Package ‘VDA’
March 20, 2013

Type Package
Title VDA
Version 1.02
Date 2013-03-20
Author Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu
Maintainer Edward Grant <edward.m.grant@gmail.com>
Description Multicategory Vertex Discriminant Analysis
License GPL-2
LazyLoad yes
LazyData yes
Depends rgl
NeedsCompilation yes
Repository CRAN
Date/Publication 2013-03-20 17:38:18

R topics documented:
cv.vda.le ......................................................... 2
cv.vda.r .......................................................... 4
plot.cv.vda.le ................................................... 5
plot.cv.vda.r ..................................................... 7
predict.vda.le ................................................... 8
predict.vda.r ..................................................... 9
print.vda.le ...................................................... 10
print.vda.r ....................................................... 11
summary.vda.le ............................................... 13
summary.vda.r ................................................ 14
Choose the optimal pair of lambdas, $\lambda_1$ and $\lambda_2$

**Description**

Use k-fold validation to choose the optimal values for the tuning parameters $\lambda_1$ and $\lambda_2$ to be used in Multicategory Vertex Discriminant Analysis (vda.le).

**Usage**

```r
cv.vda.le(x, y, kfold, lam.vec.1, lam.vec.2)
```

**Arguments**

- `x`: $n \times p$ matrix or data frame containing the cases for each feature. The rows correspond to cases and the columns to the features. Intercept column is not included in this.
- `y`: $n \times 1$ vector representing the outcome variable. Each element denotes which one of the $k$ classes that case belongs to.
- `kfold`: The number of folds to use for the k-fold validation for each set of $\lambda_1$ and $\lambda_2$.
- `lam.vec.1`: A vector containing the set of all values of $\lambda_1$, from which VDA will be conducted. To use only Euclidean penalization, set `lam.vec.2`=0.
- `lam.vec.2`: A vector containing the set of all values of $\lambda_2$, from which VDA will be conducted. vda.le is relatively insensitive to lambda values, so it is recommended that a vector of few values is used. The default value is 0.01. To use only Lasso penalization, set `lam.vec.1`=0.

**Details**

For each pair of $(\lambda_1, \lambda_2)$, k-fold cross-validation will be conducted and the corresponding average testing error over the k folds will be recorded. $\lambda_1$ represents the parameter for the lasso penalization, while $\lambda_2$ represents the parameter for the group euclidean penalization. To use only Lasso penalization, set `lam.vec.2`=0. To use only Euclidean penalization, set `lam.vec.1`=0. The optimal pair is considered the pair of values that give the smallest testing error over the cross validation.

To view a plot of the cross validation errors across lambda values, see `plot.cv.vda.le`. 
**Value**

- **kfold** The number of folds used in k-fold cross validation
- **lam.vec.1** The user supplied vector of $\lambda_1$ values
- **lam.vec.2** The user supplied vector of $\lambda_2$ values
- **error.cv** A matrix of average testing errors. The rows correspond to $\lambda_1$ values and the columns correspond to $\lambda_2$ values.
- **lam.opt** The pair of $\lambda_1$ and $\lambda_2$ values that return the lowest testing error across k-fold cross validation.

**Author(s)**

Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu

Maintainer: Edward Grant <edward.m.grant@gmail.com>

**References**


**See Also**

- vda.le
- plot.cv.vda.le

**Examples**

```r
# load data
data(zoo)

d # feature matrix without intercept
x <- zoo[,2:17]

d # class vector
y <- zoo[,18]

# Searching for the best pair, using both lasso and euclidean penalizations
cv <- cv.vda.le(x, y, kfold = 3, lam.vec.1 = (1:5)/1/zero.noslash/zero.noslash, lam.vec.2 = (1:5)/1/zero.noslash)
out <- vda.le(x,y,cv$lam.opt[1],cv$lam.opt[2])

d # To search for the best pair, using ONLY lasso penalization
cvlasso <- cv.vda.le(x, y, kfold = 3, lam.vec.1 = exp(1:1)/1/zero.noslash/zero.noslash/zero.noslash)
cvlasso$lam.opt

# To search for the best pair, using ONLY euclidean penalization (remove #)
#cveuclidian <- cv.vda.le(x, y, kfold = 3, lam.vec.1 = 0, lam.vec.2 = exp(1:10)/1000)
#cveuclidian$lam.opt
```
Choose \( \lambda \) using K-fold cross validation

Description

Choose the optimal tuning parameter \( \lambda \) for Vertex Discriminant Analysis by using K-fold cross validation.

Usage

\[
\text{cv.vda.r}(x, y, k, \text{lam.vec}) \\
\text{cv.vda}(x, y, k, \text{lam.vec})
\]

Arguments

\(x\) \hspace{1cm} n x p matrix or data frame containing the cases for each feature. The rows correspond to cases and the columns to the features. Intercept column is not included in this.

\(y\) \hspace{1cm} n x 1 vector representing the outcome variable. Each element denotes which one of the \( k \) classes that case belongs to.

\(k\) \hspace{1cm} The number of folds to be used in cross-validation.

\(\text{lam.vec}\) \hspace{1cm} A vector containing the set of all values of \( \lambda \), from which VDA will be conducted.

Details

K-fold cross validation to select optimal lambda for use in Vertex Discriminant Analysis (vda.r). The optimal value is considered the lambda value that returns the lowest testing error over the cross validation. If more than one lambda value give the minimum testing error, the largest lambda is selected.

A plot of the cross validation errors can be viewed through \text{plot.cv.vda.r}.

Value

\(k\) \hspace{1cm} The value of \( K \) used for the K-fold cross validation.

\(\text{lam.vec}\) \hspace{1cm} The values of lambda tested.

\(\text{mean.error}\) \hspace{1cm} The mean error corresponding to each lambda across k-folds

\(\text{lam.opt}\) \hspace{1cm} The determined lambda value among \(\text{lam.vec}\) that returns the smallest prediction error. This value is the optimal lambda value for use in \text{link(vda.r)}.

\(\text{error.cv}\) \hspace{1cm} The prediction error matrix returned by cross validation method.

Author(s)

Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu

Maintainer: Edward Grant <edward.m.grant@gmail.com>

See Also

vda.r plot.cv.vda.r

Examples

# load data
data(zoo)

# feature matrix without intercept
x <- zoo[,2:17]

# class vector
y <- zoo[,18]

# lambda vector
lam.vec <- (1:10)/10

# searching for the best lambda with 10-fold cross validation
obj <- cv.vda.r(x, y, 10, lam.vec)

# run VDA
out <- vda.r(x,y,obj$lam.opt)

Description

Plot a the cross validation error across lambda values

Usage

## S3 method for class 'cv.vda.le'
plot(x, ...)

Arguments

x               Object of class 'cv.vda.le', the result of a call to cv.vda.le.
...             Not used.
Details

3D plots the k-fold cross validation testing error for values across different lambda1 and lambda2 values. Use `cv.vda.le` to produce the object of class "cv.vda.le".

When lam.vec.1 or lam.vec.2 is set to 0, a 2D plot will be produced.

Author(s)

Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu

Maintainer: Edward Grant <edward.m.grant@gmail.com>

References


See Also

`vda.le, cv.vda.le`

Examples

```r
# load data
data(zoo)

# feature matrix without intercept
x <- zoo[,2:17]

# class vector
y <- zoo[,18]

# searching for the best pair, using both lasso and euclidean penalizations
cv <- cv.vda.le(x, y, kfold=3, lam.vec.1=(1:5)/100, lam.vec.2=(1:5)/100)
plot(cv)
out <- vda.le(x, y, cv$lam.opt[1], cv$lam.opt[2])

# searching for the best pair, using ONLY lasso penalization
classo <- cv.vda.le(x, y, kfold=3, lam.vec.1=exp(1:10)/1000, lam.vec.2=0)
plot(classo)
classo$lam.opt

# searching for the best pair, using ONLY euclidean penalization (remove #)
cveuclidian <- cv.vda.le(x, y, kfold=3, lam.vec.1=0, lam.vec.2=exp(1:10)/1000)
plot(cveuclidian)
cveuclidian$lam.opt
```
plot.cv.vda.r

Plot a cv.vda.r object

Description
Plot a the cross validation error across lambda values

Usage
```r
## S3 method for class 'cv.vda.r'
plot(x, ...)  # S3 method for class 'cv.vda.r'
plot(x, ...)  # S3 method for class 'cv.vda.r'
```

Arguments
- `x` Object of class `cv.vda.r`, the result of a call to `cv.vda.r`.
- `...` Not used.

Details
Plots the k-fold cross validation testing error for values across a different lambda values. Use `cv.vda.r` to produce the object of class "cv.vda.r."

Author(s)
Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu
Maintainer: Edward Grant <edward.m.grant@gmail.com>

References

See Also
`vda.r`, `cv.vda.r`

Examples
```r
# load data
data(zoo)

# feature matrix without intercept
x <- zoo[,2:17]

# class vector
y <- zoo[,18]

# lambda vector
```
lam.vec <- (1:10)/10

#run 10 fold cross validation across lambdas
object <- cv.vda.r(x, y, 10, lam.vec)

#plot
plot(object)

predict.vda.le

**Predict a vda.le object.**

**Description**

The predict function for a vda.le object.

**Usage**

```r
## S3 method for class 'vda.le'
predict(object, newdata=NULL, ...)
```

**Arguments**

- `object`: An object of class 'vda.le', usually the result of a call to `vda.le`.
- `newdata`: An optional \( n \times p \) matrix or data frame containing new data to be classified using vertex discriminant analysis. The data must contain the same number of attributes as the training data. If `newdata` is omitted, the training data is used.
- `...`: Not used.

**Details**

The prediction function for Vertex Discriminant Analysis (vda.le). Returns 1 x \( n \) vector in which each element represents the predicted value for the corresponding case.

**Author(s)**

Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu

Maintainer: Edward Grant <edward.m.grant@gmail.com>

**References**


**See Also**

`vda.le`, `summary.vda.le`, `print.vda.le`
Examples

```r
# load data
data(zoo)

# feature matrix without intercept
x <- zoo[,2:17]

# class vector
y <- zoo[,18]

#run VDA
out <- vda.le(x,y)

onecase <- matrix(c(0,1,1,0,0,0,1,1,0,0,2,1,0,1),1,16)
predict(out, onecase)

fivecases <- matrix(0,5,16)
fivecases[1,] <- c(0,0,1,0,0,1,1,0,0,1,0,1,0,0)
fivecases[2,] <- c(0,0,1,0,1,0,1,0,0,1,0,0,6,0,0)
fivecases[3,] <- c(0,0,1,0,1,1,0,0,0,0,0,4,0,0,0)
fivecases[4,] <- c(1,0,0,1,0,0,1,1,1,0,4,1,1,1)
fivecases[5,] <- c(0,0,0,0,0,1,1,1,1,0,0,1,0,1,0)
predict(out, fivecases)
```

---

**predict.vda.r**  
*Predict a vda.r object.*

Description

The predict function for a vda.r object.

Usage

```r
## S3 method for class 'vda.r'
predict(object, newdata=NULL, ...)
```

Arguments

- `object`: An object of class `vda.r`, usually the result of a call to `vda.r`.
- `newdata`: An optional n x p matrix or data frame containing new data to be classified using VDA. The data must contain the same number of attributes as the training data. If `newdata` is omitted, the training data is used.
- `...`: Not used.

Details

The prediction function for Vertex Discriminant Analysis (`vda.r`). Returns 1 x n vector in which each element represents the predicted value for the corresponding case.
Author(s)
Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu
Maintainer: Edward Grant <edward.m.grant@gmail.com>

References

See Also
vda.r, summary.vda.r, print.vda.r

Examples
# load data
data(zoo)

# feature matrix without intercept
x <- zoo[,2:17]

# class vector
y <- zoo[,18]

#run VDA
out <- vda.r(x,y)
onecase <- matrix(c(0,1,0,0,0,1,0,1,0,1,0,1,0,2,1,0,1),1,16)
predict(out, onecase)

fivecases <- matrix(0,5,16)
fivecases[1,] <- c(0,0,1,0,0,1,1,0,0,1,0,1,0,1,0,0)
fivecases[2,] <- c(0,0,1,0,1,0,1,0,0,1,0,1,0,0,0,0)
fivecases[3,] <- c(0,0,1,0,0,1,1,0,0,0,0,0,4,0,0,0)
fivecases[4,] <- c(1,0,0,1,0,0,1,1,0,0,4,1,1,1)
fivecases[5,] <- c(0,0,0,0,1,1,1,1,0,1,0,1,0,0,0,0)
predict(out, fivecases)

print.vda.le

Print a vda.le object

Description
The default print method for a vda.le object.

Usage
## S3 method for class 'vda.le'
print(x, ...)

print.vda.le
print.vda.r

Arguments

  x  Object of class 'vda.le', usually the result of a call to vda.le.
  ... Not used.

Details

  Prints out the predicted classes for given training data found using Vertex Discriminant Analysis. summary.vda.le provides more detailed information about the VDA object x.

Author(s)

  Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu
  Maintainer: Edward Grant <edward.m.grant@gmail.com>

References


See Also

  vda.le, summary.vda.le

Examples

  # load data
data(zoo)

  # feature matrix without intercept
  x <- zoo[,2:17]

  # class vector
  y <- zoo[,18]

  #run VDA
  out <- vda.le(x, y)

  print(out)

print.vda.r  Print a vda.r object

Description

  The default print method for a vda.r object.
Usage

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

Arguments

- **x**
  - Object of class 'vda.r', usually the result of a call to `vda.r`.
- **...**
  - Not used.

Details

Prints out the predicted classes for given training data found using Vertex Discriminant Analysis. `summary.vda.r` provides more detailed information about the VDA object.

Author(s)

Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu

Maintainer: Edward Grant <edward.m.grant@gmail.com>

References


See Also

`vda.r`, `summary.vda.r`

Examples

```r
# load data
data(zoo)

# feature matrix without intercept
x <- zoo[,2:17]

# class vector
y <- zoo[,18]

#run VDA
out <- vda.r(x, y)

print(out)
```
Summary for a vda.le object

Description
Takes a fitted VDA object produced by vda.le and produces various useful summaries from it.

Usage
## S3 method for class 'vda.le'
summary(object, ...)

Arguments
object An object of class 'vda.le', usually the result of a call to vda.le.
...
Not used.

Details
The function prints the number of cases, the number of classes, and the number of features in object, of class vda.le. It also prints the lambda used in the analysis. Additionally, it prints the coefficients and the resulting predictions made by Vertex Discriminant Analysis on the training set and the following training error.

Author(s)
Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu
Maintainer: Edward Grant <edward.m.grant@gmail.com>

See Also
vda.le, print.vda.le

Examples

# load data
data(zoo)

# feature matrix without intercept
x<-zoo[,2:17]

# class vector
y<-zoo[,18]

#run VDA
out<-vda.le(x, y)

summary(out)
Description
Takes a fitted vda.r object produced by \texttt{vda.r} and produces various useful summaries from it.

Usage
\begin{verbatim}
## S3 method for class 'vda.r'
summary(object, ...)
\end{verbatim}

Arguments
\begin{itemize}
\item \texttt{object} An object of class 'vda.r', usually the result of a call to \texttt{vda.r}.
\item \texttt{...} Not used.
\end{itemize}

Details
The function prints the number of cases, the number of classes, and the number of features in \texttt{object}, of class \texttt{vda.r}. It also prints the lambda used in the analysis. Additionally, it prints the coefficients and the resulting predictions made by Vertex Discriminant Analysis on the training set and the following training error.

Author(s)
Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu
Maintainer: Edward Grant <edward.m.grant@gmail.com>

See Also
\texttt{vda.r, print.vda.r}

Examples
\begin{verbatim}
# load data
data(zoo)

# feature matrix without intercept
x<-zoo[,2:17]

# class vector
y<-zoo[,18]

#run VDA
out<-vda.r(x, y)
summary(out)
\end{verbatim}
Multicategory Vertex Discriminant Analysis

Description

This package provides functions to optimize and execute Multicategory Vertex Discriminant Analysis, a method of supervised learning for an outcome with $k$ predictor categories. Outcome classification is based on linear discrimination among the vertices of a regular simplex in a $k-1$-dimensional Euclidean space, where each vertex represents a different category.

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>VDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>1.02</td>
</tr>
<tr>
<td>Date:</td>
<td>2012-02-27</td>
</tr>
<tr>
<td>License:</td>
<td>GPL-2</td>
</tr>
<tr>
<td>LazyLoad:</td>
<td>yes</td>
</tr>
</tbody>
</table>

Author(s)

Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu

Maintainer: Edward Grant <edward.m.grant@gmail.com>

References


Examples

```r
# load dataset from package
data(zoo)

# matrix containing all predictor vectors
x <- zoo[,2:17]

# outcome class vector
y <- zoo[,18]

# run VDA
out <- vda.r(x, y)
```
Multicategory Vertex Discriminant Analysis (VDA) for High-Dimensional Data

Description

The method of vertex discriminant analysis (VDA) is ideally suited to handle multiple categories and an excess of predictors over training cases. vda.le is an elaboration of VDA that simultaneously conducts classification of \( k \) possible categories and variable selection of \( p \) features, based on a data set of \( n \) cases. Variable selection is imposed using \( L1 \) (Lasso) and group Euclidean penalties. To use only Lasso penalization, set \( \lambda2=0 \). To use only Euclidean penalization, set \( \lambda1=0 \).

Usage

\[
vda.le(x, y, \text{lambda1}, \text{lambda2})
\]

Arguments

- **x**: \( n \times p \) matrix or data frame containing the cases for each feature. The rows correspond to cases and the columns to the features. Intercept column is not included in this.
- **y**: \( n \times 1 \) vector representing the outcome variable. Each element denotes which one of the \( k \) classes that case belongs to.
- **lambda1**: Tuning parameter to control the lasso penalty. The default value is \( 1/n \). For determining the optimal \( \lambda1 \), refer to \text{cv.vda.le}.
- **lambda2**: Tuning parameter to control the Euclidean penalty. The default value is 0.01. For determining the optimal \( \lambda1 \), refer to \text{cv.vda.le}.

Details

vda.le carries out cyclic coordinate descent in the context of VDA to minimize the loss function. By adding lasso (\( L1 \)-norm) and group Euclidean penalties to the VDA loss function, unnecessary predictors are eliminated, adding parsimony and making the model more interpretable. Lasso penalties are applied to each predictor coefficient separately, while Euclidean penalties couples the coefficients of a single predictor and penalize the group collectively. If \( \lambda1=0 \), then the overall penalty reduces to only group penalties. When \( \lambda1=0 \), then the overall penalty reduces to the lasso. With these penalties in place, cyclic coordinate descent accelerates estimation of all coefficients.

Value

- **feature**: Feature matrix \( x \) with an intercept vector added as the first column. All entries in the first column should equal 1.
- **stand.feature**: The feature matrix where the all columns are standardized, with the exception of the intercept column which is left unstandardized.
- **class**: Class vector \( y \). All elements should be integers between 1 and \( \text{classes} \).
cases  Number of cases, \( n \).
classes  Number of classes, \( k \).
features  Number of features, \( p \).
lambda  Vector of tuning constants where the first component is \( \lambda_1 \) and the second is \( \lambda_2 \).
predicted  Vector of predicted category values based on VDA.
coefficient  The estimated coefficient matrix where the columns represent the coefficients for each predictor variable corresponding to \( k-1 \) outcome categories. The coefficient matrix is used for classifying new cases.
training_error_rate  The percentage of instances in the training set where the predicted outcome category is not equal to the case’s true category.
nonzeros  Number of feature coefficients retained in the model. Is equal to \( p - \) number of features eliminated by penalization.
selected  An integer vector which represents the attributes that were selected after penalization.
call  The matched call.

Author(s)
Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu
Maintainer: Edward Grant <edward.m.grant@gmail.com>

References

See Also
For determining the optimal values for \( \lambda_1 \) and \( \lambda_2 \), see cv.vda.le
For VDA without variable selection, see vda.r.

Examples
#load dataset from package
data(zoo)

#matrix containing all predictor vectors
x <- zoo[,2:17]

#outcome class vector
y <- zoo[,18]

#run VDA, Only Lasso Penalization
out<-vda.le(x,y,lambda2=0)

#run VDA, Only Euclidean Penalization
out<-vda.le(x,y,lambda1=0)

#run VDA, Lasso and Euclidean Penalization
out<-vda.le(x,y)

---

**Description**

Multicategory Vertex Discriminant Analysis (VDA) for classifying an outcome with \( k \) possible categories and \( p \) features, based on a data set of \( n \) cases. The default penalty function is Ridge. Lasso, Euclidean, and a mixture of Lasso and Euclidean are also available. Please refer to `vda.le`

**Usage**

```r
vda.r(x, y, lambda)
vda(x, y, lambda)
```

**Arguments**

- **x**  
  \( n \times p \) matrix or data frame containing the cases for each feature. The rows correspond to cases and the columns to the features. Intercept column is not included in this.

- **y**  
  \( n \times 1 \) vector representing the outcome variable. Each element denotes which one of the \( k \) classes that case belongs to.

- **lambda**  
  Tuning constant. The default value is set as \( 1/n \). Can also be found using `cv.vda.r`, which uses K-fold cross validation to determine the optimal value.

**Details**

Outcome classification is based on linear discrimination among the vertices of a regular simplex in a \( k-1 \)-dimension Euclidean space, where each vertex represents one of the categories. Discrimination is phrased as a regression problem involving \( \epsilon \)-insensitive residuals and a L2 quadratic ("ridge") penalty on the coefficients of the linear predictors. The objective function can by minimized by a primal Majorization-Minimization (MM) algorithm that

1. relies on quadratic majorization and iteratively re-weighted least squares,
2. is simpler to program than algorithms that pass to the dual of the original optimization problem, and
3. can be accelerated by step doubling.
Comparisons on real and simulated data suggest that the MM algorithm for VDA is competitive in statistical accuracy and computational speed with the best currently available algorithms for discriminant analysis, such as linear discriminant analysis (LDA), quadratic discriminant analysis (QDA), k-nearest neighbor, one-vs-rest binary support vector machines, multicategory support vector machines, classification and regression tree (CART), and random forest prediction.

Value

- **feature**: Feature matrix $x$ with an intercept vector added as the first column. All entries in the first column should equal 1.
- **stand.feature**: The feature matrix where the all columns are standardized, with the exception of the intercept column which is left unstandardized.
- **class**: Class vector $y$. All elements should be integers between 1 and $k$.
- **cases**: Number of cases, $n$.
- **classes**: Number of classes, $k$.
- **features**: Number of features, $p$.
- **lambda**: Tuning constant lambda that was used during analysis.
- **predicted**: Vector of predicted category values based on VDA.
- **coefficient**: The estimated coefficient matrix where the columns represent the coefficients for each predictor variable corresponding to $k-1$ outcome categories. The coefficient matrix is used for classifying new cases.
- **training_error_rate**: The percentage of instances in the training set where the predicted outcome category is not equal to the case’s true category.
- **call**: The matched call
- **attr(,"class")**: The function results in an object of class “vda.r”

Author(s)

Edward Grant, Xia Li, Kenneth Lange, Tong Tong Wu

Maintainer: Edward Grant <edward.m.grant@gmail.com>

References


See Also

For determining the optimal values for lambda, refer to **cv.vda.r**.

For high-dimensional setting and conduct variable selection, please refer to **vda.1e**.
Examples

```r
#load dataset from package
data(zoo)

#matrix containing all predictor vectors
x <- zoo[,2:17]

#outcome class vector
y <- zoo[,18]

#run VDA
out <- vda.r(x, y)
```

---

zoo data

Description

The zoo data is a set from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/). This dataset contains 16 attributes, and 7 animal classes. The first column gives a descriptive name for each case. The next 16 columns each correspond to one feature. The last column is the classification information. The classification is a breakdown of which animals are in which of the 7 types.

Usage

```r
data(zoo)
```

Format

- [.1] name: unique for each case
- [.2] hair: Boolean
- [.3] feathers: Boolean
- [.4] eggs: Boolean
- [.5] milk: Boolean
- [.6] airborne: Boolean
- [.7] aquatic: Boolean
- [.8] predator: Boolean
- [.9] toothed: Boolean
- [.10] backbone: Boolean
- [.11] breathes: Boolean
- [.12] venomous: Boolean
- [.13] fins: Boolean
- [.14] legs. Set of values: [0,2,4,5,6,8]
- [.15] tail: Boolean
- [.16] domestic: Boolean
- [.17] catsize: Boolean
- [.18] Class labels, integer values in range [1,7].
Details

There are 7 classes all together:

1. aardvark, antelope, bear, boar, buffalo, calf, cavy, cheetah, deer, dolphin, elephant, fruitbat, giraffe, girl, goat, gorilla, hamster, hare, leopard, lion, lynx, mole, mongoose, opossum, platypus, polecat, pony, porpoise, puma, pussycat, raccoon, reindeer, seal, sealion, squirrel, vampire, vole, wallaby, wolf
2. chicken, crow, dove, duck, flamingo, gull, hawk, kiwi, lark, ostrich, parakeet, penguin, pheasant, rhea, skimmer, skua, sparrow, swan, vulture, wren
3. pitviper, seasnake, slowworm, tortoise, tuatara
4. bass, carp, catfish, chub, dogfish, haddock, herring, pike, piranha, seahorse, sole, stingray, tuna
5. frog1, frog2, newt, toad
6. flea, gnat, honeybee, housefly, ladybird, moth, termite, wasp
7. clam, crab, crayfish, lobster, octopus, scorpion, seawasp, slug, starfish, worm

Source

Index

*Topic package
  VDA, 15

  cv.vda (cv.vda.r), 4
  cv.vda.le, 2, 5, 6, 16, 17
  cv.vda.r, 4, 7, 19

  plot.cv.vda (plot.cv.vda.r), 7
  plot.cv.vda.le, 2, 3, 5
  plot.cv.vda.r, 4, 5, 7
  predict.vda (predict.vda.r), 9
  predict.vda.le, 8
  predict.vda.r, 9
  print.vda (print.vda.r), 11
  print.vda.le, 8, 10, 13
  print.vda.r, 10, 11, 14

  summary.vda (summary.vda.r), 14
  summary.vda.le, 8, 11, 13
  summary.vda.r, 10, 12, 14

VDA, 15
vda (vda.r), 18
VDA-package (VDA), 15
vda.le, 3, 6, 8, 11, 13, 16, 18, 19
vda.r, 5, 7, 9, 10, 12, 14, 17, 18

zoo, 20