R documentation

of all in 'man'

August 8, 2023

R topics documented:

`polysf-package` *polysf: Strictly feasible representation for a polytope*

Description

Provide a strictly feasible representation for a polytope with implicit equality constraints, and a sampler on the degenerated polytope. The same functionality for Lasso solutions in a non-uniqueness regime, as a special case of the degenerated polytope, is provided as well.

Author(s)

Xiaozhu Zhang

References

Telgen, J. (1982). "Minimal representation of convex polyhedral sets". Journal of Optimization Theory and Applications 38.1, pp. 1–24.

Drusvyatskiy, Dmitriy, Henry Wolkowicz, et al. (2017). "The many faces of degeneracy in conic optimization". Foundations and Trends® in Optimization 3.2, pp. 77–170.

Tibshirani, Ryan J (2013). "The lasso problem and uniqueness". Electronic Journal of Statistics 7, pp. 1456–1490.

`add_to_full` *Extend a full-row-rank matrix to a full-rank square matrix*

Description

Given any full-row-rank matrix `N` with `n=ncol(N)` and `r=rank(N)`. This function adds `n-r` independent rows to the matrix `N`, such that the resulting new matrix is a full-rank square matrix.

Usage

```
add_to_full(N)
```

Arguments

`N`	a full-row-rank matrix.

Value

A full-rank square matrix whose first `r` rows come from `N`.

Examples

```
# generate a full-row-rank matrix
N = matrix(rnorm(10), nrow = 2)

# add 3 independent rows to the matrix to form a full-rank square matrix
add_to_full(N)
```

`equi_index_lasso` *The equi-correlation set of a Lasso problem*

Description

This function finds the equi-correlation set of a Lasso problem. Given the design matrix $X \in \mathbb{R}^{n\times d}$, the response variable $y \in \mathbb{R}^n$, the tuning parameter λ, and the fitted coefficient $\hat{\beta}$, the equi-correlation set is defined as
$\mathcal{E} = \{i \in \{1, \dots, d\} : |X_i^\top (y - X\hat{\beta})| = n\lambda\}$.

Usage

```
equi_index_lasso(X, y, lambda, beta, tol = 0.01)
```

Arguments

`X`	the design matrix.
`y`	a vector denoting the response variable.
`lambda`	a numeric denoting the tuning parameter.
`beta`	a vector denoting a particular Lasso solution.
`tol`	a tolerance numeric greater than 0, which must cover the difference between λ and $\lvert X_i^\top (y - X\hat{\beta})\rvert$.

Details

The equi-correlation set includes indices for all relevant features. The detection of $\mathcal{E}$ depends on the accuracy of $\hat{\beta}$ as the minimum point of the Lasso loss function. The argument `tol` must be chosen to transcend the gap between $n\lambda$ and $|X_i^\top(y - X\hat{\beta})|$. If the tolerance `tol` is chosen too large, some irrelevant features may be included and thus the subsequent facial reduction procedure would be decelerated; on the other hand, if the tolerance is chosen too small, some relevant features may be ignored which leads to a wrong strictly feasible representation. In conclusion, a too tight tolerance is more detrimental than a too loose one. An accurate $\hat{\beta}$ (`beta`) and a slightly loose `tol` are always helpful.

Value

An integer vector containing all elements of the equi-correlation set.

Examples

```
# generate a sparse data set with non-unique Lasso solns
set.seed(1234)
n = 1000
d = 20
s = 5
rho = 0.01
Sigma = matrix(0, nrow = d, ncol = d)
for(i in 1:d) {
  for(j in 1:d) {
    Sigma[i,j] = rho^abs(i-j)
  }
}
X = mvtnorm::rmvnorm(n = n, mean = rep(0, d), sigma = Sigma)
X[,2] = -X[,1]
X[,3] = X[,1]
beta = c(1, 0, 0, rep(1, s-3), rep(0, d-s))
epsilon = rnorm(n, mean = 0, sd = 0.1)
y = X %*% beta + epsilon
# find one particular solution
lambda = 0.01
model = glmnet::glmnet(X, y, family = "gaussian", alpha = 1, lambda = lambda,
                       intercept = FALSE, standardize = FALSE)
beta = as.numeric( model$beta )

# find the equi-correlation set
equi_index_lasso(X, y, lambda, beta, tol = 1e-5)
```

`ind_rows` *Select independent rows of a matrix*

Description

Given any matrix `M`, this function selects the first `r=rank(M)` independent rows in `M` which form a row basis of `M`.

Usage

```
ind_rows(M)
```

Arguments

M	a matrix.

Value

The function `ind_rows` returns a list containing the following components:

N	a matrix with `r=rank(M)` rows. Its rows come from the selected independent rows in `M`.
idx	a logical vector. An element is `TRUE` if the corresponding row in `M` is selected, `FALSE` otherwise.

Examples

```
# generate a matrix M of rank 2 with dependent rows
M = matrix(rnorm(16), nrow = 4)
M[2,] = rnorm(1) * M[1,] + rnorm(1) * M[3,]
M[4,] = rnorm(1) * M[2,] + rnorm(1) * M[3,]

# select the first 2 independent rows
ind_rows(M)
```

---

print.sf_rep *Print the results of sf_rep*

Description

The `print` method for class `sf_rep`.

Usage

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

Arguments

x	an S3 object of class `sf_rep`.
...	further arguments passed to or from other methods.

Value

The output from `print`, summarizing the implicit equality identification, and the intrinsic dimension of `x`.

print.sf_rep_lasso — *Print the results of sf_rep_lasso*

Description

The `print` method for class `sf_rep_lasso`.

Usage

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

Arguments

`x`	an S3 object of class `sf_rep_lasso`.
`...`	further arguments passed to or from other methods.

Value

The output from `print`, summarizing the equi-correlation set, the sign of the Theta set, and the intrinsic dimension of `x`.

sample_lasso — *Sample iid Lasso solutions in a non-uniqueness regime*

Description

This function samples iid points from a set with non-unique elements, $\Theta = \{\theta \in \mathbb{R}^d : \|\theta\|_1 = \|\tilde{\theta}\|_1,\ X\tilde{\theta} = X\theta\}$ whose strictly feasible representation is given by a object of class `sf_rep_lasso`, where $\tilde{\theta}$ is a particular solution.

Usage

```
sample_lasso(obj, npoints, random_walk = NULL, distribution = NULL)
```

Arguments

`obj`	an S3 object of class `sf_rep_lasso`.
`npoints`	the number of points that the function is going to sample.
`random_walk`	an optional list that declares the random walk and some related parameters. See the argument `random_walk` in `volesti::sample_points` for details.
`distribution`	an optional list that declares the target density and some related parameters. See the argument `distribution` in `volesti::sample_points` for details.

Value

A matrix with `npoints` rows and `ncol(X)` columns. Each row is a sample point from the polytope Θ.

Examples

```
# generate a sparse data set with non-unique Lasso solns
set.seed(1234)
n = 1000
d = 20
s = 5
rho = 0.01
Sigma = matrix(0, nrow = d, ncol = d)
for(i in 1:d) {
  for(j in 1:d) {
    Sigma[i,j] = rho^abs(i-j)
  }
}
X = mvtnorm::rmvnorm(n = n, mean = rep(0, d), sigma = Sigma)
X[,2] = -X[,1]
X[,3] = X[,1]
beta = c(1, 0, 0, rep(1, s-3), rep(0, d-s))
epsilon = rnorm(n, mean = 0, sd = 0.1)
y = X %*% beta + epsilon
# find one particular solution
lambda = 0.01
model = glmnet::glmnet(X, y, family = "gaussian", alpha = 1, lambda = lambda,
                       intercept = FALSE, standardize = FALSE)
beta_fit = as.numeric( model$beta )

# generate an S3 object of class sf_poly_lasso
equidx = equi_index_lasso(X, y, lambda, beta_fit, tol = 1e-3)
fit = sf_rep_lasso(X, beta_fit, equidx = equidx)

# sample 100 uniformly distributed points
# from the predicted theta set
sample_lasso(fit, npoints = 100)
```

---

sample_polytope *Sample uniformly or normally distributed points from a polytope*

Description

This function samples uniformly or normally distributed points from a polytope $\mathcal{S} = \{x \in \mathbb{R}^n : Ax \leq b\}$ whose strictly feasible representation is given by a object of class `sf_rep`.

Usage

```
sample_polytope(obj, npoints, random_walk = NULL, distribution = NULL)
```

Arguments

`obj`	an S3 object of class `sf_rep`.
`npoints`	the number of points that the function is going to sample.
`random_walk`	an optional list that declares the random walk and some related parameters. See the argument `random_walk` in `volesti::sample_points` for details.
`distribution`	an optional list that declares the target density and some related parameters. See the argument `distribution` in `volesti::sample_points` for details.

Value

A matrix with `npoints` rows and `ncol(A)` columns. Each row is a sample point from the polytope $\mathcal{S}$.

Examples

```
# generate an S3 object of class sf_poly
A = matrix(c(1, 1, -1, -1, -1, 0, 0, -1), byrow = TRUE, nrow = 4)
b = c(1, -1, 0, 0)
fit = sf_rep(A, b)

# sample 100 uniformly distributed points from Ax<=b
sample_polytope(fit, npoints = 100)
```

`sf_rep` *Strictly feasible representation of a polytope*

Description

This function finds a strictly feasible representation of the given polytope $\mathcal{S} = \{x \in \mathbb{R}^n : Ax \leq b\}$ within a subspace of its intrinsic dimension.

Usage

```
sf_rep(A, b)
```

Arguments

`A`	the matrix A of the polytope $\mathcal{S}$.
`b`	the matrix b of the polytope $\mathcal{S}$.

Details

Given a polytope $\mathcal{S} = \{x \in \mathbb{R}^n : Ax \leq b\}$, namely a bounded intersection of finite number of half-spaces where each half-space is represented by a constraint $\{x : a_i^\top x \leq b_i\}$. A polytope $\mathcal{S}$ with implicit equality constraints is of measure 0 in $\mathbb{R}^n$ and has no strictly feasible points (or interior points) inside $\mathcal{S}$. This degeneration may fail standard sampling algorithms over a polytope, including the Ball Walk, the Hit-and-Run, and the Dikin Walk.

The function `sf_rep` finds a strictly feasible representation of any polytope $\mathcal{S}$ with at least one explicit inequality constraint. Specifically, this function identifies the implicit equality constraints of $\mathcal{S}$, presents its intrinsic dimension $\dim\mathcal{S}$, and provides an equivalent strictly feasible polytope $\mathcal{S}^* = \{y \in \mathbb{R}^{\dim\mathcal{S}} : \Gamma y \leq \gamma\}$. In addition, a transformation matrix pair (T, d) is given to convert $\mathcal{S}^*$ back to $\mathcal{S}$: For any $y \in \mathcal{S}^*$, we have $T \begin{bmatrix} d \\ y \end{bmatrix} \in \mathcal{S}$.

Value

The function `sf_rep` returns an S3 object of class sf_rep containing the following components:

`A`	the argument `A`.
`b`	the argument `b`.
`I`	an integer vector containing the indices of implicit constraints.
`indim`	an integer denoting the intrinsic dimension of $\mathcal{S}$.
`Gamma`	the matrix Γ of the polytope $\mathcal{S}^*$.
`gamma`	the vector γ of the polytope $\mathcal{S}^*$.
`Tm`	the matrix T in the transformation pair (T, d).
`d`	the vector d in the transformation pair (T, d).

Examples

```
# generate A and b of the polytope
A = matrix(c(1, 1, -1, -1, -1, 0, 0, -1), byrow = TRUE, nrow = 4)
b = c(1, -1, 0, 0)

# find the strictly feasible representation
sf_rep(A, b)
```

`sf_rep_lasso` *Strictly feasible representation of Lasso in a non-uniqueness regime*

Description

Given the design matrix $X \in \mathbb{R}^{n \times d}$ and a particular solution $\tilde{\theta}$, this function finds a strictly feasible representation of the set $\Theta = \{\theta \in \mathbb{R}^d : \|\theta\|_1 = \|\tilde{\theta}\|_1,\ X\tilde{\theta} = X\theta\}$ which can be proven a polytope with implicit equality constraints.

Usage

```
sf_rep_lasso(
  X,
  beta,
  type = NULL,
  y = NULL,
  lambda = NULL,
  tol = 0.01,
  equidx = NULL
)
```

Arguments

`X`	the design matrix.
`beta`	a vector denoting a particular solution.
`type`	a character, `"hat"` if the set Θ is the predicted theta set; `"star"` if the set Θ is the true theta set; can be left `NULL` if `equidx` is provided.

y	a vector denoting the response variable. Need be provided only when `type = "hat"`.
lambda	a numeric denoting the tuning parameter. Need be provided only when `type = "hat"`.
tol	a tolerance numeric greater than 0. Need be provided only when `type = "hat"` or `type = "star"`. See `tol` in `polysf::equi_index_lasso` for details.
equidx	an optional integer vector containing all elements of the equi-correlation set.

Details

The function `sf_rep_lasso` can deal with two types of theta set:

1. The predicted theta set:
   $\hat{\Theta} = \{\theta \in \mathbb{R}^d : \theta \in \arg\min_{\vartheta} \frac{1}{2n}\|X\vartheta - y\|_2^2 + \lambda\|\vartheta\|_1\}$;
   It can be shown that $\hat{\Theta}$ is the same as Θ when the particular solution $\tilde{\theta} \in \hat{\Theta}$.
2. The true theta set: given a particular true coefficient θ^*,
   $\Theta^* = \{\theta \in \mathbb{R}^d : \|\theta\|_1 = \|\theta^*\|_1,\ X\theta = X\theta^*\}$.

We use the generic notation Θ with the particular solution $\tilde{\theta}$ to denote both types. Under certain conditions, Θ may contain non-unique (and infinitely many) elements. For both types the steps to obtaining their strictly feasible representation are exactly the same. However, there are some caveats in terms of determining the equi-correlation set: $(X, \tilde{\beta}, y, \lambda)$ are required for the predicted theta set, while only $(X, \tilde{\beta})$ are needed for the true theta set.

Essentially, the strictly feasible representation of Θ is obtained through facial reduction restricted to the equi-correlation set, but the procedure can be even simplified due to the special structure of $(\Theta, \tilde{\theta})$. The result of facial reduction reveals the sign of the theta set which is given by

$$\mathrm{sgn}(\Theta)_j = \begin{cases} 1, & \theta_j > 0\ \exists\theta \in \Theta, \\ -1, & \theta_j < 0\ \exists\theta \in \Theta, \\ 0, & \theta_j = 0\ \forall\theta \in \Theta. \end{cases}$$

Note that all points of Θ must be placed in same the orthants so $\mathrm{sgn}(\Theta)$ is well-defined.

The strictly feasible representation of Θ is given by $\mathcal{K} = \{y \in \mathbb{R}^{\dim\Theta} : \Gamma y \leq \gamma\}$ where $\dim\Theta$ is the intrinsic dimension of Θ. Note that there exists a bijection between Θ and $\mathcal{K}$ based on the transformation pair (T, ν). Particularly, for any $y \in \Theta$, we have $T\begin{bmatrix}\nu \\ y\end{bmatrix} \in \Theta$.

Value

The function `sf_rep_lasso` returns an S3 object of class `sf_rep_lasso` containing the following components:

X	the argument `X`.
beta	the argument `beta`.
equidx	the equi-correlation set either from input or found based on `type`, `y`, `lambda`, and `tol`.
I	an integer vector for possible future sampling.
signs	an integer vector of length d denoting the signs of Θ. For each feature, +1 means the feature is active and its coefficient is always non-negative; 0 means the feature is inactive; -1 means the feature is active and its coefficient is always non-positive.

indim	an integer denoting the intrinsic dimension of Θ.
Gamma	the matrix Γ of the polytope $\mathcal{K}$.
gamma	the matrix γ of the polytope $\mathcal{K}$.
Tm	the matrix T in the transofrmation pair (T, ν).
nv	the matrix ν in the transofrmation pair (T, ν).

Examples

```
# generate a sparse data set with non-unique Lasso solns
set.seed(1234)
n = 1000
d = 20
s = 5
rho = 0.01
Sigma = matrix(0, nrow = d, ncol = d)
for(i in 1:d) {
  for(j in 1:d) {
    Sigma[i,j] = rho^abs(i-j)
  }
}
X = mvtnorm::rmvnorm(n = n, mean = rep(0, d), sigma = Sigma)
X[,2] = -X[,1]
X[,3] = X[,1]
beta = c(1, 0, 0, rep(1, s-3), rep(0, d-s))
epsilon = rnorm(n, mean = 0, sd = 0.1)
y = X %*% beta + epsilon
# find one particular solution
lambda = 0.01
model = glmnet::glmnet(X, y, family = "gaussian", alpha = 1, lambda = lambda,
                       intercept = FALSE, standardize = FALSE)
beta_fit = as.numeric( model$beta )

# eg1: find the SF Rep of the predicted theta set
# with pre-calculated equi-correlation set
equidx = equi_index_lasso(X, y, lambda, beta_fit, tol = 1e-3)
sf_rep_lasso(X, beta_fit, equidx = equidx)

# without pre-calculated equi-correlation set
sf_rep_lasso(X, beta_fit, type = "hat", y = y, lambda = lambda, tol = 1e-3)

# eg2: find the SF Rep of the true theta set
sf_rep_lasso(X, beta, type = "star", tol = 1e-5)
```

Index