Package 'L0Learn'

June 23, 2018

Type Package

Title Fast Algorithms for Best Subset Selection

Version 0.3.1

Date 2018-06-22

Author Hussein Hazimeh, Rahul Mazumder

Maintainer Hussein Hazimeh <hazimeh@mit.edu>

Description Highly optimized coordinate descent and local combinatorial search algorithms for (approximately) solving L0-regularized learning problems.

License MIT + file LICENSE

Depends R (>= 3.3.0)

Imports Rcpp (>= 0.12.13), Matrix, methods, ggplot2, reshape2

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.0.1

R topics documented:

L0Learn-package *A package for L0-regularized learning*

Description

L0Learn fits regularization paths for L0-regularized least squares problems. Specifically, it can solve either one of the following problems:

$$\min_{\beta} \frac{1}{2}||y - X\beta||^2 + \lambda||\beta||_0 \qquad (L0)$$

$$\min_{\beta} \frac{1}{2}||y - X\beta||^2 + \lambda||\beta||_0 + \gamma||\beta||_1 \quad (L0L1)$$

$$\min_{\beta} \frac{1}{2}||y - X\beta||^2 + \lambda||\beta||_0 + \gamma||\beta||_2^2 \quad (L0L2)$$

over a grid of λ and γ values. Optimization can be done using 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 very competitive running times. CD runs very fast and typically leads to relatively good solutions. Local combinatorial search leads to 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.
- `coef`: Extracts solutions(s) from the path.
- `predict`: Predicts response using a solution in the path.
- `plot`: Plots the regularization path or cross-validation error.

coef.L0Learn *Extract Solutions*

Description

Extracts a specific solution in the regularization path

Usage

```
## S3 method for class 'L0Learn'
coef(object, lambda, gamma = 0, ...)

## S3 method for class 'L0LearnCV'
coef(object, lambda, gamma = 0, ...)
```

Arguments

object	The output of L0Learn.fit or L0Learn.cvfit
lambda	The value(s) of lambda at which to extract the solution.
gamma	The value of gamma at which to extract the solution. Note that, unlike lambda, this can only take single values.
...	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 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)
```

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 + e.

Usage

```
GenSynthetic(n, p, k, seed)
```

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

Value

A list containing the data matrix X and the response vector y.

Examples

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

`L0Learn.cvfit` *Cross Validation*

Description

Fits an L0 model 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,
  tol = 1e-06, activeSet = TRUE, activeSetNum = 3, maxSwaps = 100,
  scaleDownFactor = 0.8, screenSize = 1000, autoLambda = TRUE,
  lambdaGrid = list(0), nFolds = 10, seed = 1)
```

Arguments

`x`	The data matrix.
`y`	The response vector.
`loss`	The loss function to be minimized. Currently we support the choice "SquaredError".
`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 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).
`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).
`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.
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.
tol	The tolerance which decides when to terminate CD (based on the relative change in the objective).
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 between 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	If FALSE, the user specifier a grid of Lambda values through the lambdaGrid parameter. Otherwise, if TRUE, the values of Lambda are automatically selected based on the data.
lambdaGrid	A vector of Lambda values to use in computing the regularization path. This is ignored unless autoLambda = FALSE.
nFolds	The number of folds for cross-validation.
seed	The seed used in randomly shuffling the data for cross-validation.

Value

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

cvMeans	This is a list, where the ith element is a sequence of cross-validation errors corresponding to the ith gamma value
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 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)
```

`L0Learn.fit` *Fit an L0-regularized model*

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

```
L0Learn.fit(x, y, loss = "SquaredError", penalty = "L0", algorithm = "CD",
  maxSuppSize = 100, nLambda = 100, nGamma = 10, gammaMax = 10,
  gammaMin = 1e-04, partialSort = TRUE, maxIters = 200, tol = 1e-06,
  activeSet = TRUE, activeSetNum = 3, maxSwaps = 100,
  scaleDownFactor = 0.8, screenSize = 1000, autoLambda = TRUE,
  lambdaGrid = list(0))
```

Arguments

`x`	The data matrix.
`y`	The response vector.
`loss`	The loss function to be minimized. Currently we support the choice "SquaredError".
`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 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).

`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).

`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.

`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.

`tol` The tolerance which decides when to terminate CD (based on the relative change in the objective).

`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 between 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` If FALSE, the user specifier a grid of Lambda values through the lambdaGrid parameter. Otherwise, if TRUE, the values of Lambda are automatically selected based on the data.

`lambdaGrid` A vector of Lambda values to use in computing the regularization path. This is ignored unless autoLambda = FALSE.

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 the coefficient 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 a 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 sequences of support sizes (i.e., number of non-zero coefficients) corresponding to the ith gamma value.
converged	This is a list of sequences. The ith element of the list is a sequence corresponding to the ith value of gamma, where the jth element in in the 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 Model with a maximum of 50 non-zeros using coordinate descent
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 Model with a maximum of 50 non-zeros using coordinate descent and local search
fit2 <- L0Learn.fit(X, y, penalty="L0", algorithm="CDPSI", maxSuppSize=50)
print(fit2)

# Fit an L0L2 Model with 10 values of Gamma ranging from 0.0001 to 10, using coordinate descent
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)
```

---

plot.L0Learn *Plot Regularization Path*

Description

Plots the regularization path for a given gamma.

Usage

```
## S3 method for class 'L0Learn'
plot(x, gamma, ...)
```

Arguments

x	The output of L0Learn.fit
gamma	The value of gamma at which to plot.
...	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

```
## S3 method for class 'L0LearnCV'
plot(x, gamma, ...)
```

Arguments

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

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 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

```
## S3 method for class 'L0Learn'
predict(object, newx, lambda, gamma = 0, ...)

## S3 method for class 'L0LearnCV'
predict(object, newx, lambda, gamma = 0, ...)
```

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(s) 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

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 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)
```

print.L0Learn *Print L0Learn.fit object*

Description

Prints a summary of L0Learn.fit

Usage

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

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

Arguments

x	The output of L0Learn.fit or L0Learn.cvfit
...	ignore

Index