Package ‘sparsediscrim’

October 14, 2022

Title Sparse and Regularized Discriminant Analysis
Version 0.3.0
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
from Ramey et al. (2017) <arXiv:1602.01182>. Other classifiers include
those from Dudoit et al. (2002) <doi:10.1198/016214502753479248>, Pang et
Imports bdsmatrix, corpcor, dplyr, ggplot2, mvtnorm, rlang
Suggests testthat, MASS, covr, modeldata, spelling
License MIT + file LICENSE
URL https://github.com/topepo/sparsediscrim,
      https://topepo.github.io/sparsediscrim/
RoxygenNote 7.1.1.9001
Depends R (>= 2.10)
Encoding UTF-8
Language en-US
NeedsCompilation no
Author Max Kuhn [aut, cre] (<https://orcid.org/0000-0003-2402-136X>),
       John Ramey [aut] (original author)
Maintainer Max Kuhn <mxkuhn@gmail.com>
Repository CRAN
Date/Publication 2021-07-01 07:50:02 UTC

R topics documented:

center_data ................................................................. 3
cov_autocorrelation .................................................... 3
cov_block_autocorrelation ........................................... 4
R topics documented:

cov_eigen .................................................. 5
cov_intraclass ............................................ 6
cov_list .................................................... 7
cov_mle ...................................................... 7
cov_pool ...................................................... 8
cov_shrink_diag ........................................... 8
cv_partition ................................................ 9
diag_estimates ............................................ 10
dmvnorm_diag ............................................ 11
generate_blockdiag ..................................... 12
generate_intraclass ..................................... 13
h ............................................................. 14
lda_diag .................................................... 15
lda_eigen ................................................... 17
lda_emp_bayes ........................................... 19
lda_emp_bayes_eigen .................................... 21
lda_pseudo ................................................ 23
lda_schafer ............................................... 24
lda_shrink_cov .......................................... 26
lda_shrink_mean ......................................... 28
lda_thomaz ............................................... 30
log_determinant ......................................... 32
no_intercept ............................................. 33
plot.rda_high_dim_cv .................................. 33
posterior_probs .......................................... 34
qda_diag .................................................... 34
qda_shrink_cov .......................................... 36
qda_shrink_mean ......................................... 38
quadform ................................................... 40
quadform_inv ............................................. 41
rda_cov ..................................................... 42
rda_high_dim ............................................. 42
rda_high_dim_cv ......................................... 45
rda_weights ............................................... 46
regdiscrim_estimates .................................... 46
risk_stein .................................................. 47
solve_chol .................................................. 48
tong_mean_shrinkage .................................... 49
two_class_sim_data ...................................... 49
update_rda_high_dim ................................... 50
var_shrinkage ............................................. 50

Index ..................................................... 52
**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 or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.
- **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 $\text{num_blocks}$ blocks each with size, $\text{block_size}$. The variance, $\sigma^2$, is constant for each feature and defaulted to 1.

Usage

```r
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 $p = \text{num_blocks} \times \text{block_size}$.

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 corpcor::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 corpcor::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) {
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 $\sigma^2$ is constant for each feature and defaulted to 1.

Usage

cov_intraclass(p, rho, sigma2 = 1)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>the size of the covariance matrix</td>
</tr>
<tr>
<td>rho</td>
<td>the value of the off-diagonal elements</td>
</tr>
<tr>
<td>sigma2</td>
<td>the variance of each feature</td>
</tr>
</tbody>
</table>

Details

The intraclass covariance matrix is defined as:

$$
\sigma^2 \times (\rho \times J_p + (1 - \rho) \times I_p),
$$

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

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

The value of $\rho$ must be between $1/(1-p)$ 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**

\[
\text{cov_list}(x, y)
\]

**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**

\[
\text{cov_mle}(x, \text{diag} = \text{FALSE})
\]

**Arguments**

- \( x \) data matrix with \( n \) observations and \( p \) feature vectors
- \( \text{diag} \) logical value. If \( \text{TRUE} \), assumes the population covariance matrix is diagonal. By default, we assume that \( \text{diag} \) is \( \text{FALSE} \).

**Details**

If the \( \text{diag} \) option is set to \( \text{TRUE} \), 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.
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)

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)
cv_partition

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 \( \hat{\Sigma} \) be the MLE of the covariance matrix \( \Sigma \). Then, we shrink the MLE towards its diagonal by computing

\[
\hat{\Sigma}(\gamma) = \gamma \hat{\Sigma} + (1 - \gamma) \hat{\Sigma} \circ I_p,
\]

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


---

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 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 
\[
\text{length(diag(3))}
\]
is 9.

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

Examples

```r
# The following three calls to `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

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

Arguments

- **x**: Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.
- **y**: Vector of class labels for each training observation. Only complete data are retained.
- **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. If other data have zero variances, these will be removed with a warning.

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

Value

named list with estimators for each class and necessary ancillary information

References


---

dmvnorm_diag

Computes multivariate normal density with a diagonal covariance matrix

Description

Alternative to \texttt{mvtnorm::dmvnorm}

Usage

\texttt{dmvnorm.diag(x, mean, sigma)}

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>matrix</td>
</tr>
<tr>
<td>( \text{mean} )</td>
<td>vector of means</td>
</tr>
<tr>
<td>( \text{sigma} )</td>
<td>vector containing diagonal covariance matrix</td>
</tr>
</tbody>
</table>

Value

multivariate normal density
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

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

Arguments

- `n` vector of the sample sizes of each class. The length of n determines the number of classes K.
- `mu` matrix containing the mean vectors for each class. Expected to have p rows and K columns.
- `num_blocks` the number of block matrices. See details.
- `block_size` the dimensions of the square block matrix. 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).
- `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 x 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

\[ \Sigma^{(\rho)}_{ij} = \rho^{|i-j|}. \]

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 \( \Sigma_k \) is generated by the `cov_block_autocorrelation()` function.

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

The number of features p is computed as `block_size * num_blocks`. 
**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**

```r
# Generates data from K = 3 classes.
means <- matrix(rep(1:3, each=9), ncol=3)
data <- generate_blockdiag(n = c(15, 15, 15), block_size = 3, num_blocks = 3,
                          rho = seq(.1, .9, length = 3), mu = means)
data$x
data$y

# Generates data from K = 4 classes. Notice that we use specify a variance.
means <- matrix(rep(1:4, each=9), ncol=4)
data <- generate_blockdiag(n = c(15, 15, 15, 20), block_size = 3, num_blocks = 3,
rho = seq(.1, .9, length = 4), mu = means)
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**

```r
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 $k$th class is $c_k \ast j_p$, where $j_p$ is a $p \times 1$ vector of ones and $c_k$ is a real scalar.

The intraclass covariance matrix for the $k$th class is defined as:

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

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 $1/(1 - p)$ 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

```r
# 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**


---

**lda_diag**

*Diagonal Linear Discriminant Analysis (DLDA)*

**Description**

Given a set of training data, this function builds the Diagonal Linear Discriminant Analysis (DLDA) classifier, which is often attributed to Dudoit et al. (2002). The DLDA classifier belongs to the family of Naive Bayes classifiers, where the distributions of each class are assumed to be multivariate normal and to share a common covariance matrix.

The DLDA classifier is a modification to LDA, where the off-diagonal elements of the pooled sample covariance matrix are set to zero.

**Usage**

```r
lda_diag(x, ...)
```

---

## Default S3 method:
```r
lda_diag(x, y, prior = NULL, ...)
```

## S3 method for class `'formula'`
```r
lda_diag(formula, data, prior = NULL, ...)
```

## S3 method for class `'lda_diag'`
```r
predict(object, newdata, type = c("class", "prob", "score"), ...)
```
Arguments

- **x**: Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.
- ... additional arguments (not currently used).
- **y**: Vector of class labels for each training observation. Only complete data are retained.
- **prior**: Vector with prior probabilities for each class. If NULL (default), then equal probabilities are used. See details.
- **formula**: A formula of the form `groups ~ x1 + x2 + ...`. That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.
- **data**: data frame from which variables specified in formula are preferentially to be taken.
- **object**: Fitted model object
- **newdata**: Matrix or data frame of observations to predict. Each row corresponds to a new observation.
- **type**: Prediction type: either "class", "prob", or "score".

Details

The DLDA classifier is a modification to the well-known LDA classifier, where the off-diagonal elements of the pooled sample covariance matrix are assumed to be zero – the features are assumed to be uncorrelated. Under multivariate normality, the assumption uncorrelated features is equivalent to the assumption of independent features. The feature-independence assumption is a notable attribute of the Naive Bayes classifier family. The benefit of these classifiers is that they are fast and have much fewer parameters to estimate, especially when the number of features is quite large.

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 probabilities should be nonnegative and sum to one.

Value

The model fitting function returns the fitted classifier. The `predict()` method returns either a vector (type = "class") or a data frame (all other type values).

References

Examples

```r
library(modeldata)
data(penguins)
pred_rows <- seq(1, 344, by = 20)
penguins <- penguins[, c("species", "body_mass_g", "flipper_length_mm")]
dlda_out <- lda_diag(species ~ ., data = penguins[-pred_rows, ])
predicted <- predict(dlda_out, penguins[pred_rows, -1], type = "class")

dlda_out2 <- lda_diag(x = penguins[-pred_rows, -1], y = penguins$species[-pred_rows])
predicted2 <- predict(dlda_out2, penguins[pred_rows, -1], type = "class")
all.equal(predicted, predicted2)
```

### Description

Given a set of training data, this function builds the MDMP classifier from Srivistava and Kubokawa (2007). The MDMP classifier is an adaptation of the linear discriminant analysis (LDA) classifier that is designed for small-sample, high-dimensional data. Srivastava and Kubokawa (2007) have proposed a modification of the standard maximum likelihood estimator of the pooled covariance matrix, where only the largest 95% of the eigenvalues and their corresponding eigenvectors are kept. The value of 95% is the default and can be changed via the `eigen_pct` argument.

The MDMP classifier from Srivastava and Kubokawa (2007) is an adaptation of the linear discriminant analysis (LDA) classifier that is designed for small-sample, high-dimensional data. Srivastava and Kubokawa (2007) have proposed a modification of the standard maximum likelihood estimator of the pooled covariance matrix, where only the largest 95% of the eigenvalues and their corresponding eigenvectors are kept.

### Usage

```r
lda_eigen(x, ...)

## Default S3 method:
lda_eigen(x, y, prior = NULL, eigen_pct = 0.95, ...)

## S3 method for class 'formula'
lda_eigen(formula, data, prior = NULL, ...)

## S3 method for class 'lda_eigen'
predict(object, newdata, type = c("class", "prob", "score"), ...)
```

### Arguments

- `x`: Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.
... additional arguments (not currently used).

y Vector of class labels for each training observation. Only complete data are retained.
prior Vector with prior probabilities for each class. If NULL (default), then equal probabilities are used. See details.
eigen_pct the percentage of eigenvalues kept
formula A formula of the form groups ~ x1 + x2 + ... That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.
data data frame from which variables specified in formula are preferentially to be taken.
object Fitted model object
newdata Matrix or data frame of observations to predict. Each row corresponds to a new observation.
type Prediction type: either "class", "prob", or "score".

Details
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 probabilities should be nonnegative and sum to one.

Value
lda_eigen object that contains the trained MDMP classifier

References

Examples
library(modeldata)
data(penguins)
pred_rows <- seq(1, 344, by = 20)
penguins <- penguins[, c("species", "body_mass_g", "flipper_length_mm")]
mdmp_out <- lda_eigen(species ~ ., data = penguins[-pred_rows, ])
predicted <- predict(mdmp_out, penguins[pred_rows, -1], type = "class")

mdmp_out2 <- lda_eigen(x = penguins[-pred_rows, -1], y = penguins$species[-pred_rows])
The Minimum Distance Empirical Bayesian Estimator (MDEB) classifier

Description

Given a set of training data, this function builds the MDEB classifier from Srivastava and Kubokawa (2007). The MDEB classifier is an adaptation of the linear discriminant analysis (LDA) classifier that is designed for small-sample, high-dimensional data. Rather than using the standard maximum likelihood estimator of the pooled covariance matrix, Srivastava and Kubokawa (2007) have proposed an Empirical Bayes estimator where the eigenvalues of the pooled sample covariance matrix are shrunken towards the identity matrix: the shrinkage constant has a closed form and is quick to calculate.

The MDEB classifier from Srivastava and Kubokawa (2007) is an adaptation of the linear discriminant analysis (LDA) classifier that is designed for small-sample, high-dimensional data. Rather than using the standard maximum likelihood estimator of the pooled covariance matrix, Srivastava and Kubokawa (2007) have proposed an Empirical Bayes estimator where the eigenvalues of the pooled sample covariance matrix are shrunken towards the identity matrix: the shrinkage constant has a closed form and is quick to calculate.

Usage

lda_emp_bayes(x, ...)

## Default S3 method:
lda_emp_bayes(x, y, prior = NULL, ...)

## S3 method for class 'formula'
lda_emp_bayes(formula, data, prior = NULL, ...)

## S3 method for class 'lda_emp_bayes'
predict(object, newdata, type = c("class", "prob", "score"), ...)

Arguments

x Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.

... additional arguments (not currently used).

y Vector of class labels for each training observation. Only complete data are retained.

prior Vector with prior probabilities for each class. If NULL (default), then equal probabilities are used. See details.
formula A formula of the form \( \text{groups} \sim x_1 + x_2 + \ldots \). That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.

data data frame from which variables specified in formula are preferentially to be taken.

object Fitted model object

newdata Matrix or data frame of observations to predict. Each row corresponds to a new observation.

type Prediction type: either "class", "prob", or "score".

Details

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 probabilities should be nonnegative and sum to one.

Value

\bold{lda_emp_bayes} object that contains the trained MDEB classifier

References


Examples

```r
library(modeldata)
data(penguins)
pred_rows <- seq(1, 344, by = 20)
penguins <- penguins[, c("species", "body_mass_g", "flipper_length_mm")]
mdub_out <- lda_emp_bayes(species ~ ., data = penguins[-pred_rows, ])
predicted <- predict(mdeb_out, penguins[pred_rows, -1], type = "class")

mdeb_out2 <- lda_emp_bayes(x = penguins[-pred_rows, -1], y = penguins$species[-pred_rows])
predicted2 <- predict(mdeb_out2, penguins[pred_rows, -1], type = "class")
all.equal(predicted, predicted2)
```
The Minimum Distance Rule using Modified Empirical Bayes (MDMEB) classifier

Description

Given a set of training data, this function builds the MDMEB classifier from Srivistava and Kubokawa (2007). The MDMEB classifier is an adaptation of the linear discriminant analysis (LDA) classifier that is designed for small-sample, high-dimensional data. Srivastava and Kubokawa (2007) have proposed a modification of the standard maximum likelihood estimator of the pooled covariance matrix, where only the largest 95% of the eigenvalues and their corresponding eigenvectors are kept. The resulting covariance matrix is then shrunken towards a scaled identity matrix. The value of 95% is the default and can be changed via the `eigen_pct` argument.

The MDMEB classifier is an adaptation of the linear discriminant analysis (LDA) classifier that is designed for small-sample, high-dimensional data. Srivastava and Kubokawa (2007) have proposed a modification of the standard maximum likelihood estimator of the pooled covariance matrix, where only the largest 95% of the eigenvalues and their corresponding eigenvectors are kept. The resulting covariance matrix is then shrunken towards a scaled identity matrix.

Usage

```r
lda_emp_bayes_eigen(x, ...)  
# Default S3 method:
lda_emp_bayes_eigen(x, y, prior = NULL, eigen_pct = 0.95, ...)  
# S3 method for class 'formula'
lda_emp_bayes_eigen(formula, data, prior = NULL, ...)  
# S3 method for class 'lda_emp_bayes_eigen'
predict(object, newdata, type = c("class", "prob", "score"), ...)
```

Arguments

- **x**: Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.
- **...**: additional arguments (not currently used).
- **y**: Vector of class labels for each training observation. Only complete data are retained.
- **prior**: Vector with prior probabilities for each class. If NULL (default), then equal probabilities are used. See details.
- **eigen_pct**: the percentage of eigenvalues kept
- **formula**: A formula of the form `groups ~ x1 + x2 + ...`. That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.
**lda_emp_bayes_eigen**

- **data**: data frame from which variables specified in formula are preferentially to be taken.
- **object**: Fitted model object
- **newdata**: Matrix or data frame of observations to predict. Each row corresponds to a new observation.
- **type**: Prediction type: either "class", "prob", or "score".

**Details**

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` probabilities should be nonnegative and sum to one.

**Value**

`lda_emp_bayes_eigen` object that contains the trained MDMEB classifier

**References**


**Examples**

```r
library(modeldata)
data(penguins)
pred_rows <- seq(1, 344, by = 20)
penguins <- penguins[, c("species", "body_mass_g", "flipper_length_mm")]
mdmeb_out <- lda_emp_bayes_eigen(species ~ ., data = penguins[-pred_rows, ])
predicted <- predict(mdmeb_out, penguins[pred_rows, -1], type = "class")

mdmeb_out2 <- lda_emp_bayes_eigen(x = penguins[-pred_rows, -1], y = penguins$species[-pred_rows])
predicted2 <- predict(mdmeb_out2, penguins[pred_rows, -1], type = "class")
all.equal(predicted, predicted2)
```
**Description**

Given a set of training data, this function builds the Linear Discriminant Analysis (LDA) classifier, where the distributions of each class are assumed to be multivariate normal and share a common covariance matrix. When the pooled sample covariance matrix is singular, the linear discriminant function is incalculable. A common method to overcome this issue is to replace the inverse of the pooled sample covariance matrix with the Moore-Penrose pseudo-inverse, which is unique and always exists. Note that when the pooled sample covariance matrix is nonsingular, it is equal to the pseudo-inverse.

The Linear Discriminant Analysis (LDA) classifier involves the assumption that the distributions of each class are assumed to be multivariate normal and share a common covariance matrix. When the pooled sample covariance matrix is singular, the linear discriminant function is incalculable. A common method to overcome this issue is to replace the inverse of the pooled sample covariance matrix with the Moore-Penrose pseudo-inverse, which is unique and always exists. Note that when the pooled sample covariance matrix is nonsingular, it is equal to the pseudo-inverse.

**Usage**

```r
lda_pseudo(x, ...)
```

## Default S3 method:
```r
lda_pseudo(x, y, prior = NULL, tol = 1e-08, ...)
```

## S3 method for class 'formula'
```r
lda_pseudo(formula, data, prior = NULL, tol = 1e-08, ...)
```

## S3 method for class 'lda_pseudo'
```r
predict(object, newdata, type = c("class", "prob", "score"), ...)
```

**Arguments**

- `x`  
  Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.

- `...`  
  Additional arguments (not currently used).

- `y`  
  Vector of class labels for each training observation. Only complete data are retained.

- `prior`  
  Vector with prior probabilities for each class. If NULL (default), then equal probabilities are used. See details.

- `tol`  
  Tolerance value below which eigenvalues are considered numerically equal to 0.

- `formula`  
  A formula of the form `groups ~ x1 + x2 + ...`. That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.
**data**

data frame from which variables specified in formula are preferentially to be taken.

**object**

Fitted model object

**newdata**

Matrix or data frame of observations to predict. Each row corresponds to a new observation.

**type**

Prediction type: either "class", "prob", or "score".

---

**Details**

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 probabilities should be nonnegative and sum to one.

---

**Value**

lda_pseudo object that contains the trained lda_pseudo classifier

---

**Examples**

```r
library(modeldata)
data(penguins)
pred_rows <- seq(1, 344, by = 20)
penguins <- penguins[, c("species", "body_mass_g", "flipper_length_mm")]
lda_pseudo_out <- lda_pseudo(species ~ ., data = penguins[-pred_rows, ])
predicted <- predict(lda_pseudo_out, penguins[pred_rows, -1], type = "class")

lda_pseudo_out2 <- lda_pseudo(x = penguins[-pred_rows, -1], y = penguins$species[-pred_rows])
predicted2 <- predict(lda_pseudo_out2, penguins[pred_rows, -1], type = "class")
all.equal(predicted, predicted2)
```

---

**lda_schafer**

Linear Discriminant Analysis using the Schafer-Strimmer Covariance Matrix Estimator
**Description**

Given a set of training data, this function builds the Linear Discriminant Analysis (LDA) classifier, where the distributions of each class are assumed to be multivariate normal and share a common covariance matrix. When the pooled sample covariance matrix is singular, the linear discriminant function is incalculable. This function replaces the inverse of pooled sample covariance matrix with an estimator proposed by Schafer and Strimmer (2005). The estimator is calculated via `corpcor::invcov.shrink()`.

The Linear Discriminant Analysis (LDA) classifier involves the assumption that the distributions of each class are assumed to be multivariate normal and share a common covariance matrix. When the pooled sample covariance matrix is singular, the linear discriminant function is incalculable. Here, the inverse of the pooled sample covariance matrix is replaced with an estimator from Schafer and Strimmer (2005).

**Usage**

```r
lda_schafer(x, ...)  
## Default S3 method:  
lda_schafer(x, y, prior = NULL, ...)  
## S3 method for class 'formula'  
lda_schafer(formula, data, prior = NULL, ...)  
## S3 method for class 'lda_schafer'  
predict(object, newdata, type = c("class", "prob", "score"), ...)
```

**Arguments**

- `x`: Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.
- `...`: Options passed to `corpcor::invcov.shrink()`
- `y`: Vector of class labels for each training observation. Only complete data are retained.
- `prior`: Vector with prior probabilities for each class. If NULL (default), then equal probabilities are used. See details.
- `formula`: A formula of the form `groups ~ x1 + x2 + ...` That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.
- `data`: data frame from which variables specified in `formula` are preferentially to be taken.
- `object`: Fitted model object
- `newdata`: Matrix or data frame of observations to predict. Each row corresponds to a new observation.
- `type`: Prediction type: either "class", "prob", or "score".
Details

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

Value

\( \text{lda_schafer} \) object that contains the trained classifier

References


Examples

```r
library(modeldata)
data(penguins)
pred_rows <- seq(1, 344, by = 20)
penguins <- penguins[, c("species", "body_mass_g", "flipper_length_mm")]
lda_schafer_out <- lda_schafer(species ~ ., data = penguins[-pred_rows, ])
predicted <- predict(lda_schafer_out, penguins[pred_rows, -1], type = "class")

lda_schafer_out2 <- lda_schafer(x = penguins[-pred_rows, -1], y = penguins$species[-pred_rows])
predicted2 <- predict(lda_schafer_out2, penguins[pred_rows, -1], type = "class")
all.equal(predicted, predicted2)
```

Description

Given a set of training data, this function builds the Shrinkage-based Diagonal Linear Discriminant Analysis (SDLDA) classifier, which is based on the DLDA classifier, often attributed to Dudoit et al. (2002). The DLDA classifier belongs to the family of Naive Bayes classifiers, where the distributions of each class are assumed to be multivariate normal and to share a common covariance matrix. To improve the estimation of the pooled variances, Pang et al. (2009) proposed the SDLDA classifier which uses a shrinkage-based estimators of the pooled covariance matrix.

The SDLDA classifier is a modification to LDA, where the off-diagonal elements of the pooled sample covariance matrix are set to zero. To improve the estimation of the pooled variances, we use a shrinkage method from Pang et al. (2009).
Usage

```r
lda_shrink_cov(x, ...)  
## Default S3 method:
lda_shrink_cov(x, y, prior = NULL, num_alphas = 101, ...)
## S3 method for class 'formula'
lda_shrink_cov(formula, data, prior = NULL, num_alphas = 101, ...)
## S3 method for class 'lda_shrink_cov'
predict(object, newdata, type = c("class", "prob", "score"), ...)
```

Arguments

- `x`: Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.
- `...`: additional arguments (not currently used).
- `y`: Vector of class labels for each training observation. Only complete data are retained.
- `prior`: Vector with prior probabilities for each class. If NULL (default), then equal probabilities are used. See details.
- `num_alphas`: the number of values used to find the optimal amount of shrinkage.
- `formula`: A formula of the form `groups ~ x1 + x2 + ...`. That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.
- `data`: data frame from which variables specified in `formula` are preferentially to be taken.
- `object`: Fitted model object
- `newdata`: Matrix or data frame of observations to predict. Each row corresponds to a new observation.
- `type`: Prediction type: either "class", "prob", or "score".

Details

The DLDA classifier is a modification to the well-known LDA classifier, where the off-diagonal elements of the pooled covariance matrix are assumed to be zero – the features are assumed to be uncorrelated. Under multivariate normality, the assumption uncorrelated features is equivalent to the assumption of independent features. The feature-independence assumption is a notable attribute of the Naive Bayes classifier family. The benefit of these classifiers is that they are fast and have much fewer parameters to estimate, especially when the number of features is quite large.

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 probabilities should be nonnegative and sum to one.

**Value**

`lda_shrink_cov` object that contains the trained SDLDA classifier

**References**


**Examples**

```r
library(modeldata)
data(penguins)
pred_rows <- seq(1, 344, by = 20)
penguins <- penguins[, c("species", "body_mass_g", "flipper_length_mm")]
sdlda_out <- lda_shrink_cov(species ~ ., data = penguins[-pred_rows, ])
predicted <- predict(sdlda_out, penguins[pred_rows, -1], type = "class")

sdlda_out2 <- lda_shrink_cov(x = penguins[-pred_rows, -1], y = penguins$species[-pred_rows])
predicted2 <- predict(sdlda_out2, penguins[pred_rows, -1], type = "class")
all.equal(predicted, predicted2)
```

**Description**

Given a set of training data, this function builds the Shrinkage-mean-based Diagonal Linear Discriminant Analysis (SmDLDA) classifier from Tong, Chen, and Zhao (2012). The SmDLDA classifier incorporates a Lindley-type shrunken mean estimator into the DLDA classifier from Dudoit et al. (2002). For more about the DLDA classifier, see `lda_diag()`.

The SmDLDA classifier is a modification to LDA, where the off-diagonal elements of the pooled sample covariance matrix are set to zero.
Usage

lda_shrink_mean(x, ...)

## Default S3 method:
lda_shrink_mean(x, y, prior = NULL, ...)

## S3 method for class 'formula'
lda_shrink_mean(formula, data, prior = NULL, ...)

## S3 method for class 'lda_shrink_mean'
predict(object, newdata, type = c("class", "prob", "score"), ...)

Arguments

x Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.

... additional arguments (not currently used).

y Vector of class labels for each training observation. Only complete data are retained.

prior Vector with prior probabilities for each class. If NULL (default), then equal probabilities are used. See details.

formula A formula of the form groups ~ x1 + x2 + ... That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.

data data frame from which variables specified in formula are preferentially to be taken.

object Fitted model object

newdata Matrix or data frame of observations to predict. Each row corresponds to a new observation.

type Prediction type: either "class", "prob", or "score".

Details

The DLDA classifier belongs to the family of Naive Bayes classifiers, where the distributions of each class are assumed to be multivariate normal and to share a common covariance matrix.

The DLDA classifier is a modification to the well-known LDA classifier, where the off-diagonal elements of the pooled sample covariance matrix are assumed to be zero – the features are assumed to be uncorrelated. Under multivariate normality, the assumption uncorrelated features is equivalent to the assumption of independent features. The feature-independence assumption is a notable attribute of the Naive Bayes classifier family. The benefit of these classifiers is that they are fast and have much fewer parameters to estimate, especially when the number of features is quite large.

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

\textbf{Value}

\texttt{lda_shrink\_mean} object that contains the trained SmDLDA classifier

\textbf{References}


\textbf{Examples}

\begin{verbatim}
library(modeldata)
data(penguins)
pred_rows <- seq(1, 344, by = 20)
penguins <- penguins[, c("species", "body_mass_g", "flipper_length_mm")]
smdlda_out <- lda_shrink_mean(species ~ ., data = penguins[-pred_rows, ])
predicted <- predict(smdlda_out, penguins[pred_rows, -1], type = "class")
smdlda_out2 <- lda_shrink_mean(x = penguins[-pred_rows, -1], y = penguins$species[-pred_rows])
predicted2 <- predict(smdlda_out2, penguins[pred_rows, -1], type = "class")
all.equal(predicted, predicted2)
\end{verbatim}
covariance matrix. When the pooled sample covariance matrix is singular, the linear discriminant function is incalculable. This function replaces the pooled sample covariance matrix with a regularized estimator from Thomaz et al. (2006), where the smallest eigenvalues are replaced with the average eigenvalue. Specifically, small eigenvalues here means that the eigenvalues are less than the average eigenvalue.

Usage

```r
lda_thomaz(x, ...)
```

## Default S3 method:
```r
lda_thomaz(x, y, prior = NULL, ...)
```

## S3 method for class 'formula'
```r
lda_thomaz(formula, data, prior = NULL, ...)
```

## S3 method for class 'lda_thomaz'
```r
predict(object, newdata, type = c("class", "prob", "score"), ...)
```

Arguments

- `x`: Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.
- `...`: additional arguments (not currently used).
- `y`: Vector of class labels for each training observation. Only complete data are retained.
- `prior`: Vector with prior probabilities for each class. If NULL (default), then equal probabilities are used. See details.
- `formula`: A formula of the form `groups ~ x1 + x2 + ...`. That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.
- `data`: data frame from which variables specified in formula are preferentially to be taken.
- `object`: Fitted model object
- `newdata`: Matrix or data frame of observations to predict. Each row corresponds to a new observation.
- `type`: Prediction type: either "class", prob", or "score".

Details

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 probabilities should be nonnegative and sum to one.

**Value**

lda_thomaz object that contains the trained classifier

**References**


**Examples**

```r
library(modeldata)
data(penguins)
pred_rows <- seq(1, 344, by = 20)
penguins <- penguins[, c("species", "body_mass_g", "flipper_length_mm")]
lda_thomaz_out <- lda_thomaz(species ~ ., data = penguins[-pred_rows, ])
predicted <- predict(lda_thomaz_out, penguins[pred_rows, -1], type = "class")

lda_thomaz_out2 <- lda_thomaz(x = penguins[-pred_rows, -1], y = penguins$species[-pred_rows])
predicted2 <- predict(lda_thomaz_out2, penguins[pred_rows, -1], type = "class")
all.equal(predicted, predicted2)
```

---

**log_determinant**

*Computes the log determinant of a matrix.*

**Description**

Computes the log determinant of a matrix.

**Usage**

`log_determinant(x)`

**Arguments**

- `x`  
  matrix

**Value**

log determinant of x
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**

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

**Arguments**

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

**Value**

formula with no intercept term

**Examples**

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

---

**plot.rda_high_dim_cv**  
*Plots a heatmap of cross-validation error grid for a HDRDA classifier object.*

**Description**

Uses `ggplot2::ggplot()` to plot a heatmap of the training error grid.

**Usage**

```r
## S3 method for class 'rda_high_dim_cv'
plot(x, ...)
```

**Arguments**

- `x` object to plot
- `...` unused
Value

A ggplot object.

posterior_probs

Computes posterior probabilities via Bayes Theorem under normality

Usage

posterior_probs(x, means, covs, priors)

Arguments

x                matrix of observations
means            list of means for each class
covs             list of covariance matrices for each class
priors           list of prior probabilities for each class

Value

matrix of posterior probabilities for each observation

qda_diag

Diagonal Quadratic Discriminant Analysis (DQDA)

Description

Given a set of training data, this function builds the Diagonal Quadratic Discriminant Analysis (DQDA) classifier, which is often attributed to Dudoit et al. (2002). The DQDA classifier belongs to the family of Naive Bayes classifiers, where the distributions of each class are assumed to be multivariate normal. Note that the DLDA classifier is a special case of the DQDA classifier.

The DQDA classifier is a modification to QDA, where the off-diagonal elements of the pooled sample covariance matrix are set to zero.
qda_diag

Usage

qda_diag(x, ...)

## Default S3 method:
qda_diag(x, y, prior = NULL, ...)

## S3 method for class 'formula'
qda_diag(formula, data, prior = NULL, ...)

## S3 method for class 'qda_diag'
predict(object, newdata, type = c("class", "prob", "score"), ...)

Arguments

x Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.

... additional arguments (not currently used).

y Vector of class labels for each training observation. Only complete data are retained.

prior Vector with prior probabilities for each class. If NULL (default), then equal probabilities are used. See details.

formula A formula of the form groups ~ x1 + x2 + ... That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.

data data frame from which variables specified in formula are preferentially to be taken.

object Fitted model object

newdata Matrix or data frame of observations to predict. Each row corresponds to a new observation.

type Prediction type: either "class", "prob", or "score".

Details

The DQDA classifier is a modification to the well-known QDA classifier, where the off-diagonal elements of each class covariance matrix are assumed to be zero – the features are assumed to be uncorrelated. Under multivariate normality, the assumption uncorrelated features is equivalent to the assumption of independent features. The feature-independence assumption is a notable attribute of the Naive Bayes classifier family. The benefit of these classifiers is that they are fast and have much fewer parameters to estimate, especially when the number of features is quite large.

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 probabilities should be nonnegative and sum to one.

**Value**

`qda_diag` object that contains the trained DQDA classifier

**References**


**Examples**

```r
library(modeldata)
data(penguins)
pred_rows <- seq(1, 344, by = 20)
penguins <- penguins[, c("species", "body_mass_g", "flipper_length_mm")]
dqda_out <- qda_diag(species ~ ., data = penguins[-pred_rows, ])
predicted <- predict(dqda_out, penguins[pred_rows, -1], type = "class")

dqda_out2 <- qda_diag(x = penguins[-pred_rows, -1], y = penguins$species[-pred_rows])
predicted2 <- predict(dqda_out2, penguins[pred_rows, -1], type = "class")
all.equal(predicted, predicted2)
```

---

**Description**

Given a set of training data, this function builds the Shrinkage-based Diagonal Quadratic Discriminant Analysis (SDQDA) classifier, which is based on the DQDA classifier, often attributed to Dudoit et al. (2002). The DQDA classifier belongs to the family of Naive Bayes classifiers, where the distributions of each class are assumed to be multivariate normal. To improve the estimation of the class variances, Pang et al. (2009) proposed the SDQDA classifier which uses a shrinkage-based estimators of each class covariance matrix.

The SDQDA classifier is a modification to QDA, where the off-diagonal elements of the pooled sample covariance matrix are set to zero. To improve the estimation of the pooled variances, we use a shrinkage method from Pang et al. (2009).
qda_shrink_cov

Usage

qda_shrink_cov(x, ...)

## Default S3 method:
qda_shrink_cov(x, y, prior = NULL, num_alphas = 101, ...)

## S3 method for class 'formula'
qda_shrink_cov(formula, data, prior = NULL, num_alphas = 101, ...)

## S3 method for class 'qda_shrink_cov'
predict(object, newdata, type = c("class", "prob", "score"), ...)

Arguments

x Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.

... additional arguments (not currently used).

y Vector of class labels for each training observation. Only complete data are retained.

prior Vector with prior probabilities for each class. If NULL (default), then equal probabilities are used. See details.

num_alphas the number of values used to find the optimal amount of shrinkage

formula A formula of the form groups ~ x1 + x2 + ... That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.

data data frame from which variables specified in formula are preferentially to be taken.

object Fitted model object

newdata Matrix or data frame of observations to predict. Each row corresponds to a new observation.

type Prediction type: either "class", "prob", or "score".

Details

The DQDA classifier is a modification to the well-known QDA classifier, where the off-diagonal elements of the pooled covariance matrix are assumed to be zero – the features are assumed to be uncorrelated. Under multivariate normality, the assumption uncorrelated features is equivalent to the assumption of independent features. The feature-independence assumption is a notable attribute of the Naive Bayes classifier family. The benefit of these classifiers is that they are fast and have much fewer parameters to estimate, especially when the number of features is quite large.

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` probabilities should be nonnegative and sum to one.

**Value**

`qda_shrink_cov` object that contains the trained SDQDA classifier

**References**


**Examples**

```r
library(modeldata)
data(penguins)
pred_rows <- seq(1, 344, by = 20)
penguins <- penguins[, c("species", "body_mass_g", "flipper_length_mm")]
# set.seed(42)
sdqda_out <- qda_shrink_cov(species ~ ., data = penguins[-pred_rows, ])
predicted <- predict(sdqda_out, penguins[pred_rows, -1], type = "class")

sdqda_out2 <- qda_shrink_cov(x = penguins[-pred_rows, -1], y = penguins$species[-pred_rows])
predicted2 <- predict(sdqda_out2, penguins[pred_rows, -1], type = "class")
all.equal(predicted, predicted2)
```

**Description**

Given a set of training data, this function builds the Shrinkage-mean-based Diagonal Quadratic Discriminant Analysis (SmDQDA) classifier from Tong, Chen, and Zhao (2012). The SmDQDA classifier incorporates a Lindley-type shrunken mean estimator into the DQDA classifier from Dudoit et al. (2002). For more about the DQDA classifier, see `qda_diag()`.

The SmDQDA classifier is a modification to QDA, where the off-diagonal elements of the pooled sample covariance matrix are set to zero.
Usage

qda_shrink_mean(x, ...)

## Default S3 method:
qda_shrink_mean(x, y, prior = NULL, ...)

## S3 method for class 'formula'
qda_shrink_mean(formula, data, prior = NULL, ...)

## S3 method for class 'qda_shrink_mean'
predict(object, newdata, type = c("class", "prob", "score"), ...)

Arguments

x  Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.

...  additional arguments (not currently used).

y  Vector of class labels for each training observation. Only complete data are retained.

prior  Vector with prior probabilities for each class. If NULL (default), then equal probabilities are used. See details.

formula  A formula of the form groups ~ x1 + x2 + ... That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.

data  data frame from which variables specified in formula are preferentially to be taken.

object  Fitted model object

newdata  Matrix or data frame of observations to predict. Each row corresponds to a new observation.

type  Prediction type: either "class", "prob", or "score".

Details

The DQDA classifier is a modification to the well-known QDA classifier, where the off-diagonal elements of each class covariance matrix are assumed to be zero – the features are assumed to be uncorrelated. Under multivariate normality, the assumption uncorrelated features is equivalent to the assumption of independent features. The feature-independence assumption is a notable attribute of the Naive Bayes classifier family. The benefit of these classifiers is that they are fast and have much fewer parameters to estimate, especially when the number of features is quite large.

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` probabilities should be nonnegative and sum to one.

**Value**

`qda_shrink_mean` object that contains the trained SmDQDA classifier

**References**


**Examples**

```r
library(modeldata)
data(penguins)
pred_rows <- seq(1, 344, by = 20)
penguins <- penguins[, c("species", "body_mass_g", "flipper_length_mm")]
smdqda_out <- qda_shrink_mean(species ~ ., data = penguins[-pred_rows, ])
predicted <- predict(smdqda_out, penguins[pred_rows, -1], type = "class")
smdqda_out2 <- qda_shrink_mean(x = penguins[-pred_rows, -1], y = penguins$species[-pred_rows])
predicted2 <- predict(smdqda_out2, penguins[pred_rows, -1], type = "class")
all.equal(predicted, predicted2)
```

---

**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 $p \times p$ real matrix. Then, we compute the quadratic form $q = x'Ax$.

**Usage**

`quadform(A, x)`

**Arguments**

- `A` : matrix of dimension $p \times p$
- `x` : vector of length $p`
Details

A naive way to compute the quadratic form is to explicitly write \( t(x) \ A^{-1} \ x \), but for large \( p \), this operation is inefficient. We provide a more efficient method below.

Note that we have adapted the code from: https://stat.ethz.ch/pipermail/r-help/2005-November/081940.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 rda_high_dim().

Usage
rda_cov(x, y, lambda = 1)

Arguments
- x: Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.
- 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

rda_high_dim
High-Dimensional Regularized Discriminant Analysis (HDRDA)

Description
Given a set of training data, this function builds the HDRDA classifier from Ramey, Stein, and Young (2017). Specially designed for small-sample, high-dimensional data, the HDRDA classifier incorporates dimension reduction and covariance-matrix shrinkage to enable a computationally efficient classifier.

For a given rda_high_dim object, we predict the class of each observation (row) of the the matrix given in newdata.
rda_high_dim

Usage

rda_high_dim(x, ...)

## Default S3 method:
rda_high_dim(
  x,
  y,
  lambda = 1,
  gamma = 0,
  shrinkage_type = c("ridge", "convex"),
  prior = NULL,
  tol = 1e-06,
  ...
)

## S3 method for class 'formula'
rda_high_dim(formula, data, ...)

## S3 method for class 'rda_high_dim'
predict(
  object,
  newdata,
  projected = FALSE,
  type = c("class", "prob", "score"),
  ...
)

Arguments

x Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.

... additional arguments (not currently used).

y vector of class labels for each training observation

lambda the HDRDA pooling parameter. Must be between 0 and 1, inclusively.

gamma a numeric values used for the shrinkage parameter.

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. (2017) for details.

prior vector with prior probabilities for each class. If NULL (default), then the sample proportion of observations belonging to each class equal probabilities are used. See details.

tol a threshold for determining nonzero eigenvalues.

formula A formula of the form groups ~ x1 + x2 + ... That is, the response is the grouping factor and the right hand side specifies the (non-factor) discriminators.

data data frame from which variables specified in formula are preferentially to be taken.
object Object of type rda_high_dim that contains the trained HDRDA classifier
newdata Matrix or data frame of observations to predict. Each row corresponds to a new observation.
projected logical indicating whether newdata have already been projected to a q-dimensional subspace. This argument can yield large gains in speed when the linear transformation has already been performed.
type Prediction type: either "class", "prob", or "score".

details

The HDRDA classifier utilizes a covariance-matrix estimator that is a convex combination of the covariance-matrix estimators used in the Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) classifiers. For each of the K classes given in y, (k = 1, ..., K), we first define this convex combination as

$$\hat{\Sigma}_k(\lambda) = (1 - \lambda)\hat{\Sigma}_k + \lambda\hat{\Sigma},$$

where $\lambda \in [0, 1]$ is the pooling parameter. We then calculate the covariance-matrix estimator

$$\tilde{\Sigma}_k = \alpha_k\hat{\Sigma}_k(\lambda) + \gamma I_p,$$

where $I_p$ is the $p \times p$ identity matrix. The matrix $\tilde{\Sigma}_k$ is substituted into the HDRDA classifier. See Ramey et al. (2017) for more details.

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.

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 probabilities should be nonnegative and sum to one. The order of the prior probabilities is assumed to match the levels of factor(y).

value

rda_high_dim object that contains the trained HDRDA classifier

list with predicted class and discriminant scores for each of the K classes

references


rda_high_dim_cv

*rda_high_dim_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**

```r
rda_high_dim_cv(  
x,  
y,  
num_folds = 10,  
num_lambda = 21,  
num_gamma = 8,  
shrinkage_type = c("ridge", "convex"),  
verbose = FALSE,  
...  
)
```

**Arguments**

- `x` Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.
- `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. (2017) for details.
- `verbose` If set to TRUE, summary information will be outputted as the optimal model is being determined.
- `...` Options passed to `rda_high_dim()`.

**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.
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 `rda_high_dim()`.

**Usage**

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

**Arguments**

- `x`: Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.
- `y`: Vector of class labels for each training observation.
- `lambda`: The RDA pooling parameter. Must be between 0 and 1, inclusively.

**Value**

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

**References**


---

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**

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

- \( x \)  
  Matrix or data frame containing the training data. The rows are the sample observations, and the columns are the features. Only complete data are retained.

- \( y \)  
  vector of class labels for each training observation

- \( \text{cov} \)  
  logical. Should the sample covariance matrices be computed? (Default: yes)

- \( \text{prior} \)  
  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 \texttt{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 \texttt{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, \( \text{prior} \), contains the \textit{a priori} class membership for each class. If \( \text{prior} \) is NULL (default), the class membership probabilities are estimated as the sample proportion of observations belonging to each class. Otherwise, \( \text{prior} \) should be a vector with the same length as the number of classes in \( y \). The \( \text{prior} \) probabilities 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

```r
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

A 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


---

**solve_chol**

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

---

**Description**


**Usage**

```r
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**

---

**two_class_sim_data**  
*Example bivariate classification data from caret*

**Description**
Example bivariate classification data from caret

**Details**
These data were generated using by invoking the twoClassSim() function in the caret package.

**Value**
two_class_sim_data  
a tibble
Examples

```r
data(two_class_sim_data)
```

**update_rda_high_dim**  
*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**

```r
update_rda_high_dim(obj, lambda = 1, gamma = 0)
```

**Arguments**

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

**Value**

a `rda_high_dim` 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**

```r
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.
**var_shrinkage**

**Value**

a vector of the shrunken variances for each feature.

**References**

Index

* datasets
  two_class_sim_data, 49
  center_data, 3
corpcor::fast.svd(), 5
corpcor::invcov.shrink(), 25
cov_autocorrelation, 3
cov_block_autocorrelation, 4
cov_block_autocorrelation(), 12
cov_eigen, 5
cov_intraclass, 6
cov_list, 7
cov_mle, 7
cov_pool, 8
cov_shrink_diag, 8
cv_partition, 9
diag_estimates, 10
dmvnorm_diag, 11

generate_blockdiag, 12
generate_intraclass, 13
ggplot2::ggplot2(), 33

h, 14

lda_diag, 15
lda_diag(), 28
lda_eigen, 17
lda_emp_bayes, 19
lda_emp_bayes_eigen, 21
lda_pseudo, 23
lda_schafer, 24
lda_shrink_cov, 26
lda_shrink_mean, 28
lda_thomaz, 30
log_determinant, 32

no_intercept, 33

plot.rda_high_dim_cv, 33

posterior_probs, 34

predict.lda_diag(lda_diag), 15
predict.lda_eigen(lda_eigen), 17
predict.lda_emp_bayes(lda_emp_bayes), 19
predict.lda_emp_bayes_eigen
(lda_emp_bayes_eigen), 21
predict.lda_pseudo(lda_pseudo), 23
predict.lda_schafer(lda_schafer), 24
predict.lda_shrink_cov
(lda_shrink_cov), 26
predict.lda_shrink_mean
(lda_shrink_mean), 28
predict.lda_thomaz(lda_thomaz), 30
predict.qda_diag(qda_diag), 34
predict.qda_shrink_cov
(qda_shrink_cov), 36
predict.qda_shrink_mean
(qda_shrink_mean), 38
predict.rda_high_dim(rda_high_dim), 42

qda_diag, 34
qda_diag(), 38
qda_shrink_cov, 36
qda_shrink_mean, 38
quadform, 40
quadform_inv, 41

rda_cov, 42
rda_high_dim, 42
rda_high_dim(), 42, 45, 46
rda_high_dim_cv, 45
rda_weights, 46
regdiscrim_estimates, 46
risk_stein, 47

solve(), 48
solve_chol, 48

tong_mean_shrinkage, 49
two_class_sim_data, 49
update_rda_high_dim, 50
var_shrinkage, 50