

Package: L0Learn (via r-universe)

September 29, 2024

Type Package

Title Fast Algorithms for Best Subset Selection

Version 2.1.0

Date 2023-03-04

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) <[doi:10.1287/opre.2019.1919](https://doi.org/10.1287/opre.2019.1919)>.

URL <https://github.com/hazimehh/L0Learn>
<https://pubsonline.informs.org/doi/10.1287/opre.2019.1919>

BugReports <https://github.com/hazimehh/L0Learn/issues>

License MIT + file LICENSE

Depends R (>= 3.3.0)

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

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.2.3

Encoding UTF-8

Suggests knitr, rmarkdown, testthat, pracma, raster, covr

VignetteBuilder knitr

NeedsCompilation yes

Author Hussein Hazimeh [aut, cre], Rahul Mazumder [aut], Tim Nonet [aut]

Maintainer Hussein Hazimeh <huseinhaz@gmail.com>

Repository CRAN

Date/Publication 2023-03-07 08:00:18 UTC

Contents

L0Learn-package	2
coef.L0Learn	3
GenSynthetic	4
GenSyntheticHighCorr	5
GenSyntheticLogistic	6
L0Learn.cvfit	7
L0Learn.fit	10
plot.L0Learn	13
plot.L0LearnCV	14
predict.L0Learn	15
print.L0Learn	16
Index	17

L0Learn-package	<i>A package for L0-regularized learning</i>
-----------------	--

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 λ and γ 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 ℓ 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.
- `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.

References

Hazimeh and Mazumder. Fast Best Subset Selection: Coordinate Descent and Local Combinatorial Optimization Algorithms. *Operations Research* (2020). <https://pubsonline.informs.org/doi/10.1287/opre.2019.1919>.

coef.L0Learn *Extract Solutions*

Description

Extracts a specific solution in the regularization path.

Usage

```
## 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

Value

A sparse Matrix of class `dgMatrix`, which contains the model coefficients. 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

```
# Generate synthetic data for this example
data <- GenSynthetic(n=100,p=20,k=10,seed=1)
X = data$X
y = data$y

# Fit an L0L2 Model with 3 values of Gamma ranging from 0.0001 to 10, using coordinate descent
fit <- L0Learn.fit(X, y, penalty="L0L2", nGamma=3, gammaMin=0.0001, gammaMax = 10)
print(fit)
# Extract the coefficients of the solution at lambda = 2.45513e-02 and gamma = 0.0001
coef(fit, lambda=2.45513e-02, 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 $y = XB + b_0 + e$.

Usage

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

Arguments

<code>n</code>	Number of samples
<code>p</code>	Number of features
<code>k</code>	Number of non-zeros in true vector of coefficients
<code>seed</code>	The seed used for randomly generating the data
<code>rho</code>	The threshold for setting values to 0. if $ X(i, j) > \text{rho} \Rightarrow X(i, j) <- 0$
<code>b0</code>	intercept value to translate y by.
<code>snr</code>	desired Signal-to-Noise ratio. This sets the magnitude of the error term 'e'. SNR is defined as $\text{SNR} = \text{Var}(XB)/\text{Var}(e)$

Value

A list containing: the data matrix X , the response vector y , the coefficients B , the error vector e , the intercept term b_0 .

Examples

```
data <- GenSynthetic(n=100, p=20, 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 (μ and SIG) to generate X. 3) Generate a vector B with every $\sim p/k$ 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,
  snr = 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) > \text{rho} \Rightarrow X(i, j) <- 0$
b0	intercept value to scale y by.
snr	desired Signal-to-Noise ratio. This sets the magnitude of the error term 'e'. SNR is defined as $\text{SNR} = \text{Var}(XB)/\text{Var}(e)$
mu	The mean for drawing from the Multivariate Normal Distribution. A scalar of vector of length p.
base_cor	The base correlation, A in $[i, j] = A^{ i-j }$.

Value

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

GenSyntheticLogistic *Generate Logistic Synthetic Data*

Description

Generates a synthetic dataset as follows: 1) Generate a data matrix, X , drawn from a multivariate Gaussian distribution with mean = 0, sigma = Sigma 2) Generate a vector B with k entries set to 1 and the rest are zeros. 3) Every coordinate y_i of the outcome vector y exists in $-1, 1^n$ is sampled independently from a Bernoulli distribution with success probability: $P(y_i = 1|x_i) = 1/(1 + \exp(-s \langle x_i, B \rangle))$ Source <https://arxiv.org/pdf/2001.06471.pdf> Section 5.1 Data Generation

Usage

```
GenSyntheticLogistic(
  n,
  p,
  k,
  seed,
  rho = 0,
  s = 1,
  sigma = NULL,
  shuffle_B = FALSE
)
```

Arguments

<code>n</code>	Number of samples
<code>p</code>	Number of features
<code>k</code>	Number of non-zeros in true vector of coefficients
<code>seed</code>	The seed used for randomly generating the data
<code>rho</code>	The threshold for setting values to 0. if $ X(i, j) > \text{rho} \Rightarrow X(i, j) \leftarrow 0$
<code>s</code>	Signal-to-noise parameter. As $s \rightarrow +\text{Inf}$, the data generated becomes linearly separable.
<code>sigma</code>	Correlation matrix, defaults to I .
<code>shuffle_B</code>	A boolean flag for whether or not to randomly shuffle the Beta vector, B . If <code>FALSE</code> , the first k entries in B are set to 1.

Value

A list containing: the data matrix X , the response vector y , the coefficients B .

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", "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.
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 and will be set to $\text{length}(\text{lambdaGrid})$.
gammaMax	The maximum value of Gamma when using the LOL2 penalty. For the LOL1 penalty this is automatically selected.
gammaMin	The minimum value of Gamma when using the LOL2 penalty. For the LOL1 penalty, the minimum value of gamma in the grid is set to $\text{gammaMin} * \text{gammaMax}$. 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 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 <i>i</i> th element (corresponding to the <i>i</i> th gamma) should be a decreasing sequence of lambda values which are used by the algorithm when fitting for the <i>i</i> th 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 <i>x</i> 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 <i>p</i> (number of columns of <i>X</i>) where lows[<i>i</i>] is the lower bound for coefficient <i>i</i> .
highs	Upper bounds for coefficients. Either a scalar for all coefficients to have the same bound or a vector of size <i>p</i> (number of columns of <i>X</i>) where highs[<i>i</i>] is the upper bound for coefficient <i>i</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 <i>i</i> th element is the sequence of cross-validation errors corresponding to the <i>i</i> th gamma value, i.e., the sequence cvMeans[[<i>i</i>]] corresponds to fit\$gamma[<i>i</i>]
cvSDs	This a list, where the <i>i</i> th element is a sequence of standard deviations for the cross-validation errors: cvSDs[[<i>i</i>]] corresponds to cvMeans[[<i>i</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=100,p=20,k=10,seed=1)
X = data$X
y = data$y
#'
# Perform 3-fold cross-validation on an L0L2 regression model with 3 values of
# Gamma ranging from 0.0001 to 10
fit <- L0Learn.cvfit(X, y, nFolds=3, seed=1, penalty="L0L2", maxSuppSize=20, nGamma=3,
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=2.45513e-02, gamma=0.0001)
# Apply the fitted model on X to predict the response
predict(fit, newx = X, lambda=2.45513e-02, 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,  
  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

<code>x</code>	The data matrix.
<code>y</code>	The response vector. For classification, we only support binary vectors.
<code>loss</code>	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.
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 and will be set to $\text{length}(\text{lambdaGrid})$.
gammaMax	The maximum value of Gamma when using the LOL2 penalty. For the LOL1 penalty this is automatically selected.
gammaMin	The minimum value of Gamma when using the LOL2 penalty. For the LOL1 penalty, the minimum value of gamma in the grid is set to $\text{gammaMin} * \text{gammaMax}$. 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 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 <i>i</i> th element (corresponding to the <i>i</i> th gamma) should be a decreasing sequence of lambda values which are used by the algorithm when fitting for the <i>i</i> th value of gamma (see the vignette for details).
excludeFirstK	This parameter takes non-negative integers. The first excludeFirstK features in <i>x</i> 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 <i>p</i> (number of columns of <i>X</i>) where lows[<i>i</i>] is the lower bound for coefficient <i>i</i> .
highs	Upper bounds for coefficients. Either a scalar for all coefficients to have the same bound or a vector of size <i>p</i> (number of columns of <i>X</i>) where highs[<i>i</i>] is the upper bound for coefficient <i>i</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 <i>i</i> th element of the list (i.e., a0[[<i>i</i>]]) is the sequence of intercepts corresponding to the <i>i</i> th gamma value (i.e., gamma[<i>i</i>]).
beta	This is a list of coefficient matrices. The <i>i</i> th element of the list is a <i>p</i> x length(lambda) matrix which corresponds to the <i>i</i> th gamma value. The <i>j</i> th column in each coefficient matrix is the vector of coefficients for the <i>j</i> th lambda value.
lambda	This is the list of lambda sequences used in fitting the model. The <i>i</i> th element of lambda (i.e., lambda[[<i>i</i>]]) is the sequence of Lambda values corresponding to the <i>i</i> th 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 <i>i</i> th element of the list is a sequence of support sizes (i.e., number of non-zero coefficients) corresponding to the <i>i</i> th gamma value.
converged	This is a list of sequences for checking whether the algorithm has converged at every grid point. The <i>i</i> th element of the list is a sequence corresponding to the <i>i</i> th value of gamma, where the <i>j</i> th element in each sequence indicates whether the algorithm has converged at the <i>j</i> th value of lambda.

Examples

```
# Generate synthetic data for this example
data <- GenSynthetic(n=100,p=20,k=10,seed=1)
X = data$X
y = data$y

# Fit an L0 regression model with a maximum of 20 non-zeros using coordinate descent (CD)
fit1 <- L0Learn.fit(X, y, penalty="L0", maxSuppSize=20)
print(fit1)
```

```

# Extract the coefficients at lambda = 2.28552e-02
coef(fit1, lambda=2.28552e-02)
# Apply the fitted model on X to predict the response
predict(fit1, newx = X, lambda=2.28552e-02)

# Fit an L0 regression model with a maximum of 20 non-zeros using CD and local search
fit2 <- L0Learn.fit(X, y, penalty="L0", algorithm="CDPSI", maxSuppSize=20)
print(fit2)

# Fit an L0L2 regression model with 3 values of Gamma ranging from 0.0001 to 10, using CD
fit3 <- L0Learn.fit(X, y, penalty="L0L2", maxSuppSize=20, nGamma=3, gammaMin=0.0001, gammaMax = 10)
print(fit3)
# Extract the coefficients at lambda = 2.45513e-02 and gamma = 0.0001
coef(fit3, lambda=2.45513e-02, gamma=0.0001)
# Apply the fitted model on X to predict the response
predict(fit3, newx = X, lambda=2.45513e-02, 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=10)
print(fit4)

```

plot.L0Learn

Plot Regularization Path

Description

Plots the regularization path for a given gamma.

Usage

```

## 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

Value

A ggplot object.

Examples

```
# Generate synthetic data for this example
data <- GenSynthetic(n=100,p=20,k=10,seed=1)
X = data$X
y = data$y
# Fit an L0 Model
fit <- L0Learn.fit(X, y, penalty="L0")
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 = 0, ...)
```

Arguments

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

Value

A ggplot object.

Examples

```
# Generate synthetic data for this example
data <- GenSynthetic(n=100,p=20,k=10,seed=1)
X = data$X
y = data$y

# Perform 3-fold cross-validation on an L0L2 Model with 3 values of
# Gamma ranging from 0.0001 to 10
fit <- L0Learn.cvfit(X, y, nFolds=3, seed=1, penalty="L0L2",
maxSuppSize=20, nGamma=3, 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 = NULL, gamma = NULL, ...)

## 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 <code>print(fit)</code> .
gamma	The value of gamma to use for prediction. A summary of the gammas in the regularization path can be obtained using <code>print(fit)</code> .
...	ignore

Value

A Matrix of class `dgeMatrix`, which contains the model predictions. 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

```
# Generate synthetic data for this example
data <- GenSynthetic(n=100,p=20,k=10,seed=1)
X = data$X
y = data$y

# Fit an L0L2 Model with 3 values of Gamma ranging from 0.0001 to 10, using coordinate descent
fit <- L0Learn.fit(X,y, penalty="L0L2", nGamma=3, gammaMin=0.0001, gammaMax = 10)
print(fit)
# Apply the fitted model with lambda=2.45513e-02 and gamma=0.0001 on X to predict the response
predict(fit, newx = X, lambda=2.45513e-02, 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.LOLearn	<i>Print LOLearn.fit object</i>
---------------	---------------------------------

Description

Prints a summary of LOLearn.fit

Usage

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

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

Arguments

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

Value

Prints a summary of the models to the console.

Index

`coef`, [2](#)
`coef.L0Learn`, [3](#)
`coef.L0LearnCV` (`coef.L0Learn`), [3](#)

`GenSynthetic`, [4](#)
`GenSyntheticHighCorr`, [5](#)
`GenSyntheticLogistic`, [6](#)

`L0Learn`-package, [2](#)
`L0Learn.cvfit`, [2](#), [7](#)
`L0Learn.fit`, [2](#), [9](#), [10](#)

`plot`, [2](#)
`plot.L0Learn`, [13](#)
`plot.L0LearnCV`, [14](#)
`predict`, [2](#)
`predict.L0Learn`, [15](#)
`predict.L0LearnCV` (`predict.L0Learn`), [15](#)
`print`, [2](#)
`print.L0Learn`, [16](#)
`print.L0LearnCV` (`print.L0Learn`), [16](#)