Package ‘wskm’

April 5, 2020

Version 1.4.40

Date 2020-04-04

Title Weighted k-Means Clustering

Maintainer He Zhao <Simon.Yansen.Zhao@gmail.com>

Depends R (>= 2.10), grDevices, stats, lattice, latticeExtra, fpc

Description Entropy weighted k-means (ewkm) by Liping Jing, Michael K. Ng and Joshua Zhexue Huang (2007) <doi:10.1109/TKDE.2007.1048> is a weighted subspace clustering algorithm that is well suited to very high dimensional data. Weights are calculated as the importance of a variable with regard to cluster membership. The two-level variable weighting clustering algorithm tw-k-means (twkm) by Xiaojun Chen, Xiaofei Xu, Joshua Zhexue Huang and Yunming Ye (2013) <doi:10.1109/TKDE.2011.262> introduces two types of weights, the weights on individual variables and the weights on variable groups, and they are calculated during the clustering process. The feature group weighted k-means (fgkm) by Xiaojun Chen, Yunminng Ye, Xiaofei Xu and Joshua Zhexue Huang (2012) <doi:10.1016/j.patcog.2011.06.004> extends this concept by grouping features and weighting the group in addition to weighting individual features.

License GPL (>= 3)

Copyright 2011-2014 Shenzhen Institutes of Advanced Technology Chinese Academy of Sciences

LazyLoad yes

LazyData yes

URL https://github.com/SimonYansenZhao/wskm,

http://english.siat.cas.cn/

BugReports https://github.com/SimonYansenZhao/wskm/issues

NeedsCompilation yes

Author Graham Williams [aut],
Joshua Z Huang [aut],
Xiaojun Chen [aut],
Qiang Wang [aut],
Longfei Xiao [aut],
He Zhao [cre]

Repository CRAN

Date/Publication 2020-04-05 00:00:02 UTC

R topics documented:

`ewkm` *Entropy Weighted K-Means*

Description

Perform an entropy weighted subspace k-means.

Usage

```
ewkm(x, centers, lambda=1, maxiter=100, delta=0.00001, maxrestart=10)
```

Arguments

`x`	numeric matrix of observations and variables.
`centers`	target number of clusters or the initial centers for clustering.
`lambda`	parameter for variable weight distribution.
`maxiter`	maximum number of iterations.
`delta`	maximum change allowed between iterations for convergence.
`maxrestart`	maximum number of restarts. Default is 10 so that we stand a good chance of getting a full set of clusters. Normally, any empty clusters that result are removed from the result, and so we may obtain fewer than k clusters if we don't allow restarts (i.e., maxrestart=0). If < 0 then there is no limit on the number of restarts and we are much more likely to get a full set of k clusters.

Details

The entopy weighted k-means clustering algorithm is a subspace clusterer ideal for high dimensional data. Along with each cluster we also obtain variable weights that provide a relative measure of the importance of each variable to that cluster.

The algorithm is based on the k-means approach to clustering. An initial set of k means are identified as the starting centroids. Observartions are clustered to the nearest centroid according to a distance measure. This defines the initial clustrering. New centroids are then identified based on these clusters.

Weights are then calculated for each variable within each cluster, based on the current clustering. The weights are a measure of the relative importance of each variable with regard to the membership of the observations to that cluster. These weights are then incorporated into the distance function, typically reducing the distance for the more important variables.

New centroids are then calculated, and using the weighted distance measure each observation is once again clustered to its nearest centroid.

The process continues until convergence (using a measure of dispersion and stopping when the change becomes less than delta) or until a specified number of iterations has been reached (maxiter).

Large lambda (e,g, > 3) lead to a relatively even distribution of weights across the variables. Small lambda (e.g., < 1) lead to a more uneven distribution of weights, giving more discrimintation between features. Recommended values are between 1 and 3.

Always check the number of iterations, the number of restarts, and the total number of iterations as they give a good indication of whether the algorithm converged.

As with any distance based algorithm, be sure to rescale your numeric data so that large values do not bias the clustering. A quick rescaling method to use is `scale`.

Value

Returns an object of class "kmeans" and "ewkm", compatible with other functions that work with kmeans objects, such as the 'print' method. The object is a list with the following components in addition to the components of the kmeans object:

weights: A matrix of weights recording the relative importance of each variable for each cluster.

iterations: This reports on the number of iterations before termination. Check this to see whether the maxiters was reached. If so then the algroithm may not be converging,and thus the resulting clustering may not be particularly good.

restarts: The number of times the clustering restarted because of a disappearing cluster resulting from one or more k-means having no observations associated with it. An number here greater than 0 indicates that the algorithm is not converging on a clustering for the given k. It is recommended that k be reduced.

total.iterations: The total number of iterations over all restarts.

Author(s)

Qiang Wang, Xiaojun Chen, Graham J Williams, Joshua Z Huang

References

Liping Jing, Michael K. Ng and Joshua Zhexue Huang (2007). An Entropy Weighting k-Means Algorithm for Subspace Clustering of High-Dimensional Sparse Data. IEEE Transactions on Knowledge and Data Engineering, 19(8), 1026–1041.

See Also

plot.ewkm.

Examples

```
myewkm <- ewkm(iris[1:4], 3, lambda=0.5, maxiter=100)

plot(iris[1:4], col=myewkm$cluster)

# For comparative testing

mykm <- kmeans(iris[1:4], 3)

plot(iris[1:4], col=mykm$cluster)
```

---

fgkm *Feature Group Weighting K-Means for Subspace clustering*

Description

Perform an feature group weighting subspace k-means.

Usage

```
fgkm(x, centers, groups, lambda, eta, maxiter=100, delta=0.000001,
     maxrestart=10,seed=-1)
```

Arguments

`x` numeric matrix of observations and features.

`centers` target number of clusters or the initial centers for clustering.

`groups` a string give the group information, formatted as "0,1,2,4;3,5;6,7,8" or "0-2,4;3,5;6-8", where ";" defines a group; or a vector of length of features, each element of the vector indicates the group of the feature. For example, c(1,1,1,2,1,2,3,3,3) is the same as "0-2,4;3,5;6-8", or even c("a","a","a","b","a","b","c","c","c").

`lambda` parameter of feature weight distribution.

`eta` parameter of group weight distribution.

`delta` maximum change allowed between iterations for convergence.

`maxiter` maximum number of iterations.

`maxrestart` maximum number of restarts. Default is 10 so that we stand a good chance of getting a full set of clusters. Normally, any empty clusters that result are removed from the result, and so we may obtain fewer than k clusters if we don't allow restarts(i.e., maxrestart=0). If < 0, then there is no limit on the number of restarts and we are much likely to get a full set of k clusters.

`seed` random seed. If it was set below 0, then a randomly generated number will be assigned.

Details

The feature group weighting k-means clustering algorithm is a extension to ewkm, which itself is a soft subspace clustering method.

The algorithm weights subspaces in both feature groups and individual features.

Always check the number of iterations, the number of restarts, and the total number of iterations as they give a good indication of whether the algorithm converged.

As with any distance based algorithm, be sure to rescale your numeric data so that large values do not bias the clustering. A quick rescaling method to use is `scale`.

Value

Return an object of class "kmeans" and "fgkm", compatible with other function that work with kmeans objects, such as the 'print' method. The object is a list with the following components in addition to the components of the kmeans object:

`cluster` A vector of integer (from 1:k) indicating the cluster to which each point is allocated.

`centers` A matrix of cluster centers.

`featureWeight` A matrix of weights recording the relative importance of each feature for each cluster.

`groupWeight` A matrix of group weights recording the relative importance of each feature group for each cluster.

`iterations` This report on the number of iterations before termination. Check this to see whether the maxiters was reached. If so then teh algorithm may not be converging, and thus the resulting clustering may not be particularly good.

`restarts` The number of times the clustering restarted because of a disappearing cluster resulting from one or more k-means having no observations associated with it. An number here greater than zero indicates that the algorithm is not converging on a clustering for the given k. It is recommended that k be reduced.

`totalIterations`
The total number of iterations over all restarts.

`totolCost` The total cost calculated in the cost function.

Author(s)

Longfei Xiao <`lf.xiao@siat.ac.cn`>

References

Xiaojun Chen, Yunming Ye, Xiaofei Xu and Joshua Zhexue Huang (2012). A Feature Group Weighting Method for Subspace Clustering of High-Dimensional Data. Pattern Recognition, 45(1), 434–446.

See Also

`kmeans` `ewkm` `twkm`

Examples

```
# The data fgkm.sample has 600 objects and 50 dimensions.
# Scale the data before clustering
x <- scale(fgkm.sample)

# Group information is formated as below.
# Each group is separated by ';'.
strGroup <- "0-9;10-19;20-49"
groups <- c(rep(0, 10), rep(1, 10), rep(2, 30))

# Use the fgkm algorithm.
myfgkm <- fgkm(x, 3, strGroup, 3, 1, seed=19)
myfgkm2 <- fgkm(x, 3, groups, 3, 1, seed=19)
all.equal(myfgkm, myfgkm2)

# You can print the clustering result now.
myfgkm$cluster
myfgkm$featureWeight
myfgkm$groupWeight
myfgkm$iterations
myfgkm$restarts
myfgkm$totiters
myfgkm$totss

# Use a cluster validation method from package 'fpc'.

# real.cluster is the real class label of the data 'fgkm.sample'.
real.cluster <- rep(1:3, each=200)

# cluster.stats() computes several distance based statistics.
kmstats <- cluster.stats(d=dist(x), as.integer(myfgkm$cluster), real.cluster)

# corrected Rand index
kmstats$corrected.rand

# variation of information (VI) index
kmstats$vi
```

fgkm.sample *Sample dataset to illustrate the fgkm algorithm.*

Description

A sample dataset of 50 variables and 600 observations.

Usage

```
fgkm.sample
```

Format

The `fgkm.sample` dataset is a data frame with 600 observations and 50 variables.

See Also

`fgkm`.

plot.ewkm *Plot Entropy Weighted K-Means Weights*

Description

Plot a heatmap showing the variable weights from the subspace clustering.

Usage

```
## S3 method for class 'ewkm'
plot(x, ...)
## S3 method for class 'ewkm'
levelplot(x, ...)
```

Arguments

`x`	an object of class ewkm.
`...`	arguments passed on through to heatmap.

Details

The entopy weighted k-means clustering algorithm is a subspace clusterer ideal for high dimensional data. Along with each cluster we also obtain variable weights that provide a relative measure of the importance of each variable to that cluster.

This plot visualises these relative measures of variable importance for each of the clusters using a heatmap. The top dendrogram highlights the relationship between the clusters and the right side dendrogram provides a visual clue to the correlation between the variables.

The plot.ewkm() function uses heatmap() to display the weights. The levelplot.ewkm() uses levelplot() with dendrogramGlobs from the lattice package. Note that plot() will immediately draw the plot while levelplot() does not draw immediately but returns a result object which must be plot()ed.

Author(s)

Graham J Williams

Examples

```
myewkm <- ewkm(iris[1:4], 3, lambda=0.5, maxiter=100)

plot(myewkm)
```

`predict.ewkm` *Predict method for* `ewkm` *model.*

Description

Return the nearest cluster to each observation based on a Euclidean distance with each variable weighted differently per cluster.

Usage

```
## S3 method for class 'ewkm'
predict(object, data, ...)
```

Arguments

`object`	object of class `ewkm`.
`data`	the data that needs to be predicted. Variables should have the same names and order as used in building the `ewkm` model.
`...`	other arguments.

Value

a vector of cluster numbers of length `nrow(data)`.

Author(s)

Graham Williams (Togaware)

See Also

ewkm

twkm *Two-level variable weighting clustering*

Description

Two-level variable weighting clustering.

Usage

```
twkm(x, centers, groups, lambda, eta, maxiter=100, delta=0.000001,
     maxrestart=10,seed=-1)
```

Arguments

x numeric matrix of observations and features.

centers target number of clusters or the initial centers for clustering.

groups a string give the group information, formatted as "0,1,2,4;3,5;6,7,8" or "0-2,4;3,5;6-8", where ";" defines a group; or a vector of length of features, each element of the vector indicates the group of the feature. For example, c(1,1,1,2,1,2,3,3,3) is the same as "0-2,4;3,5;6-8", or even c("a","a","a","b","a","b","c","c","c").

lambda parameter of feature weight distribution.

eta parameter of group weight distribution.

delta maximum change allowed between iterations for convergence.

maxiter maximum number of iterations.

maxrestart maximum number of restarts. Default is 10 so that we stand a good chance of getting a full set of clusters. Normally, any empty clusters that result are removed from the result, and so we may obtain fewer than k clusters if we don't allow restarts(i.e., maxrestart=0). If < 0, then there is no limit on the number of restarts and we are much likely to get a full set of k clusters.

seed random seed. If it was set below 0, then a randomly generated number will be assigned.

Details

The two-level variable weighting clustering algorithm is a extension to ewkm, which itself is a soft subspace clustering method.

The algorithm weights subspaces in both feature groups and individual features.

Always check the number of iterations, the number of restarts, and the total number of iterations as they give a good indication of whether the algorithm converged.

As with any distance based algorithm, be sure to rescale your numeric data so that large values do not bias the clustering. A quick rescaling method to use is `scale`.

Value

Return an object of class "kmeans" and "twkm", compatible with other function that work with kmeans objects, such as the 'print' method. The object is a list with the following components in addition to the components of the kmeans object:

`cluster` A vector of integer (from 1:k) indicating the cluster to which each point is allocated.

`centers` A matrix of cluster centers.

`featureWeight` A vector of weights recording the relative importance of each feature.

`groupWeight` A vector of group weights recording the relative importance of each feature group.

`iterations` This report on the number of iterations before termination. Check this to see whether the maxiters was reached. If so then teh algorithm may not be converging, and thus the resulting clustering may not be particularly good.

`restarts` The number of times the clustering restarted because of a disappearing cluster resulting from one or more k-means having no observations associated with it. An number here greater than zero indicates that the algorithm is not converging on a clustering for the given k. It is recommended that k be reduced.

`totalIterations` The total number of iterations over all restarts.

`totolCost` The total cost calculated in the cost function.

Author(s)

Xiaojun Chen <`xjchen.hitsz@gmail.com`>

References

Xiaojun Chen, Xiaofei Xu, Joshua Zhexue Huang and Yunming Ye (2013). TW-k-Means: Automated Two-level Variable Weighting Clustering Algorithm for Multiview Data. IEEE Transactions on Knowledge and Data Engineering, 25(4), 932–944.

See Also

`kmeans` `ewkm` `fgkm`

Examples

```
# The data twkm.sample has 2000 objects and 410 variables.
# Scale the data before clustering
x <- scale(twkm.sample[,1:409])

# Group information is formated as below.
# Each group is separated by ';'.
strGroup <- "0-75;76-291;292-355;356-402;403-408"
groups <- c(rep(0, abs(0-75-1)), rep(1, abs(76-291-1)), rep(2, abs(292-355-1)),
            rep(3, abs(356-402-1)), rep(4, abs(403-408-1)))


# Use the twkm algorithm.
mytwkm <- twkm(x, 10, strGroup, 3, 1, seed=19)
mytwkm2 <- twkm(x, 10, groups, 3, 1, seed=19)
all.equal(mytwkm, mytwkm2)

# You can print the clustering result now.
mytwkm$cluster
mytwkm$featureWeight
mytwkm$groupWeight
mytwkm$iterations
mytwkm$restarts
mytwkm$totiters
mytwkm$totss

# Use a cluster validation method from package 'fpc'.

# real.cluster is the real class label of the data 'twkm.sample'.
real.cluster <- twkm.sample[,410]

# cluster.stats() computes several distance based statistics.
kmstats <- cluster.stats(d=dist(x), as.integer(mytwkm$cluster), real.cluster)

# corrected Rand index
kmstats$corrected.rand

# variation of information (VI) index
kmstats$vi
```

twkm.sample *Sample dataset to test the twkm algorithm.*

Description

A sample dataset of 410 variables and 2000 observations. The last variable is the class variable, and should be removed before clustering. The grouping of the variables are "0-75;76-291;292-355;356-402;403-408".

Usage

```
twkm.sample
```

Format

The `twkm.sample` dataset is a data frame with 2000 observations and 50 variables.

See Also

`twkm`.

Index