R package klic

Alessandra Cabassi

2018-05-26

Contents

The R package `klic` (**K**ernel **L**earning **I**ntegrative **C**lustering) contains a collection of tools for integrative clustering of multiple data types.

The main functions are:

- `consensusCluster`, a function that can be used to perform consensus clustering on one dataset and obtain a co-clustering matrix (Monti et al. 2003)
- `lmkkmeansTrain`, a function implemented by Gonen and Margolin (2014) that performs localised kernel k-means on a set of kernel matrices (such as co-clustering matrices, for example). Together, these two functions can be used to do Kernel Learning Integrative Clustering (KLIC; Cabassi and Kirk, 2018).

The package also includes `coca`, a function that we implemented in order to run the Cluster-Of-Clusters-Analysis of TCGA (2012) and compare it with KLIC.

Other functions included in the package are:

- `spectrumShift`, to shift the eigenvalues of a co-clustering matrix in order to make it positive semi-definite;
- `plotSimilarityMatrix`, to plot a similarity matrix with the option to divide the rows of the matrix according to the clusters and to include a vector of labels for each row;
- `kkmeansTrain.R`, another function implemented by Gonen and Margolin (2004) to perform kernel k-means with only one kernel matrix (Girolami, 2002);
- `copheneticCorrelation`, to calculate the cophenetic correlation of a similarity matrix.

1 KLIC

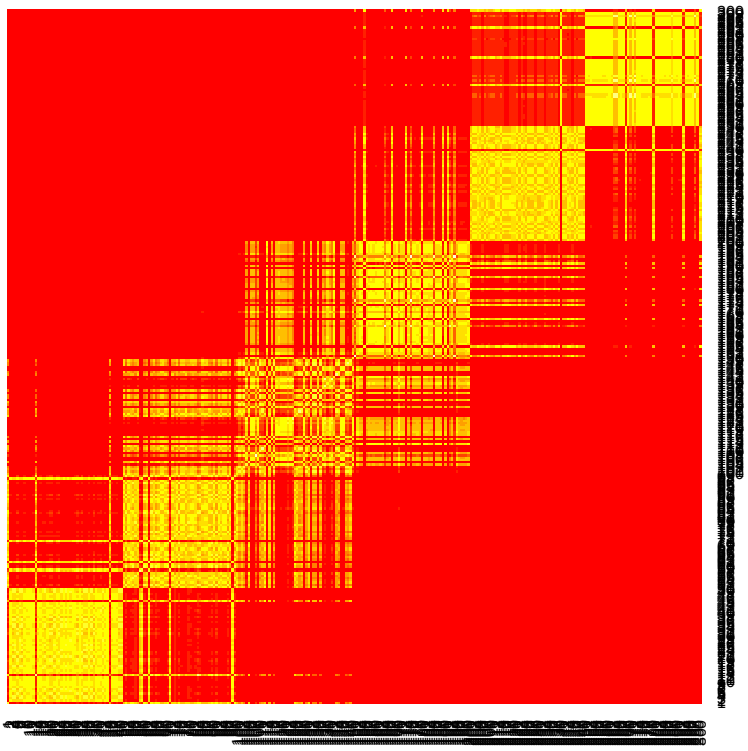
First, we generate four datasets with the same clustering structure (6 clusters of equal size) and different levels of noise.

```
## Load R packages
library(klic)
library(MASS)
## Set parameters
n_variables <- 2
n_obs_per_cluster <- 50
n_clusters <- 6
n_separation_levels <- 4
Sigma <- diag(n_variables)
N <- n_obs_per_cluster*n_clusters
```

```
P <- n_variables
## Generate synthetic data
data <- list()
for(i in 1:n_separation_levels){
    data[[i]] <- array(NA, c(N, P))
    mu = rep(NA, N)
    for(k in 1:n_clusters){
      mu = rep(k*(i-1), n_variables)
      data[[i]][((k-1)*n_obs_per_cluster+1):(k*n_obs_per_cluster),] <- mvrnorm(n = n_obs_per_cluster, mu
    }
}
```

Now we can use the `consensusClustering` function to compute a consensus matrix for each dataset.

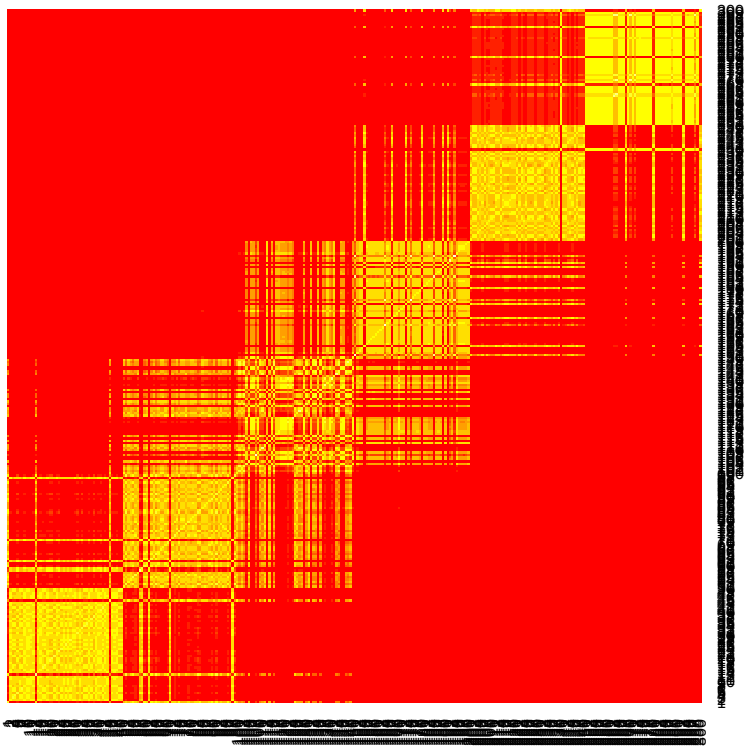
```
## Compute co-clustering matrices for each dataset
CM <- array(NA, c(N, N, n_separation_levels))
for(i in 1: n_separation_levels){
  # Scale the columns to have zero mean and unitary variance
  scaledData <- scale(data[[i]])
  # Use consensus clustering to find the consensus matrix of each dataset
  CM[,,i] <- consensusCluster(scaledData, K = 6, B = 1000)
}
## Plot consensus matrix of one of the datasets
heatmap(CM[,,3], Colv = NA, Rowv = NA)
```



Before using kernel methods, we need to make sure that all the consensus matrices are positive semi-definite.

```
## Check if consensus matrices are PSD and shift eigenvalues if needed.
for(i in 1: n_separation_levels){
  CM[,,i] <- spectrumShift(CM[,,i], verbose = FALSE)
}
```

```
## Plot updated consensus matrix of one of the datasets
heatmap(CM[,,3], Colv = NA, Rowv = NA)
```



Now we can perform localised kernel k-kmeans on the set of consensus matrices using the function `lmkkmeans` to find a global clustering. This functions also need to be provided with a list of parameters containing the prespecified number of clusters and the maximum number of iterations for the k-means algorihtm.

```
## Perform localised kernel k-means on the consensus matrices
library(Matrix)
library(Rmosek)
parameters <- list()
parameters$cluster_count <- 6 # set the number of clusters K
parameters$iteration_count <- 100 # set the maximum number of iterations
lmkkm <- lmkkmeansTrain(CM, parameters)
```

The output of `lmkkmeansTrain` contains, among other things, the final cluster labels. To compare two clusterings, such as for example the true clustering (that is known in this case) and the clustering found with KLIC, we suggest to use the `adjustedRandIndex` function of the R package `mclust`. An ARI close to 1 indicates a high similarity between the two partitions of the data.

```
## Compare clustering found with KLIC to the true one
ones <- rep(1, n_obs_per_cluster)
true_labels <- c(ones, ones*2, ones*3, ones*4, ones*5, ones*6)
library(mclust, verbose = FALSE)
```

```
## Package 'mclust' version 5.4
## Type 'citation("mclust")' for citing this R package in publications.
```

```
adjustedRandIndex(true_labels, lmkkm$clustering)
```

```
## [1] 0.9303159
```

If the global number of clusters is not known, one can find the weights and clusters for different values of K

and then find the one that maximises the silhouette.

```
## Find the value of k that maximises the silhouette

# Initialise array of kernel matrices
maxK = 6
KM <- array(0, c(N, N, maxK-1))
clLabels <- array(NA, c(maxK-1, N))

parameters <- list()
parameters$iteration_count <- 100 # set the maximum number of iterations

for(i in 2:maxK){

  # Use kernel k-means with K=i to find weights and cluster labels
  parameters$cluster_count <- i # set the number of clusters K
  lmkkm <- lmkkmeansTrain(CM, parameters)

  # Compute weighted matrix
  for(j in 1:dim(CM)[3]){
    KM[,,i-1] <- KM[,,i-1] + (lmkkm$Theta[,j]%*%t(lmkkm$Theta[,j]))*CM[,,j]
  }

  # Save cluster labels
  clLabels[i-1,] <- lmkkm$clustering
}

# Find value of K that maximises silhouette
maxSil <- maximiseSilhouette(KM, clLabels, maxK = 6)
```

```
## Loading required package: cluster
```

```
maxSil$k
```

```
## [1] 6
```

The same can be done simply using the function `klic`

```
klic <- klic(data, M = n_separation_levels, individualK = c(6,6,6,6))
```

2 COCA

COCA is an alternative method to do integrative clustering of multiple data types (TGCA, 2012).

```
## Fill label matrix with clusterings found with the k-means clustering algorithm
labelMatrix <- array(NA, c(n_clusters, N, n_separation_levels))
for(i in 1:n_separation_levels){
  output <- kmeans(data[[i]], n_clusters)
  for(k in 1:n_clusters){
    labelMatrix[k,,i] <- (output$cluster==k)
  }
}
# Convert label matrix from logic to numeric matrix
labelMatrix <- labelMatrix*1
```

`labelMatrix` is now a matrix that contains the binary labels of the data given by the k-means clustering algorithm for each dataset separately.

```
# Build MOC matrix
MOC = rbind(labelMatrix[,,1],labelMatrix[,,2], labelMatrix[,,3], labelMatrix[,,4])
# Use COCA to find global clustering
clusters <- coca(t(MOC), K = 6, hclustMethod = "average")
# Compare clustering to the true labels
ari <- adjustedRandIndex(true_labels, clusters)
ari
```

```
## [1] 0.7274115
```

3 Miscellanea

`kkmeansTrain` is the analogous of `lmkkmeans`, for just one dataset at a time.

```
## Set parameters of the kernel k-means algorithm
parameters <- list()
parameters$cluster_count <- 6
parameters$iteration_count <- 100
## Run kernel k-means
kkm <- kkmeansTrain(CM[,,3], parameters)
## Compare clustering to the true labels
clusterLabels <- kkm$clustering
adjustedRandIndex(true_labels, lmkkm$clustering)
```

```
## [1] 0.9303159
```

We also included `copheneticCorrelation`, a function that calculates the cophenetic correlation coefficient of a similarity matrix

```
## Compute cophenetic correlation coefficient for each consensus matrix
cc <- rep(NA, n_separation_levels)
for(i in 1:n_separation_levels){
  cc[i] <- copheneticCorrelation(CM[,,i])
}
cc
```

```
## [1] 0.8766906 0.8426752 0.9432639 0.9695395
```

There is also a function that can be used to plot similarity matrices, ordered by cluster label, and with an additional side label for each row and to save the output in different formats. It is called `plotSimilarityMatrix`.

4 References

Cabassi, A. and Kirk, P. D. W. (2018). Multiple kernel learning for integrative consensus clustering. In preparation.

Gonen, M. and Margolin, A. A. (2014). Localized Data Fusion for Kernel k-Means Clustering with Application to Cancer Biology. NIPS, (i), 1–9.

Monti, S. et al. (2003). Consensus Clustering: A Resampling-Based Method for Class Discovery and Visualization of Gene. Machine Learning, 52(i), 91–118.

The Cancer Genome Atlas (2012). Comprehensive molecular portraits of human breast tumours. Nature, 487(7407), 61–70.