中國人民大學
RENMIN UNIVERSITY OF CHINA

The application of copula dependence structure in unsupervised clustering: A case study on brain cells expression data

孙楚婷 2019000155 2021.05

Data_analysis_final_report

Data Preprocessing

We begin with the brain cell gene expression data and extract the distinct levels of cell types. There are 8 types of cells.

```
library(aricode)

# import data
library(readr)
brain_celltype <- read_csv("brain_celltype.txt", col_names = FALSE)
brain_celltype<-data.frame(brain_celltype)

brain_expression <- read_table2("brain_expression.txt")
brain_expression<-data.frame(brain_expression)
# transform to hit/nonhit binary data
brain_expr_01<-data.frame(brain_expression[,-1]>0)

# celltype summary
celltypes<-levels(as.factor(brain_celltype[,1]))
celltypes
```

```
## [1] "astrocytes"        "endothelial"       "fetal_quiescent"
## [4] "fetal_replicating" "microglia"         "neurons"
## [7] "oligodendrocytes"  "OPC"
```

```
set.seed(817)
```

Main Method

We apply latent gaussian variable substitution method in the unsupervised clustering task. We will include both the iid case and dependent case, and in comparison we also perform the naive K-means algorithm and Hamming distance clustering.

Average Gene Expression

```
# p_i
geneavg<-apply(brain_expr_01,1,mean)
gened<-qnorm(geneavg)

# mu_j
cellavg<-apply(brain_expr_01,2,mean)
```

i.i.d. Latent Gaussian Substitution

As mentioned in the class, we will assume that the expression of genes are controled by some latent variables following i.i.d. normal distribution and when the latent variable is below some threshold, this gene will express(hit), that is

$$Y_{ij} = 1 \Leftrightarrow X_{ij} \leq d_i, \quad X_{ij} \sim N(0,1) \quad iid, \quad d_i = \Phi^{-1}(p_i)$$

where p_i is the average expression level of i^{th} gene.

conditional expectation substitution

We substitute the hit/nonhit states with conditional expectations $E(X_{ij}|X_{ij} < \Phi^{-1}(p_i))$ if $Y_{ij} = 1$ and otherwise $E(X_{ij}|X_{ij} > \Phi^{-1}(p_i))$. Then perform following gene-scale penalization by multiplying the standard deviation of each gene aiming to control the upper limit.The matrix is also called score matrix.

$$U_{ij}^* = U_{ij} * std(U_i)$$

here $U_i = \{U_{i1}, \ldots, U_{in}\}$ is the score of i^{th} gene.

```
subfuniid<-function(x){
  p<-mean(x>0)
  y<-x
  # y==0
  y[x==0]<-dnorm(qnorm(p))/(1-p)
  # Y>=1
  y[x>0]<--dnorm(qnorm(p))/p

  return(y*sqrt(var(unlist(y))))
}
iidsub<-t(apply(brain_expr_01,1,subfuniid))
```

Dependent Structure Substitution

Similar as the iid case but we assume that the latent variables are not indepent now. The corresponding latent structure is

$$Y_{ij} = 1 \Leftrightarrow X_{ij} \leq d_i, \quad X_{ij} = \rho_i \mu_j + \sqrt{1-\rho_i^2} z_{ij}, \quad z_{ij} \sim N(0,1) \quad iid, \quad d_i = \Phi^{-1}(p_i)$$

where ρ_i is the dependent coefficent, and μ_j is the average expression level of cell which is supposed to follow a Gaussian mixture model.

Estimate ρ with two-step EM algorithm

Before substitution, we may first estimate the parameters from data. Firstly we consider an easy case and assume there's only one ρ in the latent dependent model. Then the log-likelihood is

$$\ell(Y, \mu|\rho, \pi, \nu, \sigma) = \sum_{ij} Y_{ij} \log(\frac{p_i^{\mu_j}}{1-p_i^{\mu_j}}) + \sum_{ij} \log(1-p_i^{\mu_j}) + \sum_j \ell(\mu_j|\pi, \nu, \sigma)$$

where $p_i^{\mu_j} = \Phi(\frac{\Phi^{-1}(p_i) - \rho\mu_j}{\sqrt{1-\rho^2}})$ and $\ell(\mu|\pi, \nu, \sigma)$ is the log-likehood of the Gaussian mixture model. We maximize the log likelihood function by `optim` algorithm and get the MLE of ρ.

```
subfunllh<-function(x,cellavg,rhohat){
  p<-mean(x>0)
  d<-(qnorm(p)-rhohat*cellavg)/sqrt(1-rhohat^2)
  y<-x
  # y==0
  y[x==0]<-log(1-pnorm(d[x==0]))
  # Y>=1
  y[x>0]<-log(pnorm(d[x>0]))

  return(sum(y))
}


llh<-function(rho,Y,p_i,mu_j){
  depllh<-apply(Y,1,subfunllh,cellavg=mu_j,rhohat=rho)

  l<-sum(depllh)

  return(-l)
}
```

Optimize the profile log-likelihood given μ_j:

```
res<-optim(par=c(rho=0.2),fn=llh, gr = NULL, Y=brain_expr_01,p_i=geneavg,mu_j=cell
avg,method="L-BFGS-B",lower=c(-0.8), upper=c(0.8) )

rhohat<-res$par

rhohat
```

```
##        rho
## -0.4617678
```

Conditional Expectation Substitution

Similar as the iid case, we should replace $Y_{ij} = 1$ by $E(X_{ij}|X_{ij} < \Phi^{-1}(p_i))$, however, since we only need the simularity between cells, we may need a scaled form of the substitution values.

```
subfundep<-function(x,cellavg,rhohat){
  p<-mean(x>0)
  d<-(qnorm(p)-rhohat*cellavg)/sqrt(1-rhohat^2)
  y<-x
  # y==0
  y[x==0]<-dnorm(d[x==0])/(1-pnorm(d[x==0]))
  # Y>=1
  y[x>0]<--dnorm(d[x>0])/pnorm(d[x>0])


  return(y*sqrt(var(unlist(y))))
}
depsub<-t(apply(brain_expr_01,1,subfundep,cellavg=cellavg,rhohat=rhohat))
```

Simularity Matrix

Calculate the simularity matrix by Cosine kernel:

$$s_{ij} =< U_i, U_j > /|U_i||U_j|$$

where U_i is the **cell-scale** score.

```
penfun<-function(x){
  return(1/sqrt(sum((unlist(x)^2))))
}
iidD<-diag(apply(iidsub,2,penfun))
depD<-diag(apply(depsub,2,penfun))

iidS<-as.matrix(iidD%*%t(iidsub)%*%iidsub%*%iidD)
depS<-as.matrix(depD%*%t(depsub)%*%depsub%*%depD)
```

Model Evaluation

In order to assess the performance of our method, we may introduce two metrics to evaluate the results. Here we consider two assesments: `purity` and `Normalized mutual information (nmi)` suggested in evaluation (https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html).

- purity: $\text{purity}(\Omega, \mathbb{C}) = \frac{1}{N}\sum_k \max_j \left|\omega_k \cap c_j\right|$
- nmi: $\text{NMI}(\Omega, C) = \frac{I(\Omega;\mathbb{C})}{[H(\Omega)+H(\mathbb{C})]/2}$, where $I(,)$ is the mutual information and $H(.)$ is the entropy.

We will repeat $B = 10$ times for each method and compare the average metrics

Clustering

Now with a new distance matrix, we are able to conduct unsupervised clustering by `K-medoids` algorithm which takes a distance matrix as input. Both the `k-means` and `k-medoids` algorithms attempt to minimize the distance between points labeled to be in a cluster and a point designated as the center of that cluster. But k-medoids minimizes a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances, it is more robust to noise and outliers than k-means.

```r
library(kmed)


namefun<-function(x){
  y=names(which.max(x))
  return(y)
}

B<-10


# iid case
clunum<-rep(0,B)
purity<-rep(0,B)
nmi<-rep(0,B)

for (b in 1:B) {
  iidres<- fastkmed(1-iidS, 8)
iidclu<- iidres$cluster
# show cluster center
#iidcen<-brain_celltype[iidres$medoid,1]

# show confusion matrix
acciid<-as.matrix(table(iidclu,brain_celltype[,1]))
#acciid

# major types in each cluster

clunum[b]<-length(levels(as.factor(apply(acciid,1,namefun))))

purity[b]<-sum(apply(acciid, 1, max))/sum(acciid)
nmi[b]<-NMI(iidclu,brain_celltype[,1])
}

# number of distinct types recognized
mean(clunum)
```

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

```r
# purity
mean(purity)
```

```
## [1] 0.7119048
```

```r
# NMI
mean(nmi)
```

```
## [1] 0.485767
```

```
# dependent case
clunum<-rep(0,B)
purity<-rep(0,B)
nmi<-rep(0,B)

for (b in 1:B) {
  depres<- fastkmed(1-depS, 8)
depclu<- depres$cluster
accdep<-as.matrix(table(depclu,brain_celltype[,1]))


# major types in each cluster
clunum[b]<-length(levels(as.factor(apply(accdep,1,namefun))))


# purity
purity[b]<-sum(apply(accdep, 1, max))/sum(accdep)
# nmi
nmi[b]<-NMI(depclu,brain_celltype[,1])
}

# number of distinct types recognized
mean(clunum)
```

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

```
# purity
mean(purity)
```

```
## [1] 0.7142857
```

```
# NMI
mean(nmi)
```

```
## [1] 0.517354
```

We find that the accuracy of iid case and dependent case are similar, and they tends to treat the “fetal_quiescent” as “neurons” class. The first class in the clustering result centers around “neurons”, however the major membership of this class belong to “fetal_quiescent” class.

Comparison with Other Methods

Can’t find the implement of other methods, so I only perform the kmeans and naive Hamming distance clustering.

```
# K-medoids with Hamming distance

hamdis<- function(X) {
    D <- (1 - X) %*% t(X)
    D + t(D)
}

clunum<-rep(0,B)
purity<-rep(0,B)
nmi<-rep(0,B)

for (b in 1:B) {
  hamres<-fastkmed(hamdis(t(brain_expr_01)),8)

hamclu<- hamres$cluster
accham<-as.matrix(table(hamclu,brain_celltype[,1]))

# major types in each cluster
clunum[b]<-length(levels(as.factor(apply(accham,1,namefun))))


purity[b]<-sum(apply(accham, 1, max))/sum(accham)

nmi[b]<-NMI(hamclu,brain_celltype[,1])
}
# number of distinct types recognized
mean(clunum)
```

```
## [1] 3
```

```
# purity
mean(purity)
```

```
## [1] 0.5047619
```

```
# NMI
mean(nmi)
```

```
## [1] 0.304276
```

In K-means algorithm, we may try three cases:

- pure K-means with raw data
- K-means with iid substitution
- K-means with dependent structure substitution

```
# 1. pure K-means with the raw data
clunum<-rep(0,B)
purity<-rep(0,B)
nmi<-rep(0,B)

for (b in 1:B) {
  kmres<-kmeans(t(brain_expr_01),8)
kmclu<-kmres$cluster
acckm<-as.matrix(table(kmclu,brain_celltype[,1]))


# major types in each cluster
clunum[b]<-length(levels(as.factor(apply(acckm,1,namefun))))


purity[b]<-sum(apply(acckm, 1, max))/sum(acckm)

nmi[b]<-NMI(kmclu,brain_celltype[,1])
}

# number of distinct types recognized
mean(clunum)
```

```
## [1] 4.9
```

```
# purity
mean(purity)
```

```
## [1] 0.7783333
```

```
# NMI
mean(nmi)
```

```
## [1] 0.5956164
```

```
# 2. Kmeans with iid substitute
clunum<-rep(0,B)
purity<-rep(0,B)
nmi<-rep(0,B)

for (b in 1:B) {
  kmres_U<-kmeans(t(iidsub),8)
kmclu_U<-kmres_U$cluster
acckm_U<-as.matrix(table(kmclu_U,brain_celltype[,1]))


# major types in each cluster
clunum[b]<-length(levels(as.factor(apply(acckm_U,1,namefun))))


purity[b]<-sum(apply(acckm_U, 1, max))/sum(acckm_U)

nmi[b]<-NMI(kmclu_U,brain_celltype[,1])
}
# number of distinct types recognized
mean(clunum)
```

```
## [1] 4.4
```

```
# purity
mean(purity)
```

```
## [1] 0.7635714
```

```
# NMI
mean(nmi)
```

```
## [1] 0.5639982
```

```
# 3. k-means with dependent substitution

clunum<-rep(0,B)
purity<-rep(0,B)
nmi<-rep(0,B)

for (b in 1:B) {
  kmres_D<-kmeans(t(depsub),8)
kmclu_D<-kmres_D$cluster
acckm_D<-as.matrix(table(kmclu_D,brain_celltype[,1]))


# major types in each cluster
clunum[b]<-length(levels(as.factor(apply(acckm_D,1,namefun))))


purity[b]<-sum(apply(acckm_D, 1, max))/sum(acckm_D)

nmi[b]<-NMI(kmclu_D,brain_celltype[,1])

}

# number of distinct types recognized
mean(clunum)
```

```
## [1] 4.9
```

```
# purity
mean(purity)
```

```
## [1] 0.7833333
```

```
# NMI
mean(nmi)
```

```
## [1] 0.6087817
```

In view of `purity`, we may see that K-means with dependent substitution outperforms others, including K-medoids based on latent variable substitution, K-means based on iid latent variable substitution and pure K-means based on raw data. And the dependent structure seems to have slightly better performance than the iid version, while K-mediods based on Hamming distance is worst. Furthermore, when considering the distinct types recognized from these algorithms, we observed that pure K-means extracts less distinct types than our methods.

CNAE-9 case

We are also curious about the performance of our methods on the CNAE-9 datasets.

data importation and description

```
library(readr)
cnaedata<- read_csv("phpmcGu2X.csv")
cnaedata<-data.frame(cnaedata)
dim(cnaedata)
```

```
## [1] 1080  857
```

```
grouplabel<-cnaedata[,857]
table(grouplabel)
```

```
## grouplabel
##   1   2   3   4   5   6   7   8   9
## 120 120 120 120 120 120 120 120 120
```

```
cnaedata<-data.frame(t(cnaedata[,1:856]>0))

# p_i
keyavg<-apply(cnaedata,1,mean)
keyd<-qnorm(keyavg)

# mu_j
docavg<-apply(cnaedata,2,mean)
```

naive K-means

We also conduct pure K-means based on the raw data.

```
clunum<-rep(0,B)
purity<-rep(0,B)
nmi<-rep(0,B)

for (b in 1:B) {
  kmres_doc<-kmeans(t(cnaedata),9)
kmclu_doc<-kmres_doc$cluster
acckm_doc<-as.matrix(table(kmclu_doc,grouplabel))


# major types in each cluster
clunum[b]<-length(levels(as.factor(apply(acckm_doc,1,namefun))))


# purity
purity[b]<-sum(apply(acckm_doc, 1, max))/sum(acckm_doc)

#nmi
nmi[b]<-NMI(kmclu_doc,grouplabel)
}
# number of distinct types recognized
mean(clunum)
```

```
## [1] 8.4
```

```
# purity
mean(purity)
```

```
## [1] 0.5425926
```

```
# NMI
mean(nmi)
```

```
## [1] 0.4497386
```

Hamming distance for K-medoid clustering

```
clunum<-rep(0,B)
purity<-rep(0,B)
nmi<-rep(0,B)
for (b in 1:B) {
  hamres_doc<-fastkmed(hamdis(t(cnaedata)),9)

hamclu_doc<- hamres_doc$cluster
accham_doc<-as.matrix(table(hamclu_doc,grouplabel))


# major types in each cluster
clunum[b]<-length(levels(as.factor(apply(accham_doc,1,namefun))))


#purity
purity[b]<-sum(apply(accham_doc, 1, max))/sum(accham_doc)
#nmi
nmi[b]<-NMI(hamclu_doc,grouplabel)
}
# number of distinct types recognized
mean(clunum)
```

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

```
# purity
mean(purity)
```

```
## [1] 0.3907407
```

```
# NMI
mean(nmi)
```

```
## [1] 0.2963202
```

dependent structure substitution and clustering by K-means

```
res_doc<-optim(par=c(rho=0.2),fn=llh, gr = NULL, Y=cnaedata,p_i=keyavg,mu_j=docav
g,method="L-BFGS-B",lower=c(-0.8), upper=c(0.8) )

rhohatdoc<-res_doc$par
rhohatdoc
```

```
##        rho
## -0.0782741
```

```
docsub<-t(apply(cnaedata,1,subfundep,cellavg=docavg,rhohat=rhohatdoc))

clunum<-rep(0,B)
purity<-rep(0,B)
nmi<-rep(0,B)
for (b in 1:B) {
  depkmean_doc<-kmeans(t(docsub),9)
depkmclu_doc<-depkmean_doc$cluster
accdepkm_doc<-as.matrix(table(depkmclu_doc,grouplabel))


# major types in each cluster
clunum[b]<-length(levels(as.factor(apply(accdepkm_doc,1,namefun))))


purity[b]<-sum(apply(accdepkm_doc, 1, max))/sum(accdepkm_doc)

nmi[b]<-NMI(depkmclu_doc,grouplabel)
}
# number of distinct types recognized
mean(clunum)
```

```
## [1] 7.5
```

```
# purity
mean(purity)
```

```
## [1] 0.4537963
```

```
# NMI
mean(nmi)
```

```
## [1] 0.3582335
```

```
# iid sub

iidsub_doc<-t(apply(cnaedata,1,subfuniid))

clunum<-rep(0,B)
purity<-rep(0,B)
nmi<-rep(0,B)

for (b in 1:B) {
  iidkmean_doc<-kmeans(t(iidsub_doc),9)
iidkmclu_doc<-iidkmean_doc$cluster
acciidkm_doc<-as.matrix(table(iidkmclu_doc,grouplabel))
acciidkm_doc

# major types in each cluster
clunum[b]<-length(levels(as.factor(apply(acciidkm_doc,1,namefun))))


purity[b]<-sum(apply(acciidkm_doc, 1, max))/sum(acciidkm_doc)

nmi[b]<-NMI(iidkmclu_doc,grouplabel)
}
# number of distinct types recognized
mean(clunum)
```

```
## [1] 7.6
```

```
# purity
mean(purity)
```

```
## [1] 0.4522222
```

```
# NMI
mean(nmi)
```

```
## [1] 0.3510795
```

We find that the estimated ρ in this dataset is close to 0, but the dependent case still outperforms the iid case, while behaving nearly as good as pure K-means in accuracy and better in recognizing more distinct types of cells.

Conclusion

Comparing with pure K-means methods based on raw data and Hamming distance clustering methods, the K-means based on dependent structure substitution methods slightly outperforms both in purity and nmi, with more distinct types of cells recognized. Further work may focus on improve the two step estimation algorithm of these parameters.

Appendix

We attach the result tables here.

Brain-cell data

- $\hat{\rho} = -0.4617678$
- K-Medoids results:

K-Medoids	distinct types	purity	nmi
Hamming distance	3	0.5047619	0.304276
iid similarity	6	0.7119048	0.485767
dependent similarity	6	0.7142857	0.517354

- K-means results:

K-Means	distinct types	purity	nmi
pure K-means	4.9	0.7783333	0.5956164
iid substitution	4.4	0.7635714	0.5639982
dependent substitution	4.9	0.7833333	0.6087817

- **Interpretation**:
  - $\hat{\rho} = -0.4617678$, suggesting non-zero correlation between genes. More importantly it also indicates that higher cell activity will lead to smaller X_{ij} and higher tendency of expression;
  - dependent case outperforms iid case in both algorithms;
  - K-medoids with our simularity recognize more distinct types of cells without losing much purity and nmi;
  - K-means with our dependent substitution has higher purity and nmi;
  - the result of Hamming distance metric substitution is not satisfying.

CNAE-9 dataset results

Similarly we apply these methods on this dataset.

- $\hat{\rho} = -0.0782741$: so we expect similar performance of iid and dependent case;
- our performance is second to the pure K-means based on raw data;

CNAE-9	distinct types	purity	nmi
K-medoids with Hamming distance	6	0.3907407	0.2963202
pure K-means	8.4	0.5425926	0.4497386
K-means with iid substitution	7.6	0.4522222	0.3510795
K-means with dependent substitution	7.5	0.4537963	0.3582335

- our clustering methods may work better with **large** sacle data that exhibits **correlation** in features;
- future work may focus on the estimation improvement.