]: vector of the time measurements of the new observations to be predicted.
ntree	[numeric]: Number of trees to be used in prediction not less than number of trees used in the model object MEGB. The default value is ntree=500.
...	: low levels arguments.

Value

vector of the predicted output for the new observations.

Examples

```
oldpar <- par(no.readonly = TRUE)
oldopt <- options()
set.seed(1)
data <-simLong(n = 20,p = 6,rel_p = 6,time_points = 10,rho_W = 0.6, rho_Z=0.6,
               random_sd_intercept = sqrt(0.5),
               random_sd_slope = sqrt(3),
             noise_sd = 0.5,linear=TRUE)  # Generate the data composed by n=20 individuals.
# Train a MEGB model on the generated data. Should take ~ 7 seconds
megb <-   MEGB(X=as.matrix(data[,-1:-5]),Y=as.matrix(data$Y),
Z=as.matrix(data[,4:5]),id=data$id,time=data$time,ntree=500,verbose=TRUE)
# Then we predict on the learning sample :
pred.MEGB <- predict(megb, X=as.matrix(data[,-1:-5]), Z=as.matrix(data[,4:5]),
id=data$id, time=data$time,ntree=500)
# Let's have a look at the predictions
# the predictions are in red while the real output trajectories are in blue:
par(mfrow=c(4,5),mar=c(2,2,2,2))
for (i in unique(data$id)){
  w <- which(data$id==i)
  plot(data$time[w],data$Y[w],type="l",col="blue")
  lines(data$time[w],pred.MEGB[w], col="red")
}
par(oldpar)
options(oldopt)
```

---

simLong	*Simulate Low/High Dimensional and Linear/Nonlinear Longitudinal dataset.*

Description

Simulate p-dimensional linear/Nonlinear mixed-effects model given by:

$$Y_i(t) = f(X_i(t)) + Z_i(t)\beta_i + \epsilon_i$$

with $Y_i(t)$ the output at time t for the ith individual; $X_i(t)$ the input predictors (fixed effects) at time t for the ith individual; $Z_i(t)$ are the random effects at time t for the ith individual; ϵ_i is the residual error with variance σ^2. If linear, $f(X_i(t)) = X_i(t)\theta$, where $\theta = 1, \forall p$, otherwise if nonlinear, the approach by Capitaine et al. (2021) is adapted.

Usage

```
simLong(
  n,
  p,
  rel_p = 6,
  time_points,
  rho_W = 0.5,
```

```
    rho_Z = 0.5,
    random_sd_intercept = 2,
    random_sd_slope = 1,
    noise_sd = 1,
    linear = TRUE
)
```

Arguments

n	[numeric]: Number of individuals.
p	[numeric]: Number of predictors.
rel_p	[numeric]: Number of relevant predictors (true predictors that are correlated to the outcome.). The default value is rel_p=6 if linear and rel_p=2 if nonlinear.
time_points	[numeric]: Number of realizations per individual. The default value is time_points=10.
rho_W	[numeric]: Within subject correlation. The default value is rho_W=0.5.
rho_Z	[numeric]: Correlation between intercept and slope for the random effect coefficients. The default value is rho_Z=0.5.
random_sd_intercept	
	[numeric]: Standard deviation for the random intercept. The default value is random_sd_intercept=$\sqrt{0.5}$.
random_sd_slope	
	[numeric]: Standard deviation for the random slope. The default value is random_sd_slope=$\sqrt{3}$.
noise_sd	[numeric]: Standard deviation for the random slope. The default value is noise_sd=0.5.
linear	[boolean]: If TRUE, a linear mixed effect model is simulated, if otherwise, a semi-parametric model similar to the one used in Capitaine et al. (2021).

Value

a dataframe of dimension (n*time_points) by (p+5) containing the following elements:

- id: vector of the individual IDs.
- time: vector of the time realizations.
- Y: vector of the outcomes variable.
- RandomIntercept: vector of the Random Intercept.
- RandomSlope: vector of the Random Slope.
- Vars : Remainder columns corresponding to the fixed effect variables.

Examples

```
set.seed(1)
data = simLong(n = 17,p = 6,rel_p = 6,time_points = 10,rho_W = 0.6, rho_Z=0.6,
               random_sd_intercept = sqrt(0.5),
               random_sd_slope = sqrt(3),
               noise_sd = 0.5,linear=FALSE) # Generate the data
head(data)   # first six rows of the data.
# Let's see the output :
```

```
w <- which(data$id==1)
plot(data$time[w],data$Y[w],type="l",ylim=c(min(data$Y),max(data$Y)), col="grey")
for (i in unique(data$id)){
  w <- which(data$id==i)
  lines(data$time[w],data$Y[w], col='grey')
}
# Let's see the fixed effects predictors:
oldpar <- par(no.readonly = TRUE)
oldopt <- options()
par(mfrow=c(2,3), mar=c(2,3,3,2))
for (i in 1:ncol(data[,-1:-5])){
  w <- which(data$id==1)
  plot(data$time[w],data[,-1:-5][w,i], col="grey",ylim=c(min(data[,-1:-5][,i]),
 max(data[,-1:-5][,i])),xlim=c(1,max(data$time)),main=latex2exp::TeX(paste0("$X^{(",i,")}$")))
  for (k in unique(data$id)){
    w <- which(data$id==k)
    lines(data$time[w],data[,-1:-5][w,i], col="grey")
  }
}
par(oldpar)
options(oldopt)
```

Index