Package: xtune (via r-universe)

October 11, 2024

Title Regularized Regression with Feature-Specific Penalties Integrating External Information

Version 2.0.0

Date 2023-06-18

Description Extends standard penalized regression (Lasso, Ridge, and Elastic-net) to allow feature-specific shrinkage based on external information with the goal of achieving a better prediction accuracy and variable selection. Examples of external information include the grouping of predictors, prior knowledge of biological importance, external p-values, function annotations, etc. The choice of multiple tuning parameters is done using an Empirical Bayes approach. A majorization-minimization algorithm is employed for implementation.

URL `https://github.com/JingxuanH/xtune`

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports glmnet, stats, crayon, selectiveInference, lbfgs

Suggests knitr, numDeriv, rmarkdown, testthat (>= 3.0.0), covr, pROC

Depends R (>= 2.10)

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

VignetteBuilder knitr

Config/testthat/edition 3

Repository https://jingxuanh.r-universe.dev

RemoteUrl https://github.com/jingxuanh/xtune

RemoteRef HEAD

RemoteSha 33d88886b6f66decb262fd90eb8e688e14d073bc

Contents

`coef_xtune` *Extract model coefficients from fitted* `xtune` *object*

Description

`coef_xtune` extracts model coefficients from objects returned by `xtune` object.

Usage

```
coef_xtune(object, ...)
```

Arguments

`object`	Fitted 'xtune' model object.
`...`	Not used

Details

`coef` and `predict` methods are provided as a convenience to extract coefficients and make prediction. `coef.xtune` simply extracts the estimated coefficients returned by `xtune`.

Value

Coefficients extracted from the fitted model.

See Also

`xtune`, `predict_xtune`

Examples

```
# See examples in \code{predict_xtune}.
```

`diet` *Simulated diet data to predict weight loss*

Description

The simulated `diet` data contains 100 observations, 14 predictors, and an binary outcome, weightloss. The external information Z is the nutrition fact about the dietary items. Z contains three external information variables: Calories, protein and carbohydrates.

Usage

```
data(diet)
```

Format

The `diet` object is a list containing three elements:

- DietItems: Matrix of predictors.
- weightloss: 0: no weight loss; 1: weight loss
- nutritionFact: External information of the predictors

References

S. Witte, John & Greenland, Sander & W. Haile, Robert & L. Bird, Cristy. (1994). Hierarchical Regression Analysis Applied to a Study of Multiple Dietary Exposures and Breast Cancer. Epidemiology (Cambridge, Mass.). 5. 612-21. 10.1097/00001648-199411000-00009.

See Also

`example`

Examples

```
data(diet)
X <- diet$DietItems
Y <- diet$weightloss
Z <- diet$nutritionFact
fit <- xtune(X,Y,Z, family = "binary")
fit$penalty.vector
```

`estimateVariance` *Estimate noise variance given predictor X and continuous outcome Y.*

Description

`estimateVariance` estimate noise variance.

Usage

```
estimateVariance(X, Y, n_rep = 5)
```

Arguments

`X`	predictor matrix of dimension n by p.
`Y`	continuous outcome vector of length n.
`n_rep`	number of repeated estimation. Default is 10.

Details

The `estimateSigma` function from selectiveInference is used repeatedly to estimate noise variance.

Value

Estimated noise variance of X and Y.

References

Stephen Reid, Jerome Friedman, and Rob Tibshirani (2014). A study of error variance estimation in lasso regression. arXiv:1311.5274.

See Also

selectiveInference

Examples

```
## simulate some data
set.seed(9)
n = 30
p = 10
sigma.square = 1
X = matrix(rnorm(n*p),n,p)
beta = c(2,-2,1,-1,rep(0,p-4))
Y = X%*%beta + rnorm(n,0,sqrt(sigma.square))

## estimate sigma square
sigma.square.est = estimateVariance(X,Y)
sigma.square.est
```

`example` *An simulated example dataset*

Description

The simulated `example` data contains 100 observations, 200 predictors, and an continuous outcome. Z contains 3 columns, each column is indicator variable (can be viewed as the grouping of predictors).

Usage

```
data(example)
```

Format

The `example` object is a list containing three elements:

- X: A simulated 100 by 200 matrix
- Y: Continuous response vector of length 100
- Z: A 200 by 3 matrix. Z_jk indicates whether predictor X_j has external variable Z_k or not.

Examples

```
data(example)
X <- example$X
Y <- example$Y
Z <- example$Z
xtune(X,Y,Z)
```

`example.multiclass` *Simulated data with multi-categorical outcome*

Description

The simulated data contains 600 observations, 800 predictors, 10 covariates, and an multiclass outcome with three categories. The external information Z contains five indicator prior covariates.

Usage

```
data(example.multiclass)
```

Format

The `example.multiclass` object is a list containing three elements:

- X: A simulated 600 by 800 matrix
- Y: Categorical outcome with three levels
- U: Covariates matrix with 600 by 10 dimension, will be forced in the model
- Z: A 800 by 5 matrix with indicator entries.

Examples

```
data(example.multiclass)
X <- example.multiclass$X
Y <- example.multiclass$Y
U <- example.multiclass$U
Z <- example.multiclass$Z
fit <- xtune(X = X,Y = Y, U = U, Z = Z, family = "multiclass", c = 0.5)
fit$penalty.vector
```

`gene` *Simulated gene data to predict weight loss*

Description

The simulated gene data contains 50 observations, 200 predictors, and an continuous outcome, bone mineral density. The external information Z is four previous study results that identifies the biological importance of genes.

Usage

```
data(gene)
```

Format

The gene object is a list containing three elements:

- GeneExpression: Matrix of gene expression predictors.
- bonedensity: Continuous outcome variable
- PreviousStudy: Whether each gene is identified by previous study results.

See Also

`diet`

Examples

```
data(gene)
X <- gene$GeneExpression
Y <- gene$bonedensity
Z <- gene$PreviousStudy
fit <- xtune(X,Y,Z)
fit$penalty.vector
```

`misclassification` *Calculate misclassification error*

Description

`misclassification` calculate misclassification error between predicted class and true class

Usage

```
misclassification(pred, true)
```

Arguments

`pred`	Predicted class
`true`	Actual class

Value

Misclassification error for binary or multiclass outcome.

See Also

To calculate the Area Under the Curve (AUC) for binary or multiclass outcomes, please refer to the `pROC`.

Examples

```
Y1 <- rbinom(10,1,0.5)
Y2 <- rnorm(10,1,0.5)
misclassification(Y1,Y2)
```

mse — *Calculate mean square error*

Description

`mse` calculate mean square error (MSE) between prediction values and true values for linear model

Usage

```
mse(pred, true)
```

Arguments

`pred`	Prediction values vector
`true`	Actual values vector

Value

mean square error

Examples

```
Y1 <- rnorm(10,0,1)
Y2 <- rnorm(10,0,1)
mse(Y1,Y2)
```

predict_xtune — *Model predictions based on fitted* `xtune` *object*

Description

`predict_xtune` produces predicted values fitting an xtune model to a new dataset

Usage

```
predict_xtune(object, newX, type = c("response", "class"), ...)
```

Arguments

`object`	Fitted 'xtune' model object.
`newX`	Matrix of values at which predictions are to be made.
`type`	Type of prediction required. For "linear" models it gives the fitted values. Type "response" gives the fitted probability scores of each category for "binary" or "multiclass" outcome. Type "class" applies to "binary" or "multiclass" models, and produces the class label corresponding to the maximum probability.
`...`	Not used

Details

`coef` and `predict` methods are provided as a convenience to extract coefficients and make prediction. `predict_xtune` simply calculate the predicted value using the estimated coefficients returned by `xtune`.

Value

A vector of predictions

See Also

`xtune`, `coef_xtune`

Examples

```
## If no Z provided, perform Empirical Bayes tuning
## simulate linear data
set.seed(9)
data(example)
X <- example$X
Y <- example$Y
Z <- example$Z


fit.eb <- xtune(X,Y)
coef_xtune(fit.eb)
predict_xtune(fit.eb,X)



## Feature specific shrinkage based on external information Z:

## simulate multi-categorical data
data(example.multiclass)
X <- example.multiclass$X
Y <- example.multiclass$Y
Z <- example.multiclass$Z

fit <- xtune(X,Y,Z,family = "multiclass")


## Coef and predict methods
coef_xtune(fit)
predict_xtune(fit,X, type = "class")
```

xtune *Regularized regression incorporating external information*

Description

xtune uses an Empirical Bayes approach to integrate external information into regularized regression models for both linear and categorical outcomes. It fits models with feature-specific penalty parameters based on external information.

Usage

```
xtune(
  X,
  Y,
  Z = NULL,
  U = NULL,
  family = c("linear", "binary", "multiclass"),
  c = 0.5,
  epsilon = 5,
  sigma.square = NULL,
  message = TRUE,
  control = list()
)
```

Arguments

X	Numeric design matrix of explanatory variables (n observations in rows, p predictors in columns). xtune includes an intercept by default.
Y	Outcome vector of dimension n.
Z	Numeric information matrix about the predictors (p rows, each corresponding to a predictor in X; q columns of external information about the predictors, such as prior biological importance). If Z is the grouping of predictors, it is best if user codes it as a dummy variable (i.e. each column indicating whether predictors belong to a specific group).
U	Covariates to be adjusted in the model (matrix with n observations in rows, u predictors in columns). Covariates are non-penalized in the model.
family	The family of the model according to different types of outcomes including "linear", "binary", and "multiclass".
c	The elastic-net mixing parameter ranging from 0 to 1. When c = 1, the model corresponds to Lasso. When c is set to 0, it corresponds to Ridge. For values between 0 and 1 (with a default of 0.5), the model corresponds to the elastic net.
epsilon	The parameter controls the boundary of the alpha. The maximum value that alpha could achieve equals to epsilon times of alpha max calculated by the pathwise coordinate descent. A larger value of epsilon indicates a stronger shrinkage effect (with a default of 5).

`sigma.square`	A user-supplied noise variance estimate. Typically, this is left unspecified, and the function automatically computes an estimated sigma square values using R package `selectiveinference`.
`message`	Generates diagnostic message in model fitting. Default is TRUE.
`control`	Specifies `xtune` control object. See `xtune.control` for more details.

Details

`xtune` has two main usages:

- The basic usage of it is to choose the tuning parameter λ in elastic net regression using an Empirical Bayes approach, as an alternative to the widely-used cross-validation. This is done by calling `xtune` without specifying external information matrix Z.
- More importantly, if an external information Z about the predictors X is provided, `xtune` can allow predictor-specific shrinkage parameters for regression coefficients in penalized regression models. The idea is that Z might be informative for the effect-size of regression coefficients, therefore we can guide the penalized regression model using Z.

Please note that the number of rows in Z should match with the number of columns in X. Since each column in Z is a feature about X. See here for more details on how to specify Z.

A majorization-minimization procedure is employed to fit `xtune`.

Value

An object with S3 class `xtune` containing:

`beta.est`	The fitted vector of coefficients.
`penalty.vector`	The estimated penalty vector applied to each regression coefficient. Similar to the `penalty.factor` argument in glmnet.
`lambda`	The estimated λ value. Note that the lambda value is calculated to reflect that the fact that penalty factors are internally rescaled to sum to nvars in glmnet. Similar to the `lambda` argument in glmnet.
`alpha.est`	The estimated second-level coefficient for prior covariate Z. The first value is the intercept of the second-level coefficient.
`n_iter`	Number of iterations used until convergence.
`method`	Same as in argument above
`sigma.square`	The estimated sigma square value using `estimateVariance`, if `sigma.square` is left unspecified. When `family` equals to "binary" or "multiclass", the `sigma.square` equals to NULL.
`family`	same as above
`likelihood.score`	A vector containing the marginal likelihood value of the fitted model at each iteration.

Author(s)

Jingxuan He and Chubing Zeng

See Also

predict_xtune, as well as glmnet.

Examples

```
## use simulated example data
set.seed(1234567)
data(example)
X <- example$X
Y <- example$Y
Z <- example$Z

## Empirical Bayes tuning to estimate tuning parameter, as an alternative to cross-validation:

fit.eb <- xtune(X=X,Y=Y, family = "linear")
fit.eb$lambda


### compare with tuning parameter chosen by cross-validation, using glmnet

fit.cv <- glmnet::cv.glmnet(x=X,y=Y,alpha = 0.5)
fit.cv$lambda.min


## Feature-specific penalties based on external information Z:

fit.diff <- xtune(X=X,Y=Y,Z=Z, family = "linear")
fit.diff$penalty.vector
```

`xtune.control` *Control function for xtune fitting*

Description

Control function for `xtune` fitting.

Usage

```
xtune.control(
  alpha.est.init = NULL,
  max_s = 20,
  margin_s = 1e-05,
  maxstep = 100,
  margin = 0.001,
  maxstep_inner = 100,
  margin_inner = 0.001,
  compute.likelihood = FALSE,
  verbosity = FALSE,
  standardize = TRUE,
  intercept = TRUE
)
```

Arguments

`alpha.est.init`	Initial values of alpha vector supplied to the algorithm. Alpha values are the hyper-parameters for the double exponential prior of regression coefficients, and it controls the prior variance of regression coefficients. Default is a vector of 0 with length p.
`max_s`	Maximum number of outer loop iterations for binary or multiclass outcomes. Default is 20.
`margin_s`	Convergence threshold of the outer loop for binary or multiclass outcomes. Default is 1e-5.
`maxstep`	Maximum number of iterations. Default is 100.
`margin`	Convergence threshold. Default is 1e-3.
`maxstep_inner`	Maximum number of iterations for the inner loop of the majorization-minimization algorithm. Default is 100.
`margin_inner`	Convergence threshold for the inner loop of the majorization-minimization algorithm. Default is 1e-3.
`compute.likelihood`	Should the function compute the marginal likelihood for hyper-parameters at each step of the update? Default is TRUE.
`verbosity`	Track algorithm update process? Default is FALSE.
`standardize`	Standardize X or not, same as the standardized option in `glmnet`.
`intercept`	Should intercept(s) be fitted (default=TRUE) or set to zero (FALSE), same as the intercept option in `glmnet`.

Value

A list of control objects after the checking.

Index