Package ‘gelnet’

October 13, 2022

Version 1.2.1

Date 2015-10-16

License GPL (>= 3)

Title Generalized Elastic Nets

Description Implements several extensions of the elastic net regularization scheme. These extensions include individual feature penalties for the L1 term, feature-feature penalties for the L2 term, as well as translation coefficients for the latter.

Author Artem Sokolov

Maintainer Artem Sokolov <artem.sokolov@gmail.com>

Depends R (>= 3.1.0)

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2016-04-05 08:14:29

R topics documented:

adj2lapl *Generate a graph Laplacian*

Description

Generates a graph Laplacian from the graph adjacency matrix.

Usage

```
adj2lapl(A)
```

Arguments

A	n-by-n adjacency matrix for a graph with n nodes

Details

A graph Laplacian is defined as: $l_{i,j} = deg(v_i)$, if $i = j$; $l_{i,j} = -1$, if $i \neq j$ and v_i is adjacent to v_j; and $l_{i,j} = 0$, otherwise

Value

The n-by-n Laplacian matrix of the graph

See Also

adj2nlapl

adj2nlapl *Generate a normalized graph Laplacian*

Description

Generates a normalized graph Laplacian from the graph adjacency matrix.

Usage

```
adj2nlapl(A)
```

Arguments

A	n-by-n adjacency matrix for a graph with n nodes

Details

A normalized graph Laplacian is defined as: $l_{i,j} = 1$, if $i = j$; $l_{i,j} = -1/\sqrt{deg(v_i)deg(v_j)}$, if $i \neq j$ and v_i is adjacent to v_j; and $l_{i,j} = 0$, otherwise

Value

The n-by-n Laplacian matrix of the graph

See Also

`adj2nlapl`

`gelnet` *GELnet for linear regression, binary classification and one-class problems.*

Description

Infers the problem type and learns the appropriate GELnet model via coordinate descent.

Usage

```
gelnet(X, y, l1, l2, nFeats = NULL, a = rep(1, n), d = rep(1, p),
  P = diag(p), m = rep(0, p), max.iter = 100, eps = 1e-05,
  w.init = rep(0, p), b.init = NULL, fix.bias = FALSE, silent = FALSE,
  balanced = FALSE, nonneg = FALSE)
```

Arguments

`X`	n-by-p matrix of n samples in p dimensions
`y`	n-by-1 vector of response values. Must be numeric vector for regression, factor with 2 levels for binary classification, or NULL for a one-class task.
`l1`	coefficient for the L1-norm penalty
`l2`	coefficient for the L2-norm penalty
`nFeats`	alternative parameterization that returns the desired number of non-zero weights. Takes precedence over l1 if not NULL (default: NULL)
`a`	n-by-1 vector of sample weights (regression only)
`d`	p-by-1 vector of feature weights
`P`	p-by-p feature association penalty matrix
`m`	p-by-1 vector of translation coefficients
`max.iter`	maximum number of iterations
`eps`	convergence precision
`w.init`	initial parameter estimate for the weights
`b.init`	initial parameter estimate for the bias term
`fix.bias`	set to TRUE to prevent the bias term from being updated (regression only) (default: FALSE)
`silent`	set to TRUE to suppress run-time output to stdout (default: FALSE)
`balanced`	boolean specifying whether the balanced model is being trained (binary classification only) (default: FALSE)
`nonneg`	set to TRUE to enforce non-negativity constraints on the weights (default: FALSE)

Details

The method determines the problem type from the labels argument y. If y is a numeric vector, then a regression model is trained by optimizing the following objective function:

$$\frac{1}{2n}\sum_i a_i(y_i - (w^T x_i + b))^2 + R(w)$$

If y is a factor with two levels, then the function returns a binary classification model, obtained by optimizing the following objective function:

$$-\frac{1}{n}\sum_i y_i s_i - \log(1 + \exp(s_i)) + R(w)$$

where

$$s_i = w^T x_i + b$$

Finally, if no labels are provided (y == NULL), then a one-class model is constructed using the following objective function:

$$-\frac{1}{n}\sum_i s_i - \log(1 + \exp(s_i)) + R(w)$$

where

$$s_i = w^T x_i$$

In all cases, the regularizer is defined by

$$R(w) = \lambda_1 \sum_j d_j |w_j| + \frac{\lambda_2}{2}(w - m)^T P(w - m)$$

The training itself is performed through cyclical coordinate descent, and the optimization is terminated after the desired tolerance is achieved or after a maximum number of iterations.

Value

A list with two elements:

w p-by-1 vector of p model weights

b scalar, bias term for the linear model (omitted for one-class models)

See Also

`gelnet.lin.obj`, `gelnet.logreg.obj`, `gelnet.oneclass.obj` `gelnet.cv` *k-fold cross-validation for parameter tuning.*

Description

Performs k-fold cross-validation to select the best pair of the L1- and L2-norm penalty values.

Usage

```
gelnet.cv(X, y, nL1, nL2, nFolds = 5, a = rep(1, n), d = rep(1, p),
  P = diag(p), m = rep(0, p), max.iter = 100, eps = 1e-05,
  w.init = rep(0, p), b.init = 0, fix.bias = FALSE, silent = FALSE,
  balanced = FALSE)
```

Arguments

`X`	n-by-p matrix of n samples in p dimensions
`y`	n-by-1 vector of response values. Must be numeric vector for regression, factor with 2 levels for binary classification, or NULL for a one-class task.
`nL1`	number of values to consider for the L1-norm penalty
`nL2`	number of values to consider for the L2-norm penalty
`nFolds`	number of cross-validation folds (default:5)
`a`	n-by-1 vector of sample weights (regression only)
`d`	p-by-1 vector of feature weights
`P`	p-by-p feature association penalty matrix
`m`	p-by-1 vector of translation coefficients
`max.iter`	maximum number of iterations
`eps`	convergence precision
`w.init`	initial parameter estimate for the weights
`b.init`	initial parameter estimate for the bias term
`fix.bias`	set to TRUE to prevent the bias term from being updated (regression only) (default: FALSE)
`silent`	set to TRUE to suppress run-time output to stdout (default: FALSE)
`balanced`	boolean specifying whether the balanced model is being trained (binary classification only) (default: FALSE)

Details

Cross-validation is performed on a grid of parameter values. The user specifies the number of values to consider for both the L1- and the L2-norm penalties. The L1 grid values are equally spaced on [0, L1s], where L1s is the smallest meaningful value of the L1-norm penalty (i.e., where all the model weights are just barely zero). The L2 grid values are on a logarithmic scale centered on 1.

Value

A list with the following elements:

l1 the best value of the L1-norm penalty

l2 the best value of the L2-norm penalty

w p-by-1 vector of p model weights associated with the best (l1,l2) pair.

b scalar, bias term for the linear model associated with the best (l1,l2) pair. (omitted for one-class models)

perf performance value associated with the best model. (Likelihood of data for one-class, AUC for binary classification, and -RMSE for regression)

See Also

`gelnet`

`gelnet.ker` *Kernel models for linear regression, binary classification and one-class problems.*

Description

Infers the problem type and learns the appropriate kernel model.

Usage

```
gelnet.ker(K, y, lambda, a, max.iter = 100, eps = 1e-05, v.init = rep(0,
  nrow(K)), b.init = 0, fix.bias = FALSE, silent = FALSE,
  balanced = FALSE)
```

Arguments

`K`	n-by-n matrix of pairwise kernel values over a set of n samples
`y`	n-by-1 vector of response values. Must be numeric vector for regression, factor with 2 levels for binary classification, or NULL for a one-class task.
`lambda`	scalar, regularization parameter
`a`	n-by-1 vector of sample weights (regression only)
`max.iter`	maximum number of iterations (binary classification and one-class problems only)
`eps`	convergence precision (binary classification and one-class problems only)
`v.init`	initial parameter estimate for the kernel weights (binary classification and one-class problems only)
`b.init`	initial parameter estimate for the bias term (binary classification only)
`fix.bias`	set to TRUE to prevent the bias term from being updated (regression only) (default: FALSE)

`silent`	set to TRUE to suppress run-time output to stdout (default: FALSE)
`balanced`	boolean specifying whether the balanced model is being trained (binary classification only) (default: FALSE)

Details

The entries in the kernel matrix K can be interpreted as dot products in some feature space ϕ. The corresponding weight vector can be retrieved via $w = \sum_i v_i \phi(x_i)$. However, new samples can be classified without explicit access to the underlying feature space:

$$w^T \phi(x) + b = \sum_i v_i \phi^T(x_i)\phi(x) + b = \sum_i v_i K(x_i, x) + b$$

The method determines the problem type from the labels argument y. If y is a numeric vector, then a ridge regression model is trained by optimizing the following objective function:

$$\frac{1}{2n} \sum_i a_i (z_i - (w^T x_i + b))^2 + w^T w$$

If y is a factor with two levels, then the function returns a binary classification model, obtained by optimizing the following objective function:

$$-\frac{1}{n} \sum_i y_i s_i - \log(1 + \exp(s_i)) + w^T w$$

where

$$s_i = w^T x_i + b$$

Finally, if no labels are provided (y == NULL), then a one-class model is constructed using the following objective function:

$$-\frac{1}{n} \sum_i s_i - \log(1 + \exp(s_i)) + w^T w$$

where

$$s_i = w^T x_i$$

In all cases, $w = \sum_i v_i \phi(x_i)$ and the method solves for v_i.

Value

A list with two elements:

v n-by-1 vector of kernel weights

b scalar, bias term for the linear model (omitted for one-class models)

See Also

`gelnet`

gelnet.lin.obj *Linear regression objective function value*

Description

Evaluates the linear regression objective function value for a given model. See details.

Usage

```
gelnet.lin.obj(w, b, X, z, l1, l2, a = rep(1, nrow(X)), d = rep(1, ncol(X)),
  P = diag(ncol(X)), m = rep(0, ncol(X)))
```

Arguments

w	p-by-1 vector of model weights
b	the model bias term
X	n-by-p matrix of n samples in p dimensions
z	n-by-1 response vector
l1	L1-norm penalty scaling factor λ_1
l2	L2-norm penalty scaling factor λ_2
a	n-by-1 vector of sample weights
d	p-by-1 vector of feature weights
P	p-by-p feature-feature penalty matrix
m	p-by-1 vector of translation coefficients

Details

Computes the objective function value according to

$$\frac{1}{2n} \sum_i a_i (z_i - (w^T x_i + b))^2 + R(w)$$

where

$$R(w) = \lambda_1 \sum_j d_j |w_j| + \frac{\lambda_2}{2} (w - m)^T P (w - m)$$

Value

The objective function value.

See Also

gelnet

`gelnet.logreg.obj` *Logistic regression objective function value*

Description

Evaluates the logistic regression objective function value for a given model. See details. Computes the objective function value according to

$$-\frac{1}{n}\sum_i y_i s_i - \log(1+\exp(s_i)) + R(w)$$

where

$$s_i = w^T x_i + b$$

$$R(w) = \lambda_1 \sum_j d_j |w_j| + \frac{\lambda_2}{2}(w-m)^T P(w-m)$$

When balanced is TRUE, the loss average over the entire data is replaced with averaging over each class separately. The total loss is then computes as the mean over those per-class estimates.

Usage

```
gelnet.logreg.obj(w, b, X, y, l1, l2, d = rep(1, ncol(X)),
  P = diag(ncol(X)), m = rep(0, ncol(X)), balanced = FALSE)
```

Arguments

`w`	p-by-1 vector of model weights
`b`	the model bias term
`X`	n-by-p matrix of n samples in p dimensions
`y`	n-by-1 binary response vector sampled from 0,1
`l1`	L1-norm penalty scaling factor λ_1
`l2`	L2-norm penalty scaling factor λ_2
`d`	p-by-1 vector of feature weights
`P`	p-by-p feature-feature penalty matrix
`m`	p-by-1 vector of translation coefficients
`balanced`	boolean specifying whether the balanced model is being evaluated

Value

The objective function value.

See Also

`gelnet`

`gelnet.oneclass.obj` *One-class regression objective function value*

Description

Evaluates the one-class objective function value for a given model See details.

Usage

```
gelnet.oneclass.obj(w, X, l1, l2, d = rep(1, ncol(X)), P = diag(ncol(X)),
  m = rep(0, ncol(X)))
```

Arguments

`w`	p-by-1 vector of model weights
`X`	n-by-p matrix of n samples in p dimensions
`l1`	L1-norm penalty scaling factor λ_1
`l2`	L2-norm penalty scaling factor λ_2
`d`	p-by-1 vector of feature weights
`P`	p-by-p feature-feature penalty matrix
`m`	p-by-1 vector of translation coefficients

Details

Computes the objective function value according to

$$-\frac{1}{n}\sum_i s_i - \log(1 + \exp(s_i)) + R(w)$$

where

$$s_i = w^T x_i$$

$$R(w) = \lambda_1 \sum_j d_j |w_j| + \frac{\lambda_2}{2}(w-m)^T P(w-m)$$

Value

The objective function value.

See Also

gelnet

`L1.ceiling` *The largest meaningful value of the L1 parameter*

Description

Computes the smallest value of the LASSO coefficient L1 that leads to an all-zero weight vector for a given linear regression problem.

Usage

```
L1.ceiling(X, y, a = rep(1, nrow(X)), d = rep(1, ncol(X)),
  P = diag(ncol(X)), m = rep(0, ncol(X)), l2 = 1, balanced = FALSE)
```

Arguments

`X`	n-by-p matrix of n samples in p dimensions
`y`	n-by-1 vector of response values. Must be numeric vector for regression, factor with 2 levels for binary classification, or NULL for a one-class task.
`a`	n-by-1 vector of sample weights (regression only)
`d`	p-by-1 vector of feature weights
`P`	p-by-p feature association penalty matrix
`m`	p-by-1 vector of translation coefficients
`l2`	coefficient for the L2-norm penalty
`balanced`	boolean specifying whether the balanced model is being trained (binary classification only) (default: FALSE)

Details

The cyclic coordinate descent updates the model weight w_k using a soft threshold operator $S(\cdot, \lambda_1 d_k)$ that clips the value of the weight to zero, whenever the absolute value of the first argument falls below $\lambda_1 d_k$. From here, it is straightforward to compute the smallest value of λ_1, such that all weights are clipped to zero.

Value

The largest meaningful value of the L1 parameter (i.e., the smallest value that yields a model with all zero weights)

Index