Package ‘protoclass’

February 20, 2015

Maintainer  Jacob Bien <jbien@cornell.edu>
Author       Jacob Bien and Robert Tibshirani
Version      1.0
License      GPL-3
Title        Interpretable classification with prototypes
Description  Greedy algorithm described in Bien and Tibshirani (2011)
              Prototype Selection for Interpretable Classification. Annals of
              Applied Statistics. 5(4). 2403-2424
Depends      class
Suggests    testthat, roxygen2
Collate      'protoclass.r' 'protoclass-package.r'
NeedsCompilation no
Repository   CRAN
Date/Publication  2013-05-31 11:11:41

R topics documented:

  dist2  ................................................................. 2
  plot.protoclass ...................................................... 2
  predict.protoclass .................................................. 3
  predictwithd.protoclass ............................................ 3
  print.protoclass ..................................................... 4
  protoclass .......................................................... 4
Index         7
---

### **dist2**

*Compute distances between two sets of points*

**Description**

Given \((n1)\)-by-\(p\) and \((n2)\)-by-\(p\) matrices, computes \((n1)\)-by-(\(n2\)) matrix of Euclidean distances.

**Usage**

```r
dist2(x, xx)
```

**Arguments**

- **x**: a \((n1)\)-by-\(p\) matrix
- **xx**: a \((n2)\)-by-\(p\) matrix

---

### **plot.prototype**

*Plots prototypes with data*

**Description**

This function is only for two-dimensional data.

**Usage**

```r
## S3 method for class 'protoclass'
plot(x, xtr, y, z = NULL, 
circleFLAG = TRUE, boundFLAG = FALSE, res = NULL, 
main = NULL, ...)
```

**Arguments**

- **x**: output of protoclass function
- **xtr**: \(n\) by \(p\) matrix of training features.
- **y**: \(n\)-vector of labels of the training data.
- **z**: set of potential prototypes (only required if protoclass was not run with \(z=x\)).
- **circleFLAG**: indicates whether to draw circles around each prototype.
- **boundFLAG**: indicates whether to draw classification boundary. This can be computationally expensive.
- **res**: resolution of the classification boundary. Computation increases with res. Try 100.
- **main**: optional main title for plot.
- **...**: other arguments to pass to plot.
**predict.protoclass**

#### Make predictions based on prototypes

**Description**

Performs nearest-prototype classification on newx.

**Usage**

```r
## S3 method for class 'protoclass'
predict(object, newx, z, ...)
```

**Arguments**

- `object`: output of protoclass function
- `newx`: features from some new data that we want to make predictions on.
- `z`: potential prototype positions z (often taken to be x, the training data)
- `...`: ignore this argument.

**See Also**

- `protoclass`
- `predictwithd.protoclass`

---

**predictwithd.protoclass**

#### Make predictions based on prototypes

**Description**

Performs nearest-prototype classification like predict.protoclass, except only requires the ntest-by-m matrix of distances between test features and z rather than requiring these matrices themselves. (Using this saves repeated computation of the pairwise distances...)

**Usage**

```r
predictwithd.protoclass(object, d.test.z)
```

**Arguments**

- `object`: output of protoclass function
- `d.test.z`: ntest-by-m matrix of distances between test features and z
See Also

predict.protoclass

print.protoclass  

Prints an object of class protoclass

Description

Prints an object of class protoclass

Usage

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

Arguments

x  
object of class protoclass.

...  
ignore this argument.

protoclass  
Greedy algorithm for prototype selection

Description


Usage

protoclass(x, y, z, dxz, eps, lambda = 1/n)

Arguments

x  
n by p matrix of training features (optional, see dxz).

y  
n-vector of labels of the training data.

z  
set of potential prototypes (optional, see dxz).

dxz  
instead of x and z, you can give dxz, the matrix of pairwise dissimilarities between x and z, with ij-th element giving the dissimilarity between training point x_i and prototype-candidate z_j.

eps  
size of covering balls.

lambda  
cost of adding a prototype.
Details

It's more efficient to compute dxz just once on your own rather than have protoclass repeatedly compute the pairwise distances on each call.

Value

An object of class "protoclass," which has the following elements:

- **alpha**: Matrix of dimensions nrow(z)-by-nclass. alpha[j,k] indicates whether jth potential prototype has been chosen as a prototype for class k.
- **classes**: Names of classes
- **proto.order**: The sequence of prototypes that were selected.
- **ncovered**: nproto-by-nclass matrix with ncovered[j,k] giving the number of class k training points covered by the jth prototype's ball.
- **coverlist**: n-by-nclass matrix with row i giving number of each type of prototype covering point i.
- **uncovered**: Indicates whether a training point is not covered by a prototype of its own class.
- **wrongcover**: Number of prototypes from other classes covering each training point.
- **nproto**: nclass-vector giving the number of prototypes in each class.

See Also

* predict.protoclass*

Examples

```r
# generate some data:
set.seed(1)
n <- 200
p <- 2
x <- matrix(rnorm(n * p), n, p)
y <- rep(c("A", "B"), each=n/2)
x[y=="A", ] <- x[y=="A", ] + 3
itr <- sample(n, n/2)
xtr <- x[itr, ] # train
ytr <- y[itr]
xte <- x[-itr, ] # test
yte <- y[-itr]

# take prototype candidates identical to training points:
z <- xtr
dxz <- dist2(xtr, z)
# run protoclass:
prot <- protoclass(dxz=dxz, y=ytr, eps=2, lambda=1/n)
## Not run:
plot(prot, xtr, y=1+(ytr=="A"))

## End(Not run)
# get predictions on test data:
```
pred1 <- predict(prot, xte, z=xtr)
# get predictions on test data using pairwise distances:
pred2 <- predictwithd.protoclass(prot, dist2(xte, z))
Index

dist2, 2

plot.protoclass, 2
predict.protoclass, 3, 4, 5
predictwithd.protoclass, 3, 3
print.protoclass, 4
protoclass, 3, 4
protoclass-package (protoclass), 4