Package ‘mvcluster’

April 3, 2016

Type Package

Title Multi-View Clustering

Version 1.0

Date 2016-04-01

Author Jiangwen Sun, Jin Lu, Tingyang Xu, Joseph Muller, Jinbo Bi

Maintainer Jiangwen Sun <javon@engr.uconn.edu>

Description Implementation of multi-view bi-clustering algorithms. When a sample is characterized by two or more sets of input features, it creates multiple data matrices for the same set of examples, each corresponding to a view. For instance, individuals who are diagnosed with a disorder can be described by their clinical symptoms (one view) and their genomic markers (another view). Rows of a data matrix correspond to examples and columns correspond to features. A multi-view bi-clustering algorithm groups examples (rows) consistently across the views and simultaneously identifies the subset of features (columns) in each view that are associated with the row groups. This mvcluster package includes three such methods. (1) MVSVDL1: multi-view bi-clustering based on singular value decomposition where the left singular vectors are used to identify row clusters and the right singular vectors are used to identify features (columns) for each row cluster. Each singular vector is regularized by the L1 vector norm. (2) MVLRRL0: multi-view bi-clustering based on sparse low rank representation (i.e., matrix approximation) where the decomposed components are regularized by the so-called L0 vector norm (which is not really a vector norm). (3) MVLRRL1: multi-view bi-clustering based on sparse low rank representation (i.e., matrix approximation) where the decomposed components are regularized by the L1 vector norm.

License GPL (>= 3)

Depends Rcpp (>= 0.12.0)

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

URL http://www.labhealthinfo.uconn.edu/multi-view-analytics/

Repository CRAN

Date/Publication 2016-04-03 16:31:05

R topics documented:

`gen` *Genotype data*

Description

Genotype data of 1003 subjects on 1000 simulated biallelic genetic variants.

Usage

```
gen
```

Format

1003x1000 matrix with rows representing genetic markers and columns representing subjects.

Details

Please refer to the section of simulation study in the paper (Sun et al, BMC genetics, 2014) for details.

`mvlrrl0` *Multi-view bi-clustering via L0-norm enforced sparse LRR*

Description

Identify consistent sample cluster among all views and simultaneously associated feature clusters per view. Clusters are obtained via finding sparse low rank representation (LRR) of input data matrices, where the sparsity is enforced using L0-norm. One sample cluster and its associated feature clusters are identified and returned each time this function is used. If multiple clusters are desired, call this function repeatedly with samples left unclustered.

Usage

```
mvlrrl0(datasets, svs, sz, seed=0, maxIter=1000, thres=0.00001, logLvl=0)
```

Arguments

`datasets`	List of input data, where each element is a matrix. For all the matrices, rows represent samples and columns represent features. These matrices are the characterization of a same set of samples from different perspectives (views), one matrix per view. So all the matrices should have the exact same rows, but can have different columns. The rows in the same position in all matrices represent a same sample.
`svs`	A numerical vector with length that equals to the number of view, which controls the sparsity of vector v in the decomposition of each view. It specifies the maximum number of non-zero elements in vector v.
`sz`	A number, which controls the sparsity of vector z in the decomposition of all views. It specifies the maximum number of non-zero elements in vector z.
`seed`	(Optional) A number, which gives the feature (the index) in the first view that will be used as a seed during initialization of vector v in the decomposition. Vector v is initialized with zero in all positions except the one corresponding to the given feature, which is initialized with one. If a seed is not provided, v is initialized with one in all positions. Because the optimization problem this function solves is non-convex, the returned solution is a local optimizer and could vary while different initializations are used. Based on the observation in our experiments, the seed feature is included in the returned feature cluster in most cases. So if there is a feature needed to be included in the feature cluster, it can be set as the seed feature using this argument.
`maxIter`	(Optional) Maximum number of loop iterations, which is one of the criteria that controls when to terminate the iterative process for searching for the optimal solution. Please read the referenced paper for details. The default value is 1000.
`thres`	(Optional) The other criteria (besides above 'maxIter' argument) for terminating the optimizing loop. When the difference between the two z from the two most recent consecutive iterations passes (is smaller than) this threshold, the loop is terminated. The default value is 0.00001.
`logLvl`	(Optional) Logging level, which can be set to either 0 or 1, controls the amount of printing, where a larger value means more printing. Default value is 0, which turns off the logging.

Details

This method identifies clusters via finding multi-view sparse low rank representations (LRR) of input data matrices. The LRR is obtained via matrix decomposition in the form: (zu)v, where both z and u are column vectors and their length is equal to the number of samples, (zu) represents the element-wise product between z and u, and v is a column vector and its length is equal to the number of features in the view. Vector z is a multiplier shared across all views, while both u and v are view specific. Sample cluster is read off directly from z by assigning samples with non-zero component in z to the cluster. Because z is shared across all views, the obtained sample cluster is a common cluster among all views. Feature cluster is obtained by assigning features with non-zero component in v to the cluster. Because v is view specific, feature cluster is view specific as well. Sample cluster and feature cluster are associated in the sense that they help determine each other. The sparsity of vector z and v is enforced using L0-norm.

Value

A list with following named fields:

`Cluster`	A binary vector (with 1 or 0 entries) of length equal to the sample size. It indicates whether a sample is in the identified cluster (with 1 in its corresponding position) or not (with 0).
`FeatClusters`	A list of binary vectors that give identified feature cluster for each view. Value 1 indicates the corresponding feature belongs to the identified feature cluster.
`U`	A matrix of with size of n x m, where n is the number of samples, m is the number of views. So each column gives the u in the decomposition of the corresponding view.
`V`	A list of vectors that give the v in the decomposition of each view.
`z`	The common multiplier shared across all views in their decomposition.

References

Jiangwen Sun, Jin Lu, Tingyang Xu, and Jinbo Bi *Multi-view Sparse Co-clustering via Proximal Alternating Linearized Minimization* Proceedings of the 32nd International Conference on Machine Learning (ICML), pp. 757-766

Examples

```
library(mvcluster)
data(phe)
data(gen)
        views <- list(phe,gen)
        result <- mvlrrl0(views,c(3,10),320)
```

`mvlrrl1` *Multi-view bi-clustering via L1-norm enforced sparse LRR*

Description

Identify consistent sample cluster among all views and simultaneously associated feature clusters per view. Clusters are obtained via finding sparse low rank representation (LRR) of input data matrices, where the sparsity is enforced using L1-norm. One sample cluster and its associated feature clusters are identified and returned each time this function is used. If multiple clusters are desired, call this function repeatedly with samples left unclustered.

Usage

```
mvlrrl1(datasets, lus, lvs, lz, maxOuter=100000, thresOuter=.00001, maxInner=10000,
 thresInner=.00001, logLvl=0)
```

Arguments

`datasets`	List of input data, where each element is a matrix. For all the matrices, rows represent samples and columns represent features. These matrices are the characterization of a same set of samples from different perspectives (views), one matrix per view. So all the matrices should have the exact same rows, but can have different columns. The rows in the same position in all matrices represent a same sample.
`lus`	A numerical vector with length that equals to the number of views, which controls the sparsity of vector u in the decomposition of each view. This parameter helps control the size of the sample cluster. A larger value indicates stronger sparsity and thus leads to a smaller sample cluster.
`lvs`	A numerical vector with length that equals to the number of views, which controls the sparsity of vector v in the decomposition of each view. This parameter helps control the size of the feature clusters. A larger value indicates stronger sparsity and thus leads to smaller feature clusters.
`lz`	A number, which controls the sparsity of vector z in the decomposition of all views. It helps to tune the size of the identified sample cluster. A larger value means stronger sparsity and thus leads to a smaller sample cluster.
`maxOuter`	(Optional) Maximum number of outer loop iterations, which is one of the criteria that controls when to terminate the outer loop in the process of searching for an optimal solution. The default value is 100000.
`thresOuter`	(Optional) The other criteria (besides the above 'maxOuter' argument) for terminating the outer loop. When the sum of squares of the difference between two consecutive outer loop iterations of vector z passes (is smaller than) this threshold, the loop is terminated. The default value is 0.00001.
`maxInner`	(Optional) Maximum number of inner loop iterations. It works the same way as above 'maxOuter' argument, but controls the inner loop in the optimization process. The default value is 10000.
`thresInner`	(Optional) This works the same way as above 'thresOuter' argument, but looks at the sum of squares of the difference between consecutive vector u (the other multiplier (besides vector z) of left singular vector) and controls the inner loop in the optimization process. The default value is 0.00001.
`logLvl`	(Optional) Logging level, which can be set to 0, 1 or 2. It controls the amount of printing, where a larger value means more printing. The default value is 0, which turns off the logging.

Details

This method identifies clusters via finding multi-view sparse low rank representations (LRR) of input data matrices. The LRR is obtained via matrix decomposition in the form: (zu)v, where both z and u are column vectors and their length are equal to the number of samples, (zu) represents the element-wise product between z and u, and v is a column vector and its length is equal to the number of features in the view. Vector z is a multiplier shared across all views, while both u and v are view specific. Sample cluster is read off directly from z by assigning samples with non-zero component in z to the cluster. Because z is shared across all views, the obtained sample cluster is a common cluster among all views. Feature cluster is obtained by assigning features with non-zero component in v to the cluster. Because v is view specific, feature cluster is view specific as well. Sample cluster and feature cluster are associated in the sense that they help determine each other. The sparsity of vector z, u and v is enforced using L1-norm.

Value

A list with following named fields:

`Cluster`	A binary vector (with 1 or 0 entries) of length equal to the sample size. It indicates whether a sample is in the identified cluster (with 1 in its corresponding position) or not (with 0).
`FeatClusters`	A list of binary vectors that give identified feature cluster for each view. Value 1 indicates the corresponding feature belongs to the identified feature cluster.
`U`	A matrix with size of n x m, where n is the number of samples, m is the number of views. So each column gives the u in the decomposition of the corresponding view.
`V`	A list of vectors that give the v in the decomposition of each view.
`z`	The common multiplier shared across all views in their decomposition.

Examples

```
library(mvcluster)
data(view1)
data(view2)
        views <- list(view1,view2)
        result <- mvlrrl1(views,c(1.2,1.2),c(166.6667,133.3333),1.2)
```

`mvsvdl1` *Multi-view bi-clustering via SSVD*

Description

Identify consistent sample cluster among all views and simultaneously associated feature clusters per view. Clusters are obtained via multi-view sparse singular value decomposition (SSVD). One sample cluster and its associated feature clusters are identified and returned through each call of this function. If multiple clusters are desired, call this function repeatedly with samples left unclustered.

Usage

```
mvsvdl1(datasets, lvs, lz, maxOuter=100000, thresOuter=.00001, maxInner=10000,
thresInner=.00001, logLvl=0)
```

Arguments

`datasets`	List of input data, where each element is a matrix. For all the matrices, rows represent samples and columns represent features. These matrices are the characterization of a same set of samples from different perspectives(views), one matrix per view. So all the matrices should have the exact same rows, but can have different columns. The rows in the same position in all matrices represent a same sample.
`lvs`	A numerical vector with length that equals to the number of view, which controls the sparsity of the right singular vector in the SVD of each view. This in turn controls the size of the feature clusters. Larger value indicates stronger sparsity and thus leads to smaller feature clusters.
`lz`	A number, which controls the sparsity of the left singular vector in the SSVD of all views. It helps to tune the size of the identified sample cluster. A larger value means stronger sparsity and thus a smaller sample cluster.
`maxOuter`	(Optional) Maximum number of outer loop iterations, which is one of the criteria that controls when to terminate the outer loop in the process of searching for an optimal solution. Please read the referenced paper for details. The default value is 100000.
`thresOuter`	(Optional) the other criteria (besides the above 'maxOuter' argument) for terminating the outer loop. When the sum of squares of the difference between two consecutive outer loop iterations of vector z passes (is smaller than) this threshold, the loop is terminated. The default value is 0.00001.
`maxInner`	(Optional) Maximum number of inner loop iterations. It works the same way as above 'maxOuter' argument, but controls the inner loop in the optimization process. The default value is 10000.
`thresInner`	(Optional) This works the same way as the above 'thresOuter' argument, but looks at the sum of squares of the difference between two consecutive iterations of vector u (the other multiplier (besides vector z) of left singular vector) and controls the inner loop in the optimization process. The default value is 0.00001.
`logLvl`	(Optional) Logging level, which can be set to 0, 1 and 2. This controls the amount of printing, where a larger value means more printing. The default value is 0, which turns off the logging.

Details

This method is multi-view sparse singular value decomposition (SSVD) based. Consistent sample cluster among views and associated view specific feature clusters are simultaneously identified. Sample and feature clusters are associated in the sense that they help determine each other. The sparsity of the singular vectors in the decomposition is enforced using L1-norm. Please refer to the referenced paper for more details.

Value

A list with following named fields:

`Cluster`	A binary vector (with 1 or 0 entries) of length equal to the sample size. It indicates whether a sample is in the identified cluster (with 1 in its corresponding position) or not (with 0).

`FeatClusters`	A list of binary vectors that give identified feature cluster for each view. Value 1 indicates the corresponding feature belongs to the identified feature cluster.
`U`	A matrix with size of n x m, where n is the number of samples, m is the number of views. So each column of this matrix is one of the two multipliers of the left singular vector in the SSVD of corresponding view.
`V`	A list of vectors that give the right singular vector in the SSVD of each view.
`z`	The common multiplier of the left singular vector in SSVD of all views.

References

Jiangwen Sun, Jinbo Bi, and Henry R. Kranzler *Multi-view Singular Value Decomposition for Disease Subtyping and Genetic Associations* BMC Genetics, 15(73):1-12, 2014

Examples

```
library(mvcluster)
data(phe)
data(gen)
        views <- list(phe,gen)
        result <- mvsvdl1(views,c(0.45,0.65),0.016)
```

---

phe *Phenotype data*

Description

Phenotype data of 1003 subjects on 10 simulated phenotypic variables.

Usage

```
phe
```

Format

1003x10 matrix with rows representing subjects and columns representing phenotypic variables.

Details

Please refer to the section of simulation study in the paper (Sun et al, BMC genetics, 2014) for details.

`view1` *View1 data*

Description

One of the two views simulated to mimic datasets from a real study, in which genes are characterized with expression patterns.

Usage

```
view1
```

Format

999x12 matrix with rows representing genes and columns representing various sources, such as different tissues or time points.

Details

We mimicked datasets from a real study in which genes are characterized with expression patterns and simulated datasets with implanted block structures that give both clusters of subjects and variables. Two views of data for 1000 subjects were created. There were 12 variables in view 1, and 15 variables in view 2. The data matrix of each view is created by randomly setting 0 or 1 to each entry with varying probability that is determined according to prefixed block structures projected in the data. More specifically, we start from a data matrix filled with all 0. Then we reset data entries inside and outside the blocks to 1 with probability 0.9 and 0.1, respectively. For simplifying the process and easy presentation, we had subjects in the two datasets well aligned and indexed from 1 to 1000; and variables were also indexed using consecutive number starting from 1. View 1 was designed to have two blocks. The first block consists of subjects from 1 to 400 and variables from 1 to 3. The second includes the 200 subjects indexed from 481 to 680 and variables from 4 to 6. Three blocks were included in view 2. The first block contains subjects from 1 to 240 and the first three variables. The second block consists of subjects from 241 to 480 and variable 4, 5 and 6. The last block includes 320 subjects indexed from 481 to 800 and variables from 7 to 9. By comparing blocks of the two views, it is obvious that there are three consistent blocks (i.e., containing same subjects) among the two views. Block 1 consists of 240 subjects and associates with variables from 1 to 3 in both view. There are 200 subjects in block 2. The associated variables are 4, 5 and 6 in view 1 and 7, 8 and 9 in view 2. Block 3 consists of 160 subjects and associates with variable from 1 to 3 in view 1 and from 4 to 6 in view 2.

`view2` *View2 data*

Description

The other one of the two views simulated to mimic datasets from a real study, in which genes are characterized with expression patterns.

Usage

```
view2
```

Format

999x12 matrix with rows representing genes and columns representing various sources, such as different tissues or time points.

Details

Please refer to the details included in the description for dataset view1.

Index