Package 'sparsediscrim'

July 2, 2014

Title Sparse and Regularized Discriminant Analysis

Version 0.2

Date 2014-03-31

Author John A. Ramey <johnramey@gmail.com>

Maintainer John A. Ramey <johnramey@gmail.com>

Description A collection of sparse and regularized discriminant analysis methods intended for small-sample, high-dimensional data sets. The package features the High-Dimensional Regularized Discriminant Analysis classifier.

Imports corpcor, bdsmatrix, mvtnorm

License MIT + file LICENSE

URL https://github.com/ramey/sparsediscrim, http://ramhiser.com

NeedsCompilation no

Repository CRAN

Date/Publication 2014-03-31 16:34:04

R topics documented:

`center_data` *Centers the observations in a matrix by their respective class sample means*

Description

Centers the observations in a matrix by their respective class sample means

Usage

```
center_data(x, y)
```

Arguments

`x`	matrix containing the training data. The rows are the sample observations, and the columns are the features.
`y`	vector of class labels for each training observation

Value

matrix with observations centered by its corresponding class sample mean

cov_autocorrelation *Generates a $p \times p$ autocorrelated covariance matrix*

Description

This function generates a $p \times p$ autocorrelated covariance matrix with autocorrelation parameter `rho`. The variance `sigma2` is constant for each feature and defaulted to 1.

Usage

```
cov_autocorrelation(p, rho, sigma2 = 1)
```

Arguments

`p`	the size of the covariance matrix
`rho`	the autocorrelation parameter. Must be less than 1 in absolute value.
`sigma2`	the variance of each feature

Details

The autocorrelated covariance matrix is defined as: The (i, j)th entry of the autocorrelated covariance matrix is defined as: $\rho^{|i-j|}$.

The value of `rho` must be such that $|\rho| < 1$ to ensure that the covariance matrix is positive definite.

Value

autocorrelated covariance matrix

cov_block_autocorrelation *Generates a $p \times p$ block-diagonal covariance matrix with autocorrelated blocks.*

Description

This function generates a $p \times p$ covariance matrix with autocorrelated blocks. The autocorrelation parameter is `rho`. There are `num_blocks` blocks each with size, `block_size`. The variance, `sigma2`, is constant for each feature and defaulted to 1.

Usage

```
cov_block_autocorrelation(num_blocks, block_size, rho, sigma2 = 1)
```

Arguments

num_blocks	the number of blocks in the covariance matrix
block_size	the size of each square block within the covariance matrix
rho	the autocorrelation parameter. Must be less than 1 in absolute value.
sigma2	the variance of each feature

Details

The autocorrelated covariance matrix is defined as:

$$\Sigma = \Sigma^{(\rho)} \oplus \Sigma^{(-\rho)} \oplus \ldots \oplus \Sigma^{(\rho)},$$

where $\oplus$ denotes the direct sum and the (i, j)th entry of $\Sigma^{(\rho)}$ is

$$\Sigma_{ij}^{(\rho)} = \{\rho^{|i-j|}\}.$$

The matrix $\Sigma^{(\rho)}$ is the autocorrelated block discussed above.

The value of rho must be such that $|\rho| < 1$ to ensure that the covariance matrix is positive definite.

The size of the resulting matrix is $p \times p$, where p = num_blocks * block_size.

Value

autocorrelated covariance matrix

---

cov_eigen *Computes the eigenvalue decomposition of the maximum likelihood estimators (MLE) of the covariance matrices for the given data matrix*

Description

For the classes given in the vector y, we compute the eigenvalue (spectral) decomposition of the class sample covariance matrices (MLEs) using the data matrix x.

Usage

```
cov_eigen(x, y, pool = FALSE, fast = FALSE, tol = 1e-06)
```

Arguments

x	data matrix with n observations and p feature vectors
y	class labels for observations (rows) in x
pool	logical. Should the sample covariance matrices be pooled?
fast	logical. Should the Fast SVD be used? See details.
tol	tolerance value below which the singular values of x are considered zero.

Details

If the `fast` argument is selected, we utilize the so-called Fast Singular Value Decomposition (SVD) to quickly compute the eigenvalue decomposition. To compute the Fast SVD, we use the `fast.svd` function, which employs a well-known trick for tall data (large `n`, small `p`) and wide data (large `p`, small `n`) to compute the SVD corresponding to the nonzero singular values. For more information about the Fast SVD, see `fast.svd`.

Value

a list containing the eigendecomposition for each class. If `pool = TRUE`, then a single list is returned.

Examples

```
cov_eigen(x = iris[, -5], y = iris[, 5])
cov_eigen(x = iris[, -5], y = iris[, 5], pool = TRUE)
cov_eigen(x = iris[, -5], y = iris[, 5], pool = TRUE, fast = TRUE)

# Generates a data set having fewer observations than features.
# We apply the Fast SVD to compute the eigendecomposition corresponding to the
# nonzero eigenvalues of the covariance matrices.
set.seed(42)
n <- 5
p <- 20
num_classes <- 3
x <- lapply(seq_len(num_classes), function(k) {
  replicate(p, rnorm(n, mean = k))
})
x <- do.call(rbind, x)
y <- gl(num_classes, n)
cov_eigen(x = x, y = y, fast = TRUE)
cov_eigen(x = x, y = y, pool = TRUE, fast = TRUE)
```

`cov_intraclass` *Generates a $p \times p$ intraclass covariance matrix*

Description

This function generates a $p \times p$ intraclass covariance matrix with correlation `rho`. The variance `sigma2` is constant for each feature and defaulted to 1.

Usage

```
cov_intraclass(p, rho, sigma2 = 1)
```

Arguments

`p`	the size of the covariance matrix
`rho`	the value of the off-diagonal elements
`sigma2`	the variance of each feature

Details

The intraclass covariance matrix is defined as:

$$sigma2 * (\rho * J_p + (1 - \rho) * I_p),$$

where J_p is the $p \times p$ matrix of ones and I_p is the $p \times p$ identity matrix.

By default, with `sigma2 = 1`, the diagonal elements of the intraclass covariance matrix are all 1, while the off-diagonal elements of the matrix are all `rho`.

The value of `rho` must be between $(1 - p)^(- 1)$ and 1, exclusively, to ensure that the covariance matrix is positive definite.

Value

intraclass covariance matrix

`cov_list` *Computes the covariance-matrix maximum likelihood estimators for each class and returns a list.*

Description

For a sample matrix, `x`, we compute the MLE for the covariance matrix for each class given in the vector, `y`.

Usage

```
cov_list(x, y)
```

Arguments

`x`	data matrix with `n` observations and `p` feature vectors
`y`	class labels for observations (rows) in `x`

Value

list of the sample covariance matrices of size $p \times p$ for each class given in `y`.

`cov_mle` *Computes the maximum likelihood estimator for the sample covariance matrix under the assumption of multivariate normality.*

Description

For a sample matrix, `x`, we compute the sample covariance matrix of the data as the maximum likelihood estimator (MLE) of the population covariance matrix.

Usage

```
cov_mle(x, diag = FALSE)
```

Arguments

`x`	data matrix with `n` observations and `p` feature vectors
`diag`	logical value. If TRUE, assumes the population covariance matrix is diagonal. By default, we assume that `diag` is `FALSE`.

Details

If the `diag` option is set to `TRUE`, then we assume the population covariance matrix is diagonal, and the MLE is computed under this assumption. In this case, we return a vector of length `p` instead.

Value

sample covariance matrix of size $p \times p$. If `diag` is `TRUE`, then a vector of length `p` is returned instead.

`cov_pool` *Computes the pooled maximum likelihood estimator (MLE) for the common covariance matrix*

Description

For the matrix `x`, we compute the MLE for the population covariance matrix under the assumption that the data are sampled from K multivariate normal populations having equal covariance matrices.

Usage

```
cov_pool(x, y)
```

Arguments

`x`	data matrix with `n` observations and `p` feature vectors
`y`	class labels for observations (rows) in `x`

Value

pooled sample covariance matrix of size $p \times p$

Examples

```
cov_pool(iris[, -5], iris$Species)
```

`cov_shrink_diag` *Computes a shrunken version of the maximum likelihood estimator for the sample covariance matrix under the assumption of multivariate normality.*

Description

For a sample matrix, `x`, we compute the sample covariance matrix as the maximum likelihood estimator (MLE) of the population covariance matrix and shrink it towards its diagonal.

Usage

```
cov_shrink_diag(x, gamma = 1)
```

Arguments

`x`	data matrix with `n` observations and `p` feature vectors
`gamma`	the shrinkage parameter. Must be between 0 and 1, inclusively. By default, the shrinkage parameter is 1, which simply yields the MLE.

Details

Let $\widehat{\Sigma}$ be the MLE of the covariance matrix Σ. Then, we shrink the MLE towards its diagonal by computing

$$\widehat{\Sigma}(\gamma) = \gamma\widehat{\Sigma} + (1-\gamma)\widehat{\Sigma} \circ I_p,$$

where $\circ$ denotes the Hadamard product and $\gamma \in [0, 1]$.

For $\gamma < 1$, the resulting shrunken covariance matrix estimator is positive definite, and for $\gamma = 1$, we simply have the MLE, which can potentially be positive semidefinite (singular).

The estimator given here is based on Section 18.3.1 of the Hastie et al. (2008) text.

Value

shrunken sample covariance matrix of size $p \times p$

References

Hastie, T., Tibshirani, R., and Friedman, J. (2008), "The Elements of Statistical Learning: Data Mining, Inference, and Prediction," 2nd edition. http://www-stat.stanford.edu/~tibs/ElemStatLearn/

`cv_partition` *Randomly partitions data for cross-validation.*

Description

For a vector of training labels, we return a list of cross-validation folds, where each fold has the indices of the observations to leave out in the fold. In terms of classification error rate estimation, one can think of a fold as a the observations to hold out as a test sample set. Either the `hold_out` size or the number of folds, `num_folds`, can be specified. The number of folds defaults to 10, but if the `hold_out` size is specified, then `num_folds` is ignored.

Usage

```
cv_partition(y, num_folds = 10, hold_out = NULL, seed = NULL)
```

Arguments

`y`	a vector of class labels
`num_folds`	the number of cross-validation folds. Ignored if `hold_out` is not `NULL`. See Details.
`hold_out`	the hold-out size for cross-validation. See Details.
`seed`	optional random number seed for splitting the data for cross-validation

Details

We partition the vector `y` based on its length, which we treat as the sample size, 'n'. If an object other than a vector is used in `y`, its length can yield unexpected results. For example, the output of `length(diag(3))` is 9.

Value

list the indices of the training and test observations for each fold.

Examples

```
# The following three calls to \code{cv_partition} yield the same partitions.
set.seed(42)
cv_partition(iris$Species)
cv_partition(iris$Species, num_folds = 10, seed = 42)
cv_partition(iris$Species, hold_out = 15, seed = 42)
```

diag_estimates *Computes estimates and ancillary information for diagonal classifiers*

Description

Computes the maximum likelihood estimators (MLEs) for each class under the assumption of multivariate normality for each class. Also, computes ancillary information necessary for classifier summary, such as sample size, the number of features, etc.

Usage

```
diag_estimates(x, y, prior = NULL, pool = FALSE, est_mean = c("mle",
  "tong"))
```

Arguments

x	matrix containing the training data. The rows are the sample observations, and the columns are the features.
y	vector of class labels for each training observation
prior	vector with prior probabilities for each class. If NULL (default), then equal probabilities are used. See details.
pool	logical value. If TRUE, calculates the pooled sample variances for each class.
est_mean	the estimator for the class means. By default, we use the maximum likelihood estimator (MLE). To improve the estimation, we provide the option to use a shrunken mean estimator proposed by Tong et al. (2012).

Details

This function computes the common estimates and ancillary information used in all of the diagonal classifiers in the sparsediscrim package.

The matrix of training observations are given in x. The rows of x contain the sample observations, and the columns contain the features for each training observation.

The vector of class labels given in y are coerced to a factor. The length of y should match the number of rows in x.

An error is thrown if a given class has less than 2 observations because the variance for each feature within a class cannot be estimated with less than 2 observations.

The vector, prior, contains the *a priori* class membership for each class. If prior is NULL (default), the class membership probabilities are estimated as the sample proportion of observations belonging to each class. Otherwise, prior should be a vector with the same length as the number of classes in y. The prior probabilties should be nonnegative and sum to one.

Value

named list with estimators for each class and necessary ancillary information

References

Tong, T., Chen, L., and Zhao, H. (2012), "Improved Mean Estimation and Its Application to Diagonal Discriminant Analysis," Bioinformatics, 28, 4, 531-537. http://bioinformatics.oxfordjournals.org/content/28/4/531.long

---

`generate_blockdiag` *Generates data from* `K` *multivariate normal data populations, where each population (class) has a covariance matrix consisting of block-diagonal autocorrelation matrices.*

Description

This function generates `K` multivariate normal data sets, where each class is generated with a constant mean vector and a covariance matrix consisting of block-diagonal autocorrelation matrices. The data are returned as a single matrix `x` along with a vector of class labels `y` that indicates class membership.

Usage

```
generate_blockdiag(n, block_size, num_blocks, rho, mu, sigma2 = rep(1, K))
```

Arguments

`n`	vector of the sample sizes of each class. The length of `n` determines the number of classes `K`.
`block_size`	the dimensions of the square block matrix. See details.
`num_blocks`	the number of block matrices. See details.
`rho`	vector of the values of the autocorrelation parameter for each class covariance matrix. Must equal the length of `n` (i.e., equal to `K`).
`mu`	vector containing the mean for each class. Must equal the length of `n` (i.e., equal to `K`).
`sigma2`	vector of the variance coefficients for each class covariance matrix. Must equal the length of `n` (i.e., equal to `K`).

Details

For simplicity, we assume that a class mean vector is constant for each feature. That is, we assume that the mean vector of the kth class is $c_k * j_p$, where j_p is a $p \times 1$ vector of ones and c_k is a real scalar.

The kth class covariance matrix is defined as

$$\Sigma_k = \Sigma^{(\rho)} \oplus \Sigma^{(-\rho)} \oplus \ldots \oplus \Sigma^{(\rho)},$$

where $\oplus$ denotes the direct sum and the (i, j)th entry of $\Sigma^{(\rho)}$ is

$$\Sigma_{ij}^{(\rho)} = \{\rho^{|i-j|}\}.$$

The matrix $\Sigma^{(\rho)}$ is referred to as a block. Its dimensions are provided in the `block_size` argument, and the number of blocks are specified in the `num_blocks` argument.

Each matrix Σ_k is generated by the `cov_block_autocorrelation` function.

The number of classes `K` is determined with lazy evaluation as the length of `n`.

Value

named list with elements:

- `x`: matrix of observations with `n` rows and `p` columns
- `y`: vector of class labels that indicates class membership for each observation (row) in `x`.

Examples

```
# Generates data from K = 3 classes.
data <- generate_blockdiag(n = c(15, 15, 15), block_size = 3, num_blocks = 3,
rho = seq(.1, .9, length = 3), mu = c(0, 3, -2))
data$x
data$y

# Generates data from K = 4 classes. Notice that we use specify a variance.
data <- generate_blockdiag(n = c(15, 15, 15, 20), block_size = 3, num_blocks = 3,
rho = seq(.1, .9, length = 4), mu = c(0, 3, -2, 6))
data$x
data$y
```

---

generate_intraclass *Generates data from `K` multivariate normal data populations, where each population (class) has an intraclass covariance matrix.*

Description

This function generates `K` multivariate normal data sets, where each class is generated with a constant mean vector and an intraclass covariance matrix. The data are returned as a single matrix `x` along with a vector of class labels `y` that indicates class membership.

Usage

```
generate_intraclass(n, p, rho, mu, sigma2 = rep(1, K))
```

Arguments

`n`	vector of the sample sizes of each class. The length of `n` determines the number of classes `K`.
`p`	the number of features (variables) in the data
`rho`	vector of the values of the off-diagonal elements for each intraclass covariance matrix. Must equal the length of `n`.

mu	vector containing the mean for each class. Must equal the length of n (i.e., equal to K).
sigma2	vector of variances for each class. Must equal the length of n. Default is 1 for each class.

Details

For simplicity, we assume that a class mean vector is constant for each feature. That is, we assume that the mean vector of the kth class is $c_k * j_p$, where j_p is a $p \times 1$ vector of ones and c_k is a real scalar.

The intraclass covariance matrix for the kth class is defined as:

$$\sigma_k^2 * (\rho_k * J_p + (1 - \rho_k) * I_p),$$

where J_p is the $p \times p$ matrix of ones and I_p is the $p \times p$ identity matrix.

By default, with $sigma_k^2 = 1$, the diagonal elements of the intraclass covariance matrix are all 1, while the off-diagonal elements of the matrix are all rho.

The values of rho must be between $(1 - p)^(- 1)$ and 1, exclusively, to ensure that the covariance matrix is positive definite.

The number of classes K is determined with lazy evaluation as the length of n.

Value

named list with elements:

- x: matrix of observations with n rows and p columns
- y: vector of class labels that indicates class membership for each observation (row) in x.

Examples

```
# Generates data from K = 3 classes.
data <- generate_intraclass(n = 3:5, p = 5, rho = seq(.1, .9, length = 3),
                            mu = c(0, 3, -2))
data$x
data$y

# Generates data from K = 4 classes. Notice that we use specify a variance.
data <- generate_intraclass(n = 3:6, p = 4, rho = seq(0, .9, length = 4),
                            mu = c(0, 3, -2, 6), sigma2 = 1:4)
data$x
data$y
```

h — *Bias correction function from Pang et al. (2009).*

Description

This function computes the function $h_{nu,p}(t)$ on page 1023 of Pang et al. (2009).

Usage

```
h(nu, p, t = -1)
```

Arguments

nu	a specified constant (nu = N - K)
p	the feature space dimension.
t	a constant specified by the user that indicates the exponent to use with the variance estimator. By default, t = -1 as in Pang et al. See the paper for more details.

Value

the bias correction value

References

Pang, H., Tong, T., & Zhao, H. (2009). "Shrinkage-based Diagonal Discriminant Analysis and Its Applications in High-Dimensional Data," Biometrics, 65, 4, 1021-1029. http://onlinelibrary.wiley.com/doi/10.1111/j.1541-0420.2009.01200.x/abstract

hdrda_cv — *Helper function to optimize the HDRDA classifier via cross-validation*

Description

For a given data set, we apply cross-validation (cv) to select the optimal HDRDA tuning parameters.

Usage

```
hdrda_cv(x, y, num_folds = 10, num_lambda = 21, num_gamma = 7,
  shrinkage_type = c("ridge", "convex"), ...)
```

Arguments

x	matrix containing the training data. The rows are the sample observations, and the columns are the features.
y	vector of class labels for each training observation
num_folds	the number of cross-validation folds.
num_lambda	The number of values of lambda to consider
num_gamma	The number of values of gamma to consider
shrinkage_type	the type of covariance-matrix shrinkage to apply. By default, a ridge-like shrinkage is applied. If convex is given, then shrinkage similar to Friedman (1989) is applied. See Ramey et al. (2014) for details.
...	Additional arguments passed to hdrda.

Details

The number of cross-validation folds is given in num_folds.

Value

list containing the HDRDA model that minimizes cross-validation as well as a data.frame that summarizes the cross-validation results.

---

no_intercept *Removes the intercept term from a formula if it is included*

Description

Often, we prefer not to have an intercept term in a model, but user-specified formulas might have included the intercept term. In this case, we wish to update the formula but without the intercept term. This is especially true in numerous classification models, where errors and doom can occur if an intercept is included in the model.

Usage

```
no_intercept(formula, data)
```

Arguments

formula	a model formula to remove its intercept term
data	data frame

Value

formula with no intercept term

Examples

```
iris_formula <- formula(Species ~ .)
sparsediscrim:::no_intercept(iris_formula, data = iris)
```

quadform *Quadratic form of a matrix and a vector*

Description

We compute the quadratic form of a vector and a matrix in an efficient manner. Let x be a real vector of length p, and let A be a p x p real matrix. Then, we compute the quadratic form $q = x'Ax$.

Usage

```
quadform(A, x)
```

Arguments

A	matrix of dimension p x p
x	vector of length p

Details

A naive way to compute the quadratic form is to explicitly write `t(x) %*% A %*% x`, but for large p, this operation is inefficient. We provide a more efficient method below.

Note that we have adapted the code from: http://tolstoy.newcastle.edu.au/R/help/05/11/14989.html

Value

scalar value

quadform_inv *Quadratic Form of the inverse of a matrix and a vector*

Description

We compute the quadratic form of a vector and the inverse of a matrix in an efficient manner. Let x be a real vector of length p, and let A be a p x p nonsingular matrix. Then, we compute the quadratic form $q = x'A^{-1}x$.

Usage

```
quadform_inv(A, x)
```

Arguments

A	matrix that is p x p and nonsingular
x	vector of length p

Details

A naive way to compute the quadratic form is to explicitly write `t(x) %*% solve(A) %*% x`, but for large `p`, this operation is inefficient. We provide a more efficient method below.

Note that we have adapted the code from: http://tolstoy.newcastle.edu.au/R/help/05/11/14989.html

Value

scalar value

---

rda_cov *Calculates the RDA covariance-matrix estimators for each class*

Description

For the classes given in the vector `y`, this function calculates the class covariance-matrix estimators employed in the HDRDA classifier, implemented in `hdrda`.

Usage

```
rda_cov(x, y, lambda = 1)
```

Arguments

`x`	matrix containing the training data. The rows are the sample observations, and the columns are the features.
`y`	vector of class labels for each training observation
`lambda`	the RDA pooling parameter. Must be between 0 and 1, inclusively.

Value

list containing the RDA covariance-matrix estimators for each class given in `y`

References

Ramey, J. A., Stein, C. K., and Young, D. M. (2013), "High-Dimensional Regularized Discriminant Analysis."

`rda_weights` *Computes the observation weights for each class for the HDRDA classifier*

Description

This function calculates the weight for each observation in the data matrix `x` in order to calculate the covariance matrices employed in the HDRDA classifier, implemented in `hdrda`.

Usage

```
rda_weights(x, y, lambda = 1)
```

Arguments

`x`	matrix containing the training data. The rows are the sample observations, and the columns are the features.
`y`	vector of class labels for each training observation
`lambda`	the RDA pooling parameter. Must be between 0 and 1, inclusively.

Value

list containing the observations for each class given in `y`

References

Ramey, J. A., Stein, C. K., and Young, D. M. (2013), "High-Dimensional Regularized Discriminant Analysis."

`regdiscrim_estimates` *Computes estimates and ancillary information for regularized discriminant classifiers*

Description

Computes the maximum likelihood estimators (MLEs) for each class under the assumption of multivariate normality for each class. Also, computes ancillary information necessary for classifier summary, such as sample size, the number of features, etc.

Usage

```
regdiscrim_estimates(x, y, cov = TRUE, prior = NULL)
```

Arguments

x	matrix containing the training data. The rows are the sample observations, and the columns are the features.
y	vector of class labels for each training observation
cov	logical. Should the sample covariance matrices be computed? (Default: yes)
prior	vector with prior probabilities for each class. If NULL (default), then the sample proportions are used. See details.

Details

This function computes the common estimates and ancillary information used in all of the regularized discriminant classifiers in the `regdiscrim` package.

The matrix of training observations are given in `x`. The rows of `x` contain the sample observations, and the columns contain the features for each training observation.

The vector of class labels given in `y` are coerced to a `factor`. The length of `y` should match the number of rows in `x`.

An error is thrown if a given class has less than 2 observations because the variance for each feature within a class cannot be estimated with less than 2 observations.

The vector, `prior`, contains the *a priori* class membership for each class. If `prior` is NULL (default), the class membership probabilities are estimated as the sample proportion of observations belonging to each class. Otherwise, `prior` should be a vector with the same length as the number of classes in `y`. The `prior` probabilties should be nonnegative and sum to one.

Value

named list with estimators for each class and necessary ancillary information

---

`risk_stein` *Stein Risk function from Pang et al. (2009).*

Description

This function finds the value for $alpha \in [0, 1]$ that empirically minimizes the average risk under a Stein loss function, which is given on page 1023 of Pang et al. (2009).

Usage

```
risk_stein(N, K, var_feature, num_alphas = 101, t = -1)
```

Arguments

N	the sample size.
K	the number of classes.
var_feature	a vector of the sample variances for each dimension.
num_alphas	The number of values used to find the optimal amount of shrinkage.
t	a constant specified by the user that indicates the exponent to use with the variance estimator. By default, t = -1 as in Pang et al. See the paper for more details.

Value

list with

- alpha: the alpha that minimizes the average risk under a Stein loss function. If the minimum is not unique, we randomly select an alpha from the minimizers.
- risk: the minimum average risk attained.

References

Pang, H., Tong, T., & Zhao, H. (2009). "Shrinkage-based Diagonal Discriminant Analysis and Its Applications in High-Dimensional Data," Biometrics, 65, 4, 1021-1029. http://onlinelibrary.wiley.com/doi/10.1111/j.1541-0420.2009.01200.x/abstract

---

solve_chol — *Computes the inverse of a symmetric, positive-definite matrix using the Cholesky decomposition*

Description

This often faster than solve for larger matrices. See, for example: http://blog.phytools.org/2012/12/faster-inversion-of-square-symmetric.html and http://stats.stackexchange.com/questions/14951/efficient-calculation-of-matrix-inverse-in-r.

Usage

```
solve_chol(x)
```

Arguments

x	symmetric, positive-definite matrix

Value

the inverse of x

tong_mean_shrinkage *Tong et al. (2012)'s Lindley-type Shrunken Mean Estimator*

Description

An implementation of the Lindley-type shrunken mean estimator utilized in shrinkage-mean-based diagonal linear discriminant analysis (SmDLDA).

Usage

```
tong_mean_shrinkage(x, r_opt = NULL)
```

Arguments

x	a matrix with n rows and p columns.
r_opt	the shrinkage coefficient. If NULL (default), we calculate the shrinkage coefficient with the formula given just above Equation 5 on page 533 and denoted by $\hat{r}_{opt}$. We allow the user to specify an alternative value to investigate better approximations.

Value

vector of length p with the shrunken mean estimator

References

Tong, T., Chen, L., and Zhao, H. (2012), "Improved Mean Estimation and Its Application to Diagonal Discriminant Analysis," Bioinformatics, 28, 4, 531-537. http://bioinformatics.oxfordjournals.org/content/28/4/531.long

update_hdrda *Helper function to update tuning parameters for the HDRDA classifier*

Description

This function updates some of the quantities in the HDRDA classifier based on updated values of lambda and gamma. The update can greatly expedite cross-validation to examine a large grid of values for lambda and gamma.

Usage

```
update_hdrda(obj, lambda = 1, gamma = 0)
```

Arguments

obj	a hdrda object
lambda	a numeric value between 0 and 1, inclusively
gamma	a numeric value (nonnegative)

Details

When $p < N$, the HDRDA covariance-matrix estimator is singular when $(lambda, gamma) = (0, 0)$. In this case, we set $gamma$ to 0.01 to shrink the estimators slightly to ensure positive-definiteness.

Value

a hdrda object with updated estimates

---

var_shrinkage *Shrinkage-based estimator of variances for each feature from Pang et al. (2009).*

Description

This function computes the shrinkage-based estimator of variance of each feature (variable) from Pang et al. (2009) for the SDLDA classifier.

Usage

```
var_shrinkage(N, K, var_feature, num_alphas = 101, t = -1)
```

Arguments

N	the sample size.
K	the number of classes.
var_feature	a vector of the sample variances for each feature.
num_alphas	The number of values used to find the optimal amount of shrinkage.
t	a constant specified by the user that indicates the exponent to use with the variance estimator. By default, t = -1 as in Pang et al. See the paper for more details.

Value

a vector of the shrunken variances for each feature.

References

Pang, H., Tong, T., & Zhao, H. (2009). "Shrinkage-based Diagonal Discriminant Analysis and Its Applications in High-Dimensional Data," Biometrics, 65, 4, 1021-1029. http://onlinelibrary.wiley.com/doi/10.1111/j.1541-0420.2009.01200.x/abstract

Index