Package 'DWD'

February 14, 2012

Type Package

Title DWD implementation based on A IPM SOCP solver

Version 0.10

Date 2011-08-02

Author Hanwen Huang, Perry Haaland, Xiaosun Lu, Yufeng Liu, J. S. Marron

Maintainer Hanwen Huang <hanwenh@email.unc.edu>

Description This package provides the implementation of distance weighted discrimination (DWD) using an interior point method for the solution of second order cone programming problems,originally described by K.C. Toh, M.J. Todd, R.H. Tutuncu (1999).

Depends R (>= 2.10), Matrix, methods

License GPL-2

LazyLoad yes

Repository CRAN

Repository/R-Forge/Project dwd

Repository/R-Forge/Revision 30

Date/Publication 2011-11-06 08:39:38

R topics documented:

---

kdwd *Distance Weighted Discrimination*

Description

Distance Weighted Discriminations are an excellent tool for classification. kdwd supports the well known binary classification formulations along with native multi-class classification formulations.

Usage

```
## S4 method for signature 'formula'
kdwd(x, data = NULL, ..., subset,na.action = na.omit,
scaled = TRUE)
## S4 method for signature 'matrix'
kdwd(x, y = NULL, scaled = TRUE, type = "bdwd",
C = 100, fit = TRUE, cross = 0, class.weights = NULL,
     subset, na.action = na.omit)
```

Arguments

x	a symbolic description of the model to be fit. When not using a formula x can be a matrix containing the training data.
data	an optional data frame containing the training data, when using a formula.
y	a response vector with one label for each row/component of x. Can be a factor.
scaled	A logical vector indicating the variables to be scaled. If scaled is of length 1, the value is recycled as many times as needed and all non-binary variables are scaled.
type	kdwd can be used for binary and multiclass classification Depending on whether y is a level two factor or not, the default setting for type is bdwd, but can be overwritten by setting an explicit value. Valid options are: • bdwd Binary classification or one-vs-one for multi-class classification • mdwd Global multi-class classification
C	cost of constraints violation (default: 1) this is the 'C'-constant of the regularization term in the Lagrange formulation.
class.weights	a named vector of weights for the different classes, used for asymmetric class sizes. Not all factor levels have to be supplied (default weight: 1). All components have to be named.

cross	if a integer value k>0 is specified, a k-fold cross validation on the training data is performed to assess the quality of the model: the accuracy rate for classification.
fit	indicates whether the fitted values should be computed and included in the model or not (default: TRUE).
subset	An index vector specifying the cases to be used in the training sample. (NOTE: If given, this argument must be named.)
na.action	A function to specify the action to be taken if NAs are found. The default action is na.omit, which leads to rejection of cases with missing values on any required variable. An alternative is na.fail, which causes an error if NA cases are found. (NOTE: If given, this argument must be named.)
...	further arguments.

Details

kdwd uses SDPT3 infeasible path-following algorithm to solve the DWD SOCP problem.
For multiclass-classification with k classes, $k > 2$, bdwd uses the 'one-against-one'-approach, in which $k(k-1)/2$ binary classifiers are trained; the appropriate class is found by a voting scheme, The mdwd formulations deal with the multiclass-classification problems by solving a single SOCP problem involving all the classes.
In classification when cross is k a k-fold cross validation is performed on the data. The data can be passed to the kdwd function in a matrix or a data.frame.

Value

An S4 object of class "kdwd" containing the fitted model, Accessor functions can be used to access the slots of the object (see examples) which include:

w	The direction vector pointing towards positive class, unit vector (i.e length 1). In case of one-against-one classification this is a list of vectors. In case of global multiclass classification, it is a matrix.
b0	The intercept. In case of one-against-one classification this is a list of scalers. In case of global multiclass classification, it is a vector.
obj	The value of the objective function. In case of one-against-one classification this is a vector of values.
error	Training error.
cross	Cross validation error, (when cross > 0).
fitted	Fitted class label for training data.

Author(s)

Hanwen Huang: <hanwenh@email.unc.edu>; Perry Haaland: <Perry_Haaland@bd.com>; Xiaosun Lu: <Xiaosun_Lu@bd.com>; Yufeng Liu: <yfliu@email.unc.edu>; J. S. Marron: <marron@email.unc.edu>

References

- Kim-Chuan Toh , Michael J. Todd, and Reha H. Tutuncu
SDPT3 version 4.0 – a MATLAB software for semidefinite-quadratic-linear programming
http://www.math.nus.edu.sg/~mattohkc/sdpt3.html
- J. S. Marron and Michael Todd
Distance Weighted Discrimination http://ecommons.cornell.edu/bitstream/1813/9217/1/TR001339.pdf

See Also

predict.kdwd, kdwd-class

Examples

```
    ## simple binary classification example
data(promotergene)

## train DWD
gene <- kdwd(Class~.,data=promotergene,C=100,scaled=TRUE,cross=5)

gene@fitted

## simple multiclass example using the famous iris data
data(iris)

## train an OVO multiclass DWD
irismodel <- kdwd(Species~.,data=iris,type="bdwd",C=100,scaled=TRUE,cross=5)

## get fitted values
irismodel@fitted

## Test on the training set
predict(irismodel, iris)
```

---

kdwd-class *Class "kdwd"*

Description

An S4 class containing the output (model) of the kdwd Distance Weighted Discrimination function

Objects from the Class

Objects can be created by calls of the form `new("kdwd", ...)` or by calls to the kdwd function.

Slots

`type`: Object of class `"character"` containing the DWD type ("bdwd", "mdwd")

`kcall`: Object of class `"ANY"` containing the `kdwd` function call

`scaling`: Object of class `"ANY"` containing the scaling information performed on the data

`terms`: Object of class `"ANY"` containing the terms representation of the symbolic model used (when using a formula)

`fitted`: Object of class `"output"` with the fitted values, predictions using the training set.

`lev`: Object of class `"vector"` with the levels of the response (in the case of classification)

`nclass`: Object of class `"numeric"` containing the number of classes (in the case of classification)

`w`: Object of class `"ANY"` containing the resulting coefficients

`b0`: Object of class `"numeric"` containing the resulting offset

`index`: Object of class `"list"` containing the indexes of classifiers

`obj`: Object of class `vector` containing the value of the objective function. When using one-against-one in multiclass classification this is a vector.

`error`: Object of class `"numeric"` containing the training error

`cross`: Object of class `"numeric"` containing the cross-validation error

`na.action`: Object of class `"ANY"` containing the action performed for NA

Author(s)

Hanwen Huang: <hanwenh@email.unc.edu>; Perry Haaland: <Perry_Haaland@bd.com>; Xiaosun Lu: <Xiaosun_Lu@bd.com>; Yufeng Liu: <yfliu@email.unc.edu>; J. S. Marron: <marron@email.unc.edu>

See Also

kdwd

Examples

```
## simple example using the promotergene data set
data(promotergene)

## train a support vector machine
gene <- kdwd(Class~.,data=promotergene,C=100,cross=4)

# the fitted values
gene@fitted
# the cross validation error
gene@cross
```

musk *Musk data set*

Description

This dataset describes a set of 92 molecules of which 47 are judged by human experts to be musks and the remaining 45 molecules are judged to be non-musks.

Usage

```
data(musk)
```

Format

A data frame with 476 observations on the following 167 variables.

Variables 1-162 are "distance features" along rays. The distances are measured in hundredths of Angstroms. The distances may be negative or positive, since they are actually measured relative to an origin placed along each ray. The origin was defined by a "consensus musk" surface that is no longer used. Hence, any experiments with the data should treat these feature values as lying on an arbitrary continuous scale. In particular, the algorithm should not make any use of the zero point or the sign of each feature value.

Variable 163 is the distance of the oxygen atom in the molecule to a designated point in 3-space. This is also called OXY-DIS.

Variable 164 is the X-displacement from the designated point.

Variable 165 is the Y-displacement from the designated point.

Variable 166 is the Z-displacement from the designated point.

Class: 0 for non-musk, and 1 for musk

Source

UCI Machine Learning data repository

Examples

```
data(musk)

muskm <- kdwd(Class~.,data=musk,C=100)
```

predict.kdwd *predict method for DWD object*

Description

Prediction of test data using distance weighted discrimination

Usage

```
## S4 method for signature 'kdwd'
predict(object, newdata)
```

Arguments

object	an S4 object of class kdwd created by the kdwd function
newdata	a data frame or matrix containing new data

Value

predicted classes.

Author(s)

Hanwen Huang: <hanwenh@email.unc.edu>; Perry Haaland: <Perry_Haaland@bd.com>; Xiaosun Lu: <Xiaosun_Lu@bd.com>; Frances Tong: <Frances_Tong@bd.com>; Elaine McVey: <Elaine_McVey@bd.com>; Yufeng Liu: <yfliu@email.unc.edu>; J. S. Marron: <marron@email.unc.edu>

Examples

```
## example using the promotergene data set
data(promotergene)

## create test and training set
ind <- sample(1:dim(promotergene)[1],20)
genetrain <- promotergene[-ind, ]
genetest <- promotergene[ind, ]

## train a distance weighted discrimination
gene <- kdwd(Class~.,data=genetrain,C=100,cross=5,scaled=TRUE)
gene@fitted

## predict gene type on the test set
genetype <- predict(gene,genetest)
genetype
```

promotergene *E. coli promoter gene sequences (DNA)*

Description

Promoters have a region where a protein (RNA polymerase) must make contact and the helical DNA sequence must have a valid conformation so that the two pieces of the contact region spatially align. The data contains DNA sequences of promoters and non-promoters.

Usage

```
data(promotergene)
```

Format

A data frame with 106 observations and 58 variables. The first variable `Class` is a factor with levels `+` for a promoter gene and `-` for a non-promoter gene. The remaining 57 variables `V2 to V58` are factors describing the sequence. The DNA bases are coded as follows: `a` adenine `c` cytosine `g` guanine `t` thymine

Source

UCI Machine Learning data repository
ftp://ftp.ics.uci.edu/pub/machine-learning-databases/molecular-biology/promoter-gene-sequences

References

Towell, G., Shavlik, J. and Noordewier, M.
Refinement of Approximate Domain Theories by Knowledge-Based Artificial Neural Networks.
In Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)

Examples

```
data(promotergene)

## Create model using Distance Weighted Discrimination

promsv <- kdwd(Class~.,data=promotergene,C=100,cross=4)
```

reuters *Reuters Text Data*

Description

A small sample from the Reuters news data set.

Usage

```
data(reuters)
```

Format

A list of 40 text documents along with the labels. `reuters` contains the text documents and `rlabels` the labels in a vector.

Details

This dataset contains a list of 40 text documents along with the labels. The data consist out of 20 documents from the `acq` category and 20 documents from the crude category. The labels are stored in `rlabels`

Source

Reuters

solve_QP_SOCP *Solve a Quadratic Programming Problem*

Description

This routine implements the second order cone programming method from Kim-Chuan Toh , Michael J. Todd, and Reha H. Tutuncu for solving quadratic programming problems of the form $\min(-d^T b + 1/2 b^T D b)$ with the constraints $A^T b >= b_0$.

Usage

```
solve_QP_SOCP(Dmat, dvec, Amat, bvec)
```

Arguments

`Dmat`	matrix appearing in the quadratic function to be minimized.
`dvec`	vector appearing in the quadratic function to be minimized.
`Amat`	matrix defining the constraints under which we want to minimize the quadratic function.
`bvec`	vector holding the values of b_0 (defaults to zero).

Value

a list with the following components:

`solution` vector containing the solution of the quadratic programming problem.

Author(s)

Hanwen Huang: <hanwenh@email.unc.edu>; Perry Haaland: <Perry_Haaland@bd.com>; Xiaosun Lu: <Xiaosun_Lu@bd.com>; Frances Tong: <Frances_Tong@bd.com>; Elaine McVey: <Elaine_McVey@bd.com>; Yufeng Liu: <yfliu@email.unc.edu>; J. S. Marron: <marron@email.unc.edu>

References

Kim-Chuan Toh , Michael J. Todd, and Reha H. Tutuncu
SDPT3 version 4.0 – a MATLAB software for semidefinite-quadratic-linear programming
http://www.math.nus.edu.sg/~mattohkc/sdpt3.html

See Also

sqlp

Examples

```
##
## Assume we want to minimize: -(0 5 0) %*% b + 1/2 b^T b
## under the constraints:      A^T b >= b0
## with b0 = (-8,2,0)^T
## and      (-4  2  0)
##      A = (-3  1 -2)
##          ( 0  0  1)
## we can use solve.QP as follows:
##
Dmat       <- matrix(0,3,3)
diag(Dmat) <- 1
dvec       <- c(0,5,0)
Amat       <- matrix(c(-4,-3,0,2,1,0,0,-2,1),3,3)
bvec       <- c(-8,2,0)
solve_QP_SOCP(Dmat,dvec,Amat,bvec=bvec)
```

---

spam *Spam E-mail Database*

Description

A data set collected at Hewlett-Packard Labs, that classifies 4601 e-mails as spam or non-spam. In addition to this class label there are 57 variables indicating the frequency of certain words and characters in the e-mail.

Usage

```
data(spam)
```

Format

A data frame with 4601 observations and 58 variables.

The first 48 variables contain the frequency of the variable name (e.g., business) in the e-mail. If the variable name starts with num (e.g., num650) the it indicates the frequency of the corresponding number (e.g., 650). The variables 49-54 indicate the frequency of the characters ';', '(', '[', '!', '\$', and '\#'. The variables 55-57 contain the average, longest and total run-length of capital letters. Variable 58 indicates the type of the mail and is either `"nonspam"` or `"spam"`, i.e. unsolicited commercial e-mail.

Details

The data set contains 2788 e-mails classified as `"nonspam"` and 1813 classified as `"spam"`.

The "spam" concept is diverse: advertisements for products/web sites, make money fast schemes, chain letters, pornography... This collection of spam e-mails came from the collectors' postmaster and individuals who had filed spam. The collection of non-spam e-mails came from filed work and personal e-mails, and hence the word 'george' and the area code '650' are indicators of non-spam. These are useful when constructing a personalized spam filter. One would either have to blind such non-spam indicators or get a very wide collection of non-spam to generate a general purpose spam filter.

Source

- Creators: Mark Hopkins, Erik Reeber, George Forman, Jaap Suermondt at Hewlett-Packard Labs, 1501 Page Mill Rd., Palo Alto, CA 94304
- Donor: George Forman (gforman at nospam hpl.hp.com) 650-857-7835

These data have been taken from the UCI Repository Of Machine Learning Databases at http://www.ics.uci.edu/~mlearn/MLRepository.html

References

T. Hastie, R. Tibshirani, J.H. Friedman. *The Elements of Statistical Learning.* Springer, 2001.

`spirals` *Spirals Dataset*

Description

A toy data set representing two spirals with Gaussian noise. The data was created with the `mlbench.spirals` function in `mlbench`.

Usage

```
data(spirals)
```

Format

A matrix with 300 observations and 2 variables.

Examples

```
data(spirals)
plot(spirals)
```

`sqlp` *Solve Second Order Cone Programs*

Description

solve an SOCP program by infeasible path-following method

Usage

```
sqlp(blk,At,C,b,OPTIONS,X0,y0,Z0)
```

Arguments

`blk`	a list describing the structure of the SOCP data.
`At`	a matrix containing the coefficients for the linear and second order cone constraints. At should have m columns, where m is the number of constraints. The number of rows in At should be `sum(C)`.
`C`	a vector containing the coefficients of the objective function to be minimized.
`b`	a vector containing the right hand side of the constraints.
`OPTIONS`	a list that specifies parameters required in sqlp.
`X0`	an initial iterate for primal solution.
`y0,Z0`	initial iterates for dual solution.

Details

A second order cone program (SOCP) is an optimization problem similar to a linear program (LP), except that some variables can be constrained by second order cones. An exact mathematical definition can be found in Kim-Chuan Toh , Michael J. Todd, and Reha H. Tutuncu. This function implements the algorithm given in that paper which allows for multiple second order cone constraints as well as linear constraints. The objective function is given by `sum(C*x)` while the constraints are `A%*%x == b`, with `x` belonging to the cartesian product of second order cones described by `blk`.

Value

A list containing named elements:

`x`	The optimal solution to the SOCP.
`y,Z`	The dual solutions.
`info`	Summary information.
`runhist`	Run history.

Author(s)

Hanwen Huang: <hanwenh@email.unc.edu>; Perry Haaland: <Perry_Haaland@bd.com>; Xiaosun Lu: <Xiaosun_Lu@bd.com>; Frances Tong: <Frances_Tong@bd.com>; Elaine McVey: <Elaine_McVey@bd.com>; Yufeng Liu: <yfliu@email.unc.edu>; J. S. Marron: <marron@email.unc.edu>

References

Kim-Chuan Toh , Michael J. Todd, and Reha H. Tutuncu
SDPT3 version 4.0 – a MATLAB software for semidefinite-quadratic-linear programming
http://www.math.nus.edu.sg/~mattohkc/sdpt3.html

Examples

```
#Load an example SOCP
data(sqlpData)

#Solve the socp
soln <- sqlp(blk=sqlpData$blk,At=sqlpData$At,C=sqlpData$C,b=sqlpData$b,X0=sqlpData$X0,y0=sqlpData$y0,Z0=sqlpDat
```

---

sqlpData *SOCP input Data*

Description

A simulated example which can be directed input into `sqlp`.

Usage

```
data(sqlpData)
```

Format

A list of 7 entries which serve as the direct input arguments for `sqlp`.

Details

This dataset contains a list of 7 entries:

- blka list describing the structure of the SOCP data.
- Ata matrix containing the coefficients for the linear and second order cone constraints. At should have m columns, where m is the number of constraints. The number of rows in At should be `sum(C)`.
- Ca vector containing the coefficients of the objective function to be minimized.
- ba vector containing the right hand side of the constraints.
- X0an initial iterate for primal solution.
- y0,Z0initial iterates for dual solution.

ticdata *The Insurance Company Data*

Description

This data set used in the CoIL 2000 Challenge contains information on customers of an insurance company. The data consists of 86 variables and includes product usage data and socio-demographic data derived from zip area codes. The data was collected to answer the following question: Can you predict who would be interested in buying a caravan insurance policy and give an explanation why ?

Usage

```
data(ticdata)
```

Format

ticdata: Dataset to train and validate prediction models and build a description (9822 customer records). Each record consists of 86 attributes, containing sociodemographic data (attribute 1-43) and product ownership (attributes 44-86). The sociodemographic data is derived from zip codes. All customers living in areas with the same zip code have the same sociodemographic attributes. Attribute 86, `CARAVAN:Number of mobile home policies`, is the target variable.

Data Format

1	STYPE	Customer Subtype
2	MAANTHUI	Number of houses 1 - 10
3	MGEMOMV	Avg size household 1 - 6
4	MGEMLEEF	Average age
5	MOSHOOFD	Customer main type
6	MGODRK	Roman catholic
7	MGODPR	Protestant ...
8	MGODOV	Other religion
9	MGODGE	No religion
10	MRELGE	Married
11	MRELSA	Living together
12	MRELOV	Other relation
13	MFALLEEN	Singles
14	MFGEKIND	Household without children
15	MFWEKIND	Household with children
16	MOPLHOOG	High level education
17	MOPLMIDD	Medium level education
18	MOPLLAAG	Lower level education
19	MBERHOOG	High status
20	MBERZELF	Entrepreneur
21	MBERBOER	Farmer
22	MBERMIDD	Middle management
23	MBERARBG	Skilled labourers

24	MBERARBO	Unskilled labourers
25	MSKA	Social class A
26	MSKB1	Social class B1
27	MSKB2	Social class B2
28	MSKC	Social class C
29	MSKD	Social class D
30	MHHUUR	Rented house
31	MHKOOP	Home owners
32	MAUT1	1 car
33	MAUT2	2 cars
34	MAUT0	No car
35	MZFONDS	National Health Service
36	MZPART	Private health insurance
37	MINKM30	Income >30.000
38	MINK3045	Income 30-45.000
39	MINK4575	Income 45-75.000
40	MINK7512	Income 75-122.000
41	MINK123M	Income <123.000
42	MINKGEM	Average income
43	MKOOPKLA	Purchasing power class
44	PWAPART	Contribution private third party insurance
45	PWABEDR	Contribution third party insurance (firms)
46	PWALAND	Contribution third party insurance (agriculture)
47	PPERSAUT	Contribution car policies
48	PBESAUT	Contribution delivery van policies
49	PMOTSCO	Contribution motorcycle/scooter policies
50	PVRAAUT	Contribution lorry policies
51	PAANHANG	Contribution trailer policies
52	PTRACTOR	Contribution tractor policies
53	PWERKT	Contribution agricultural machines policies
54	PBROM	Contribution moped policies
55	PLEVEN	Contribution life insurances
56	PPERSONG	Contribution private accident insurance policies
57	PGEZONG	Contribution family accidents insurance policies
58	PWAOREG	Contribution disability insurance policies
59	PBRAND	Contribution fire policies
60	PZEILPL	Contribution surfboard policies
61	PPLEZIER	Contribution boat policies
62	PFIETS	Contribution bicycle policies
63	PINBOED	Contribution property insurance policies
64	PBYSTAND	Contribution social security insurance policies
65	AWAPART	Number of private third party insurance 1 - 12
66	AWABEDR	Number of third party insurance (firms) ...
67	AWALAND	Number of third party insurance (agriculture)
68	APERSAUT	Number of car policies
69	ABESAUT	Number of delivery van policies
70	AMOTSCO	Number of motorcycle/scooter policies
71	AVRAAUT	Number of lorry policies

72	AAANHANG	Number of trailer policies
73	ATRACTOR	Number of tractor policies
74	AWERKT	Number of agricultural machines policies
75	ABROM	Number of moped policies
76	ALEVEN	Number of life insurances
77	APERSONG	Number of private accident insurance policies
78	AGEZONG	Number of family accidents insurance policies
79	AWAOREG	Number of disability insurance policies
80	ABRAND	Number of fire policies
81	AZEILPL	Number of surfboard policies
82	APLEZIER	Number of boat policies
83	AFIETS	Number of bicycle policies
84	AINBOED	Number of property insurance policies
85	ABYSTAND	Number of social security insurance policies
86	CARAVAN	Number of mobile home policies 0 - 1

Note: All the variables starting with M are zipcode variables. They give information on the distribution of that variable, e.g., Rented house, in the zipcode area of the customer.

Details

Information about the insurance company customers consists of 86 variables and includes product usage data and socio-demographic data derived from zip area codes. The data was supplied by the Dutch data mining company Sentient Machine Research and is based on a real world business problem. The training set contains over 5000 descriptions of customers, including the information of whether or not they have a caravan insurance policy. The test set contains 4000 customers. The test and data set are merged in the ticdata set. More information about the data set and the CoIL 2000 Challenge along with publications based on the data set can be found at http://www.liacs.nl/~putten/library/cc2000/.

Source

- UCI KDD Archive:http://kdd.ics.uci.edu
- Donor: Sentient Machine Research
  Peter van der Putten
  Sentient Machine Research
  Baarsjesweg 224
  1058 AA Amsterdam
  The Netherlands
  +31 20 6186927
  pvdputten@hotmail.com, putten@liacs.nl

References

Peter van der Putten, Michel de Ruiter, Maarten van Someren *CoIL Challenge 2000 Tasks and Results: Predicting and Explaining Caravan Policy Ownership*
http://www.liacs.nl/~putten/library/cc2000/

Index