Package ‘prclust’

December 13, 2016

Type Package
Title Penalized Regression-Based Clustering Method
Version 1.3
Date 2016-12-12
Depends R (>= 3.1.1)
Author Chong Wu, Wei Pan
Maintainer Chong Wu <wuxx0845@umn.edu>
Description Clustering is unsupervised and exploratory in nature. Yet, it can be performed through penalized regression with grouping pursuit. In this package, we provide two algorithms for fitting the penalized regression-based clustering (PRclust) with non-convex grouping penalties, such as group truncated lasso, MCP and SCAD. One algorithm is based on quadratic penalty and difference convex method. Another algorithm is based on difference convex and ADMM, called DC-ADD, which is more efficient. Generalized cross validation and stability based method were provided to select the tuning parameters. Rand index, adjusted Rand index and Jaccard index were provided to estimate the agreement between estimated cluster memberships and the truth.
License GPL-2 | GPL-3
Imports Rcpp (>= 0.12.1), parallel
LinkingTo Rcpp
NeedsCompilation yes
Repository CRAN
Date/Publication 2016-12-13 07:57:15

R topics documented:

   prclust-package ......................................................... 2
   clusterStat .............................................................. 3
   GCV ................................................................. 4
   PRclust ............................................................. 5
   stability ............................................................. 7

Index 10
Description

Clustering analysis is widely used in many fields. Traditionally clustering is regarded as unsupervised learning for its lack of a class label or a quantitative response variable, which in contrast is present in supervised learning such as classification and regression. Here we formulate clustering as penalized regression with grouping pursuit. In addition to the novel use of a non-convex group penalty and its associated unique operating characteristics in the proposed clustering method, a main advantage of this formulation is its allowing borrowing some well established results in classification and regression, such as model selection criteria to select the number of clusters, a difficult problem in clustering analysis. In particular, we propose using the generalized cross-validation (GCV) based on generalized degrees of freedom (GDF) to select the number of clusters. we further develop this method by developing a more efficient algorithm for scalable computation as well as a new theory for PRclust. This algorithm, called DC-ADMM, combines difference of convex programming with the alternating direction method of multipliers (ADMM). This method is more efficient than the quadratic penalty algorithm used in Pan et al. (2013) due to the availability of closed-form updating formulas.

Details

Package: prclust
Type: Package
Version: 1.3
Date: 2016-12-12
License: GPL-2 | GPL-3

Author(s)

Chong Wu, Wei Pan
Maintainer: Chong Wu <wuxx0845@umn.edu>

References


Examples

## In default, we use DC-ADMM, a faster algorithm to solve
library("prclust")
# generate the data
data = matrix(NA,2,100)
data[1,1:50] = rnorm(50,0,0.33)
data[2,1:50] = rnorm(50,0,0.33)
data[1,51:100] = rnorm(50,1,0.33)
data[2,51:100] = rnorm(50,1,0.33)

# clustering via PRclust
a = PRclust(data,lambda1=0.4,lambda2=1,tau=0.5)
a$mu
a$group

cclusterStat

## the objective function and get the clustering result.

### Description

Suppose we know the true cluster results beforehand. `clusterStat` provides Rand, adjusted Rand, Jaccard index to measure the quality of a cluster results.

### Usage

`clusterStat(trueGroup, group)`

### Arguments

- `trueGroup`: The true cluster results.
- `group`: The estimated cluster results, not necessary calculating by PRclust.

### Value

The return value is a "clusterStat" class, providing the following information.

- Rand: Rand Index
- AdjustedRand: Adjusted Rand Index
- Jaccard: Jaccard Index

### Author(s)

Chong Wu

### Examples

```r
a <- rep(1:3,3)
a
b <- rep(c(4:6),3)
b
clusterStat(a,b)
```
Description

Calculate the generalized cross-validation statistic with generalized degrees of freedom.

Usage

GCV(data, lambda1, lambda2, tau, sigma, b = 100, loss.method = c("quadratic", "lasso"), grouping.penalty = c("gtlp", "L1", "SCAD", "MCP"), algorithm = c("ADMM", "Quadratic"), epsilon = 0.001)

Arguments

data Numeric data matrix.

lambda1 Tuning parameter or step size: lambda1, typically set at 1 for quadratic penalty based algorithm; 0.4 for revised ADMM.

lambda2 Tuning parameter: lambda2, the magnitude of grouping penalty.

tau Tuning parameter: tau, related to grouping penalty.

sigma The perturbation size.

b The Monte Carlo time. The default value is 100.

loss.method character may be abbreviated. "lasso" stands for $L_1$ loss function, while "quadratic" stands for the quadratic loss function.

grouping.penalty character: may be abbreviated. "gtlp" means generalized group lasso is used for grouping penalty. "lasso" means lasso is used for grouping penalty. "SCAD" and "MCP" are two other non-convex penalty.

algorithm character: may be abbreviated. The algorithm will use for finding the solution. The default algorithm is "ADMM", which stands for the DC-ADMM.

epsilon The stopping criterion parameter. The default is 0.001.

Details

A bonus with the regression approach to clustering is the potential application of many existing model selection methods for regression or supervised learning to clustering. We propose using generalized cross-validation (GCV). GCV can be regarded as an approximation to leave-one-out cross-validation (CV). Hence, GCV provides an approximately unbiased estimate of the prediction error.

We use the generalized degrees of freedom (GDF) to consider the data-adaptive nature in estimating the centroids of the observations.

The chosen tuning parameters are the one giving the smallest GCV error.
Value

Return value: the Generalized cross-validation statistic (GCV)

Author(s)

Chong Wu, Wei Pan

References


Examples

set.seed(1)
library("prclust")
data = matrix(NA,2,50)
data[1,1:25] = rnorm(25,0,0.33)
data[2,1:25] = rnorm(25,0,0.33)
data[1,26:50] = rnorm(25,1,0.33)
data[2,26:50] = rnorm(25,1,0.33)

#case 1
gcv1 = GCV(data,lambda1=1,lambda2=1,tau=0.5,sigma=0.25,B =10)
gcv1

#case 2
gcv2 = GCV(data,lambda1=1,lambda2=0.7,tau=0.3,sigma=0.25,B = 10)
gcv2

# Note that the combination of tuning parameters in case 1 are better than
# the combination of tuning parameters in case 2 since the value of GCV in case 1 is
# less than the value in case 2.

PRclust

Find the Solution of Penalized Regression-Based Clustering.

Description

Clustering is unsupervised and exploratory in nature. Yet, it can be performed through penalized regression with grouping pursuit. Prclust helps us perform penalized regression-based clustering with various loss functions and grouping penalties via two algorithms (DC-ADMM and quadratic penalty).

Usage

PRclust(data, lambda1, lambda2, tau, 
loss.method = c("quadratic","lasso"),
grouping.penalty = c("gtlp","L1","SCAD","MCP"),
algorithm = c("ADMM","Quadratic"), epsilon=0.001)
**Arguments**

- **data**
  - input matrix, of dimension nvars x nob; each column is an observation vector.

- **lambda1**
  - Tuning parameter or step size: lambda1, typically set at 1 for quadratic penalty based algorithm; 0.4 for revised ADMM.

- **lambda2**
  - Tuning parameter: lambda2, the magnitude of grouping penalty.

- **tau**
  - Tuning parameter: tau, related to grouping penalty.

- **loss.method**
  - The loss method. "lasso" stands for $L_1$ loss function, while "quadratic" stands for the quadratic loss function.

- **grouping.penalty**
  - Grouping penalty. Character: may be abbreviated. "gtp" means generalized group lasso is used for grouping penalty. "lasso" means lasso is used for grouping penalty. "SCAD" and "MCP" are two other non-convex penalty.

- **algorithm**
  - Character: may be abbreviated. The algorithm to use for finding the solution. The default algorithm is "ADMM", which stands for the new algorithm we developed.

- **epsilon**
  - The stopping criterion parameter. The default is 0.001.

**Details**

Clustering analysis has been widely used in many fields. In the absence of a class label, clustering analysis is also called unsupervised learning. However, penalized regression-based clustering adopts a novel framework for clustering analysis by viewing it as a regression problem. In this method, a novel non-convex penalty for grouping pursuit was proposed which data-adaptively encourages the equality among some unknown subsets of parameter estimates. This new method can deal with some complex clustering situation, for example, in the presence of non-convex cluster, in which the K-means fails to work, PRclust might perform much better.

**Value**

The return value is a list. In this list, it contains the following matrix.

- **mu**
  - The centroid of the each observations.

- **theta**
  - The theta value for the data set, not very useful.

- **group**
  - The group for each points.

- **count**
  - The iteration times.

**Note**

Choosing tuning parameter is kind of time consuming job. It is always based on "trials and errors".

**Author(s)**

Chong Wu, Wei Pan
References


Examples

```r
library("prclust")

# To let you have a better understanding about the power and strength
# of PRclust method, 6 examples in original prclust paper were provided.

### case 1

# generate the data
data = matrix(NA,2,100)
data[1,1:50] = rnorm(50,0,0.33)
data[2,1:50] = rnorm(50,0,0.33)
data[1,51:100] = rnorm(50,1,0.33)
data[2,51:100] = rnorm(50,1,0.33)

# set the tuning parameter
lambda1 = 1
lambda2 = 3
tau = 0.5
a =PRclust(data,lambda1,lambda2,tau)
a
```

---

**stability**

*Calculate the stability based statistics*

**Description**

Calculate the the stability based statistics. We try with various tuning parameter values, obtaining their corresponding stability based statistics average prediction strengths, then choose the set of the tuning parameters with the maximum average prediction strength.

**Usage**

```r
stability(data,rho,lambda,tau,
  loss.function = c("quadratic","L1","MCP","SCAD"),
  grouping.penalty = c("gtlp","tlp"),
  algorithm = c("DCADMM","Quadratic"),
  epsilon = 0.001,n.times = 10)
```
### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>Input matrix. Each column is an observation vector.</td>
</tr>
<tr>
<td>rho</td>
<td>Tuning parameter or step size: rho, typically set at 1 for quadratic penalty based algorithm; 0.4 for DC-ADMM. (Note that rho is the lambda1 in quadratic penalty based algorithm.)</td>
</tr>
<tr>
<td>lambda</td>
<td>Tuning parameter: lambda, the magnitude of grouping penalty.</td>
</tr>
<tr>
<td>tau</td>
<td>Tuning parameter: tau, a nonnegative tuning parameter controlling the trade-off between the model fit and the number of clusters.</td>
</tr>
<tr>
<td>loss.function</td>
<td>The loss function. &quot;L1&quot; stands for $L_1$ loss function, while &quot;quadratic&quot; stands for the quadratic loss function.</td>
</tr>
<tr>
<td>grouping.penalty</td>
<td>Grouping penalty. Character: may be abbreviated. &quot;gtlp&quot; means generalized group lasso is used for grouping penalty. &quot;lasso&quot; means lasso is used for grouping penalty. &quot;SCAD&quot; and &quot;MCP&quot; are two other non-convex penalty.</td>
</tr>
<tr>
<td>algorithm</td>
<td>Two algorithms for PRclust. &quot;DC-ADMM&quot; and &quot;Quadratic&quot; stand for the DC-ADMM and quadratic penalty based criterion respectively. &quot;DC-ADMM&quot; is much faster than &quot;Quadratic&quot; and thus recommend it here.</td>
</tr>
<tr>
<td>epsilon</td>
<td>The stopping criterion parameter corresponding to DC-ADMM. The default is 0.001.</td>
</tr>
<tr>
<td>n.times</td>
<td>Repeat times. Based on our limited simulations, we find 10 is usually good enough.</td>
</tr>
</tbody>
</table>

### Details

A generalized degrees of freedom (GDF) together with generalized cross validation (GCV) was proposed for selection of tuning parameters for clustering (Pan et al., 2013). This method, while yielding good performance, requires extensive computation and specification of a hyper-parameter perturbation size. Here, we provide an alternative by modifying a stability-based criterion (Tibshirani and Walther, 2005; Liu et al., 2016) for determining the tuning parameters.

The main idea of the method is based on cross-validation. That is, (1) randomly partition the entire data set into a training set and a test set with an almost equal size; (2) cluster the training and test sets separately via PRclust with the same tuning parameters; (3) measure how well the training set clusters predict the test clusters.

We try with various tuning parameter values, obtaining their corresponding stability based statistics average prediction strengths, then choose the set of the tuning parameters with the maximum average prediction strength.

### Value

Return value: the average prediction score.

### Author(s)

Chong Wu
References


Examples

```r
set.seed(1)
library("prclust")
data = matrix(NA,2,50)
data[1,1:25] = rnorm(25,0,0.33)
data[2,1:25] = rnorm(25,0,0.33)
data[1,26:50] = rnorm(25,1,0.33)
data[2,26:50] = rnorm(25,1,0.33)

#case 1
stab1 = stability(data,rho=1,lambda=1,tau=0.5,n.times = 2)
stab1

#case 2
stab2 = stability(data,rho=1,lambda=0.7,tau=0.3,n.times = 2)
stab2
# Note that the combination of tuning parameters in case 1 are better than
# the combination of tuning parameters in case 2 since the value of GCV in case 1 is
# less than the value in case 2.
```
Index

*Topic GCV
  GCV, 4
*Topic PRclus
  PRclus, 5
*Topic clusterStat
  clusterStat, 3
*Topic prclust-package
  prclust-package, 2
*Topic stability
  stability, 7

clusterStat, 3

GCV, 4

PRclus, 5
prclust (prclust-package), 2
prclust-package, 2

stability, 7