Package ‘L0Learn’

April 3, 2021

Type Package
Title Fast Algorithms for Best Subset Selection
Version 2.0.3
Date 2021-03-28
Description Highly optimized toolkit for approximately solving L0-regularized learning problems (a.k.a. best subset selection). The algorithms are based on coordinate descent and local combinatorial search. For more details, check the paper by Hazimeh and Mazumder (2020); the link is provided in the URL field below.

https://github.com/hazimehh

BugReports https://github.com/hazimehh/L0Learn/issues
License MIT + file LICENSE
Depends R (>= 3.3.0)
SystemRequirements C++11
Imports Rcpp (>= 0.12.13), Matrix, methods, ggplot2, reshape2, MASS
LinkingTo Rcpp, RcppArmadillo
RoxygenNote 7.1.1
Suggests knitr, rmarkdown, testthat, pracma, raster
VignetteBuilder knitr
NeedsCompilation yes
Author Hussein Hazimeh [aut, cre], Rahul Mazumder [aut], Tim Nonet [aut]
Maintainer Hussein Hazimeh <hazimeh@mit.edu>
Repository CRAN
Date/Publication 2021-04-03 13:50:28 UTC
Description

L0Learn fits regularization paths for L0-regularized regression and classification problems. Specifically, it can solve either one of the following problems over a grid of $\lambda$ and $\gamma$ values:

$$ \min_{\beta_0, \beta} \sum_{i=1}^{n} \ell(y_i, \beta_0 + \langle x_i, \beta \rangle) + \lambda \|\beta\|_0 \quad (L0) $$

$$ \min_{\beta_0, \beta} \sum_{i=1}^{n} \ell(y_i, \beta_0 + \langle x_i, \beta \rangle) + \lambda \|\beta\|_0 + \gamma \|\beta\|_1 \quad (L0L1) $$

$$ \min_{\beta_0, \beta} \sum_{i=1}^{n} \ell(y_i, \beta_0 + \langle x_i, \beta \rangle) + \lambda \|\beta\|_0 + \gamma \|\beta\|_2^2 \quad (L0L2) $$

where $\ell$ is the loss function. We currently support regression using squared error loss and classification using either logistic loss or squared hinge loss. Pathwise optimization can be done using either cyclic coordinate descent (CD) or local combinatorial search. The core of the toolkit is implemented in C++ and employs many computational tricks and heuristics, leading to competitive running times. CD runs very fast and typically leads to relatively good solutions. Local combinatorial search can find higher-quality solutions (at the expense of increased running times). The toolkit has the following six main methods:

- `L0Learn.fit`: Fits an L0-regularized model.
- `L0Learn.cvfit`: Performs k-fold cross-validation.
- `print`: Prints a summary of the path.
- `coeff`: Extracts solutions(s) from the path.
- `predict`: Predicts response using a solution in the path.
- `plot`: Plots the regularization path or cross-validation error.
References


---

coef.L0Learn

### Extract Solutions

**Description**

Extracts a specific solution in the regularization path.

**Usage**

```r
## S3 method for class 'L0Learn'
coef(object, lambda = NULL, gamma = NULL, ...)

## S3 method for class 'L0LearnCV'
coef(object, lambda = NULL, gamma = NULL, ...)
```

**Arguments**

- `object`: The output of `L0Learn.fit` or `L0Learn.cvfit`
- `lambda`: The value of lambda at which to extract the solution.
- `gamma`: The value of gamma at which to extract the solution.
- `...`: ignore

**Details**

If both lambda and gamma are not supplied, then a matrix of coefficients for all the solutions in the regularization path is returned. If lambda is supplied but gamma is not, the smallest value of gamma is used.

**Examples**

```r
# Generate synthetic data for this example
data <- GenSynthetic(n=500,p=1000,k=10,seed=1)
X = data$X
y = data$y

# Fit an L0L2 Model with 10 values of Gamma ranging from 0.0001 to 10, using coordinate descent
fit <- L0Learn.fit(X, y, penalty="L0L2", maxSuppSize=50, nGamma=10, gammaMin=0.0001, gammaMax = 10)
print(fit)

# Extract the coefficients of the solution at lambda = 0.0361829 and gamma = 0.0001
coef(fit, lambda=0.0361829, gamma=0.0001)

# Extract the coefficients of all the solutions in the path
coef(fit)
```
GenSynthetic 

Generate Synthetic Data

Description

Generates a synthetic dataset as follows: 1) Sample every element in data matrix X from N(0,1). 2) Generate a vector B with the first k entries set to 1 and the rest are zeros. 3) Sample every element in the noise vector e from N(0,1). 4) Set y = XB + b0 + e.

Usage

GenSynthetic(n, p, k, seed, rho = 0, b0 = 0, error_ratio = 1)

Arguments

n 
Number of samples
p 
Number of features
k 
Number of non-zeros in true vector of coefficients
seed 
The seed used for randomly generating the data
rho 
The threshold for setting values to 0. if |X(i, j)| > rho => X(i, j) <- 0
b0 
intercept value to translate y by.
error_ratio 
multiplier for the magnitude of the error term 'e'.

Value

A list containing: the data matrix X, the response vector y, the coefficients B, the error vector e, the intercept term b0.

Examples

data <- GenSynthetic(n=500, p=1000, k=10, seed=1)
X = data$X
y = data$y

GenSyntheticHighCorr 

Generate Exponential Correlated Synthetic Data

Description

Generates a synthetic dataset as follows: 1) Generate a correlation matrix, SIG, where item [i, j] = A^|i-j|. 2) Draw from a Multivariate Normal Distribution using (mu and SIG) to generate X. 3) Generate a vector B with every ~p/k entry set to 1 and the rest are zeros. 4) Sample every element in the noise vector e from N(0,1). 4) Set y = XB + b0 + e.
Usage

GenSyntheticHighCorr(
    n,
    p,
    k,
    seed,
    rho = 0,
    b0 = 0,
    noise_ratio = 1,
    mu = 0,
    base_cor = 0.9
)

Arguments

n: Number of samples
p: Number of features
k: Number of non-zeros in true vector of coefficients
seed: The seed used for randomly generating the data
rho: The threshold for setting values to 0. if |X(i, j)| > rho => X(i, j) <- 0
b0: intercept value to scale y by.
noise_ratio: The multiplier of noise to apply when calculating e. e[i] = noise_ratio*N(0, 1).

Value

A list containing: the data matrix X, the response vector y, the coefficients B, the error vector e, the intercept term b0.

Examples

data <- GenSyntheticHighCorr(n=500, p=1000, k=10, seed=1)
X = data$X
y = data$y

Description

Computes a regularization path and performs K-fold cross-validation.
Usage

L0Learn.cvfit(
    x,
    y,
    loss = "SquaredError",
    penalty = "L0",
    algorithm = "CD",
    maxSuppSize = 100,
    nLambda = 100,
    nGamma = 10,
    gammaMax = 10,
    gammaMin = 1e-04,
    partialSort = TRUE,
    maxIters = 200,
    rtol = 1e-06,
    atol = 1e-09,
    activeSet = TRUE,
    activeSetNum = 3,
    maxSwaps = 100,
    scaleDownFactor = 0.8,
    screenSize = 1000,
    autoLambda = NULL,
    lambdaGrid = list(),
    nFolds = 10,
    seed = 1,
    excludeFirstK = 0,
    intercept = TRUE,
    lows = -Inf,
    highs = Inf
)

Arguments

x The data matrix.
y The response vector. For classification, we only support binary vectors.
loss The loss function. Currently we support the choices "SquaredError" (for regression), "Logistic" (for logistic regression), and "SquaredHinge" (for smooth SVM).
penalty The type of regularization. This can take either one of the following choices: "L0", "LOL2", and "LOL1".
algorithm The type of algorithm used to minimize the objective function. Currently "CD" and "CDPSI" are are supported. "CD" is a variant of cyclic coordinate descent and runs very fast. "CDPSI" performs local combinatorial search on top of CD and typically achieves higher quality solutions (at the expense of increased running time).
maxSuppSize The maximum support size at which to terminate the regularization path. We recommend setting this to a small fraction of min(n,p) (e.g. 0.05 * min(n,p)) as
L0 regularization typically selects a small portion of non-zeros.

**nLambda**
The number of Lambda values to select (recall that Lambda is the regularization parameter corresponding to the L0 norm). This value is ignored if 'lambdaGrid' is supplied.

**nGamma**
The number of Gamma values to select (recall that Gamma is the regularization parameter corresponding to L1 or L2, depending on the chosen penalty). This value is ignored if 'lambdaGrid' is supplied.

**gammaMax**
The maximum value of Gamma when using the L0L2 penalty. For the L0L1 penalty this is automatically selected.

**gammaMin**
The minimum value of Gamma when using the L0L2 penalty. For the L0L1 penalty, the minimum value of gamma in the grid is set to gammaMin * gammaMax. Note that this should be a strictly positive quantity.

**partialSort**
If TRUE partial sorting will be used for sorting the coordinates to do greedy cycling (see our paper for for details). Otherwise, full sorting is used.

**maxIters**
The maximum number of iterations (full cycles) for CD per grid point.

**rtol**
The relative tolerance which decides when to terminate optimization (based on the relative change in the objective between iterations).

**atol**
The absolute tolerance which decides when to terminate optimization (based on the absolute L2 norm of the residuals).

**activeSet**
If TRUE, performs active set updates.

**activeSetNum**
The number of consecutive times a support should appear before declaring support stabilization.

**maxSwaps**
The maximum number of swaps used by CDPSI for each grid point.

**scaleDownFactor**
This parameter decides how close the selected Lambda values are. The choice should be strictly between 0 and 1 (i.e., 0 and 1 are not allowed). Larger values lead to closer lambdas and typically to smaller gaps between the support sizes. For details, see our paper - Section 5 on Adaptive Selection of Tuning Parameters.

**screenSize**
The number of coordinates to cycle over when performing initial correlation screening.

**autoLambda**
Ignored parameter. Kept for backwards compatibility.

**lambdaGrid**
A grid of Lambda values to use in computing the regularization path. This is by default an empty list and is ignored. When specified, LambdaGrid should be a list of length 'nGamma', where the ith element (corresponding to the ith gamma) should be a decreasing sequence of lambda values which are used by the algorithm when fitting for the ith value of gamma (see the vignette for details).

**nFolds**
The number of folds for cross-validation.

**seed**
The seed used in randomly shuffling the data for cross-validation.

**excludeFirstK**
This parameter takes non-negative integers. The first excludeFirstK features in x will be excluded from variable selection, i.e., the first excludeFirstK variables will not be included in the L0-norm penalty (they will still be included in the L1 or L2 norm penalties).
intercept  If FALSE, no intercept term is included in the model.

lows  Lower bounds for coefficients. Either a scalar for all coefficients to have the same bound or a vector of size p (number of columns of X) where lows[i] is the lower bound for coefficient i.

highs  Upper bounds for coefficients. Either a scalar for all coefficients to have the same bound or a vector of size p (number of columns of X) where highs[i] is the upper bound for coefficient i.

Value

An S3 object of type "L0LearnCV" describing the regularization path. The object has the following members.

cvMeans  This is a list, where the ith element is the sequence of cross-validation errors corresponding to the ith gamma value, i.e., the sequence cvMeans[i] corresponds to fit$gamma[i].

cvSDs  This a list, where the ith element is a sequence of standard deviations for the cross-validation errors: cvSDs[i] corresponds to cvMeans[i].

fit  The fitted model with type "L0Learn", i.e., this is the same object returned by L0Learn.fit.

Examples

# Generate synthetic data for this example
data <- GenSynthetic(n=500,p=1000,k=10,seed=1)
X = data$X
y = data$y

# Perform 5-fold cross-validation on an L0L2 regression model with 5 values of Gamma ranging from 0.0001 to 10
fit <- L0Learn.cvfit(X, y, nFolds=5, seed=1, penalty="L0L2", maxSuppSize=20, nGamma=5, gammaMin=0.0001, gammaMax = 10)
print(fit)
# Plot the graph of cross-validation error versus lambda for gamma = 0.0001
plot(fit, gamma=0.0001)
# Extract the coefficients at lambda = 0.0361829 and gamma = 0.0001
coef(fit, lambda=0.0361829, gamma=0.0001)
# Apply the fitted model on X to predict the response
predict(fit, newx = X, lambda=0.0361829, gamma=0.0001)

Description

Computes the regularization path for the specified loss function and penalty function (which can be a combination of the L0, L1, and L2 norms).
Usage

```r
LOLearn.fit(
  x, y,
  loss = "SquaredError",
  penalty = "L0",
  algorithm = "CD",
  maxSuppSize = 100,
  nLambda = 100,
  nGamma = 10,
  gammaMax = 10,
  gammaMin = 1e-04,
  partialSort = TRUE,
  maxIters = 200,
  rtol = 1e-06,
  atol = 1e-09,
  activeSet = TRUE,
  activeSetNum = 3,
  maxSwaps = 100,
  scaleDownFactor = 0.8,
  screenSize = 1000,
  autoLambda = NULL,
  lambdaGrid = list(),
  excludeFirstK = 0,
  intercept = TRUE,
  lows = -Inf,
  highs = Inf
)
```

Arguments

- **x**: The data matrix.
- **y**: The response vector. For classification, we only support binary vectors.
- **loss**: The loss function. Currently we support the choices "SquaredError" (for regression), "Logistic" (for logistic regression), and "SquaredHinge" (for smooth SVM).
- **penalty**: The type of regularization. This can take either one of the following choices: "L0", "L0L2", and "L0L1".
- **algorithm**: The type of algorithm used to minimize the objective function. Currently "CD" and "CDPSI" are supported. "CD" is a variant of cyclic coordinate descent and runs very fast. "CDPSI" performs local combinatorial search on top of CD and typically achieves higher quality solutions (at the expense of increased running time).
- **maxSuppSize**: The maximum support size at which to terminate the regularization path. We recommend setting this to a small fraction of min(n,p) (e.g. 0.05 * min(n,p)) as L0 regularization typically selects a small portion of non-zeros.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nLambda</td>
<td>The number of Lambda values to select (recall that Lambda is the regularization parameter corresponding to the L0 norm). This value is ignored if 'lambdaGrid' is supplied.</td>
</tr>
<tr>
<td>nGamma</td>
<td>The number of Gamma values to select (recall that Gamma is the regularization parameter corresponding to L1 or L2, depending on the chosen penalty). This value is ignored if 'lambdaGrid' is supplied and will be set to length(lambdaGrid)</td>
</tr>
<tr>
<td>gammaMax</td>
<td>The maximum value of Gamma when using the L0L2 penalty. For the L0L1 penalty this is automatically selected.</td>
</tr>
<tr>
<td>gammaMin</td>
<td>The minimum value of Gamma when using the L0L2 penalty. For the L0L1 penalty, the minimum value of gamma in the grid is set to gammaMin * gammaMax. Note that this should be a strictly positive quantity.</td>
</tr>
<tr>
<td>partialSort</td>
<td>If TRUE partial sorting will be used for sorting the coordinates to do greedy cycling (see our paper for for details). Otherwise, full sorting is used.</td>
</tr>
<tr>
<td>maxIters</td>
<td>The maximum number of iterations (full cycles) for CD per grid point.</td>
</tr>
<tr>
<td>rtol</td>
<td>The relative tolerance which decides when to terminate optimization (based on the relative change in the objective between iterations).</td>
</tr>
<tr>
<td>atol</td>
<td>The absolute tolerance which decides when to terminate optimization (based on the absolute L2 norm of the residuals).</td>
</tr>
<tr>
<td>activeSet</td>
<td>If TRUE, performs active set updates.</td>
</tr>
<tr>
<td>activeSetNum</td>
<td>The number of consecutive times a support should appear before declaring support stabilization.</td>
</tr>
<tr>
<td>maxSwaps</td>
<td>The maximum number of swaps used by CDPSI for each grid point.</td>
</tr>
<tr>
<td>scaleDownFactor</td>
<td>This parameter decides how close the selected Lambda values are. The choice should be strictly between 0 and 1 (i.e., 0 and 1 are not allowed). Larger values lead to closer lambdas and typically to smaller gaps between the support sizes. For details, see our paper - Section 5 on Adaptive Selection of Tuning Parameters.</td>
</tr>
<tr>
<td>screenSize</td>
<td>The number of coordinates to cycle over when performing initial correlation screening.</td>
</tr>
<tr>
<td>autoLambda</td>
<td>Ignored parameter. Kept for backwards compatibility.</td>
</tr>
<tr>
<td>lambdaGrid</td>
<td>A grid of Lambda values to use in computing the regularization path. This is by default an empty list and is ignored. When specified, LambdaGrid should be a list of length 'nGamma', where the ith element (corresponding to the ith gamma) should be a decreasing sequence of lambda values which are used by the algorithm when fitting for the ith value of gamma (see the vignette for details).</td>
</tr>
<tr>
<td>excludeFirstK</td>
<td>This parameter takes non-negative integers. The first excludeFirstK features in x will be excluded from variable selection, i.e., the first excludeFirstK variables will not be included in the L0-norm penalty (they will still be included in the L1 or L2 norm penalties.).</td>
</tr>
<tr>
<td>intercept</td>
<td>If FALSE, no intercept term is included in the model.</td>
</tr>
<tr>
<td>lows</td>
<td>Lower bounds for coefficients. Either a scalar for all coefficients to have the same bound or a vector of size p (number of columns of X) where lows[i] is the lower bound for coefficient i.</td>
</tr>
</tbody>
</table>

**L0Learn.fit**
highs

Upper bounds for coefficients. Either a scalar for all coefficients to have the
same bound or a vector of size p (number of columns of X) where highs[i] is the
upper bound for coefficient i.

Value

An S3 object of type "L0Learn" describing the regularization path. The object has the following
members.

a0  a0 is a list of intercept sequences. The ith element of the list (i.e., a0[[i]]) is the
sequence of intercepts corresponding to the ith gamma value (i.e., gamma[i]).

beta  This is a list of coefficient matrices. The ith element of the list is a p x length(lambda)
matrix which corresponds to the ith gamma value. The jth column in each coef-
ficient matrix is the vector of coefficients for the jth lambda value.

lambda  This is the list of lambda sequences used in fitting the model. The ith element
of lambda (i.e., lambda[[i]]) is the sequence of Lambda values corresponding to
the ith gamma value.

gamma  This is the sequence of gamma values used in fitting the model.

suppSize  This is a list of support size sequences. The ith element of the list is a sequence
of support sizes (i.e., number of non-zero coefficients) corresponding to the ith
gamma value.

converged  This is a list of sequences for checking whether the algorithm has converged at
every grid point. The ith element of the list is a sequence corresponding to the
ith value of gamma, where the jth element in each sequence indicates whether
the algorithm has converged at the jth value of lambda.

Examples

# Generate synthetic data for this example
data <- GenSynthetic(n=500,p=1000,k=10,seed=1)
X = data$X
y = data$y

# Fit an L0 regression model with a maximum of 50 non-zeros using coordinate descent (CD)
fit1 <- L0Learn.fit(X, y, penalty="L0", maxSuppSize=50)
print(fit1)
# Extract the coefficients at lambda = 0.0325142
coef(fit1, lambda=0.0325142)
# Apply the fitted model on X to predict the response
predict(fit1, newx = X, lambda=0.0325142)

# Fit an L0 regression model with a maximum of 50 non-zeros using CD and local search
fit2 <- L0Learn.fit(X, y, penalty="L0", algorithm="CDPSI", maxSuppSize=50)
print(fit2)

# Fit an L0L2 regression model with 10 values of Gamma ranging from 0.0001 to 10, using CD
fit3 <- L0Learn.fit(X, y, penalty="L0L2", maxSuppSize=50, nGamma=10, gammaMin=0.0001, gammaMax = 10)
print(fit3)
# Extract the coefficients at lambda = 0.0361829 and gamma = 0.0001
coef(fit3, lambda=0.0361829, gamma=0.0001)
# Apply the fitted model on X to predict the response
predict(fit3, newx = X, lambda=0.0361829, gamma=0.0001)

# Fit an L0 logistic regression model
# First, convert the response to binary
y = sign(y)
fit4 <- L0Learn.fit(X, y, loss="Logistic", maxSuppSize=20)
print(fit4)

plot.L0Learn

## S3 method for class 'L0Learn'
plot(x, gamma = 0, showLines = FALSE, ...)

### Arguments

- **x**: The output of L0Learn.fit
- **gamma**: The value of gamma at which to plot.
- **showLines**: If TRUE, the lines connecting the points in the plot are shown.
- **...**: ignore

### Examples

# Generate synthetic data for this example
data <- GenSynthetic(n=500, p=1000, k=10, seed=1)
X = data$X
y = data$y
# Fit an L0 Model with a maximum of 50 non-zeros
fit <- L0Learn.fit(X, y, penalty="L0", maxSuppSize=50)
plot(fit, gamma=0)
plot.L0LearnCV
Plot Cross-validation Errors

Description
Plots cross-validation errors for a given gamma.

Usage
```r
## S3 method for class 'L0LearnCV'
plot(x, gamma = 0, ...)
```

Arguments
- `x` The output of L0Learn.cvfit
- `gamma` The value of gamma at which to plot.
- `...` ignore

Examples
```r
# Generate synthetic data for this example
data <- GenSynthetic(n=500, p=1000, k=10, seed=1)
X = data$X
y = data$y

# Perform 5-fold cross-validation on an L0L2 Model with 5 values of
# Gamma ranging from 0.0001 to 10
fit <- L0Learn.cvfit(X, y, nFolds=5, seed=1, penalty="L0L2",
maxSuppSize=20, nGamma=5, gammaMin=0.0001, gammaMax = 10)
# Plot the graph of cross-validation error versus lambda for gamma = 0.0001
plot(fit, gamma=0.0001)
```

predict.L0Learn
Predict Response

Description
Predicts the response for a given sample.

Usage
```r
## S3 method for class 'L0Learn'
predict(object, newx, lambda = NULL, gamma = NULL, ...)
```

```r
## S3 method for class 'L0LearnCV'
predict(object, newx, lambda = NULL, gamma = NULL, ...)
```
Arguments

- **object**
  The output of L0Learn.fit or L0Learn.cvfit

- **newx**
  A matrix on which predictions are made. The matrix should have p columns.

- **lambda**
  The value of lambda to use for prediction. A summary of the lambdas in the regularization path can be obtained using print(fit).

- **gamma**
  The value of gamma to use for prediction. A summary of the gammas in the regularization path can be obtained using print(fit).

- **...**
  Ignore

Details

If both lambda and gamma are not supplied, then a matrix of predictions for all the solutions in the regularization path is returned. If lambda is supplied but gamma is not, the smallest value of gamma is used. In case of logistic regression, probability values are returned.

Examples

```r
# Generate synthetic data for this example
data <- GenSynthetic(n=500, p=1000, k=10, seed=1)
X = data$X
y = data$y

# Fit an L0L2 Model with 10 values of Gamma ranging from 0.0001 to 10, using coordinate descent
fit <- L0Learn.fit(X, y, penalty="L0L2", maxSuppSize=50, nGamma=10, gammaMin=0.0001, gammaMax = 10)
print(fit)

# Apply the fitted model with lambda=0.0361829 and gamma=0.0001 on X to predict the response
predict(fit, newx = X, lambda=0.0361829, gamma=0.0001)

# Apply the fitted model on X to predict the response for all the solutions in the path
predict(fit, newx = X)
```

print.L0Learn

Print L0Learn.fit object

Description

Prints a summary of L0Learn.fit

Usage

```r
## S3 method for class 'L0Learn'
print(x, ...)

## S3 method for class 'L0LearnCV'
print(x, ...)
```
print.L0Learn

Arguments

x  The output of L0Learn.fit or L0Learn.cvfit

...  ignore
Index

coeff, 2
coeff.L0Learn, 3
coeff.L0LearnCV (coeff.L0Learn), 3

GenSynthetic, 4
GenSyntheticHighCorr, 4

L0Learn-package, 2
L0Learn.cvfit, 2, 5
L0Learn.fit, 2, 8, 8

plot, 2
plot.L0Learn, 12
plot.L0LearnCV, 13
predict, 2
predict.L0Learn, 13
predict.L0LearnCV (predict.L0Learn), 13
print, 2
print.L0Learn, 14
print.L0LearnCV (print.L0Learn), 14