Homework6

Jun Rao

9/14/2020

Model selection for clustering.

Model selection means selecting the clustering parameters which are best for a given data set (without using the labels). The goal of this homework is to compute and plot model selection criteria for clustering.

1.Split the zip.test data observations/rows into 50% train, 50% validation. Use base::set.seed and base::sample for pseudo-random assignment with a fixed seed. Use base::table(digit.label, set) to print and display a contingency table. Are there about the same number of each digit/label in each set?

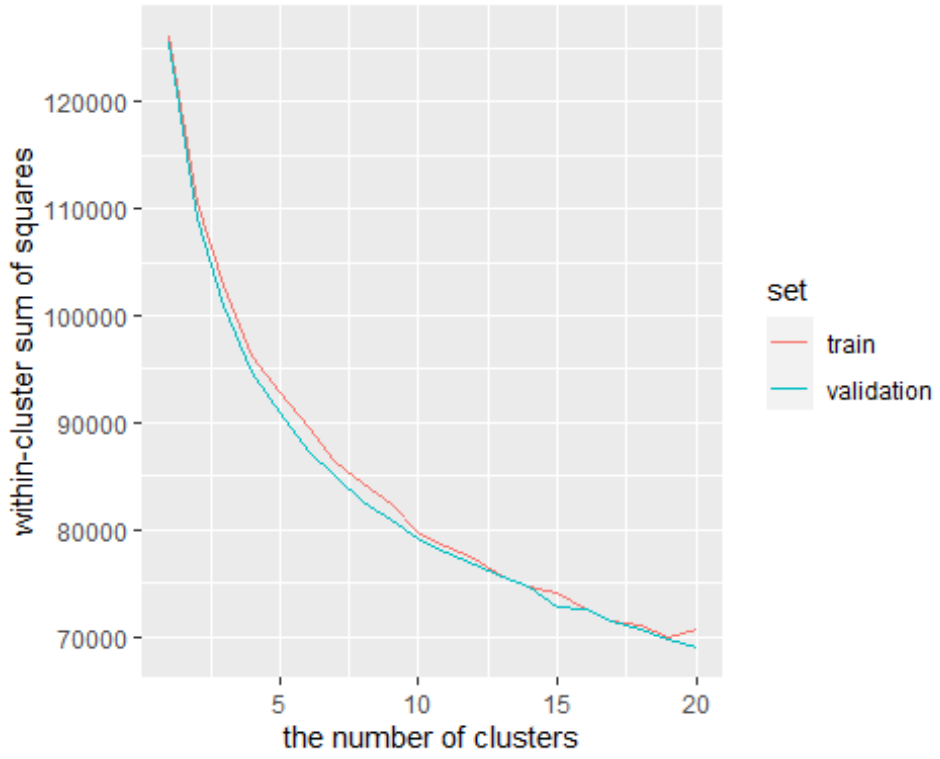
```
set.prop.vec <- c(validation=0.5, train=0.5)
N <- nrow(dt)
rounded.counts <- floor(set.prop.vec*(N))
not.shuffled.sets <- rep(names(set.prop.vec), rounded.counts)
set.seed(1)
shuffled.sets <- sample(not.shuffled.sets)
table(set=shuffled.sets, label=dt[2:2007]$V1)
```

```
##             label
## set            0   1   2   3   4   5   6   7   8   9
##   train      160 136 109  82 107  84  86  75  83  81
##   validation 199 128  89  84  93  76  84  72  83  95
```

2.Use stats::kmeans on the train data, for 1 to 20 clusters. For each number of clusters, and for each set (train/validation), compute the within-cluster sum of squares (to do that you need to first compute the closest cluster center to each observation, using the "centers" component of the list returned by kmeans). You can check your work by comparing your value with "tot.withinss" (they should be the same). Plot y=sum of squares as a function of x=number of clusters, with a different colored line for each set (e.g. train=red, validation=black). Do both of your lines decrease as expected?

```
clusters <- 20
result.dt.list <- list()


for(n.clusters in 1:clusters){
    for (set in names(set.prop.vec)) {
        data.set <- df[shuffled.sets == set,]
        kcluster <- kmeans(data.set,n.clusters,iter.max = clusters)
        SOS = kcluster$tot.withinss
        result.dt.list[[paste(n.clusters, SOS, set)]] <- data.table(n.cluster
```

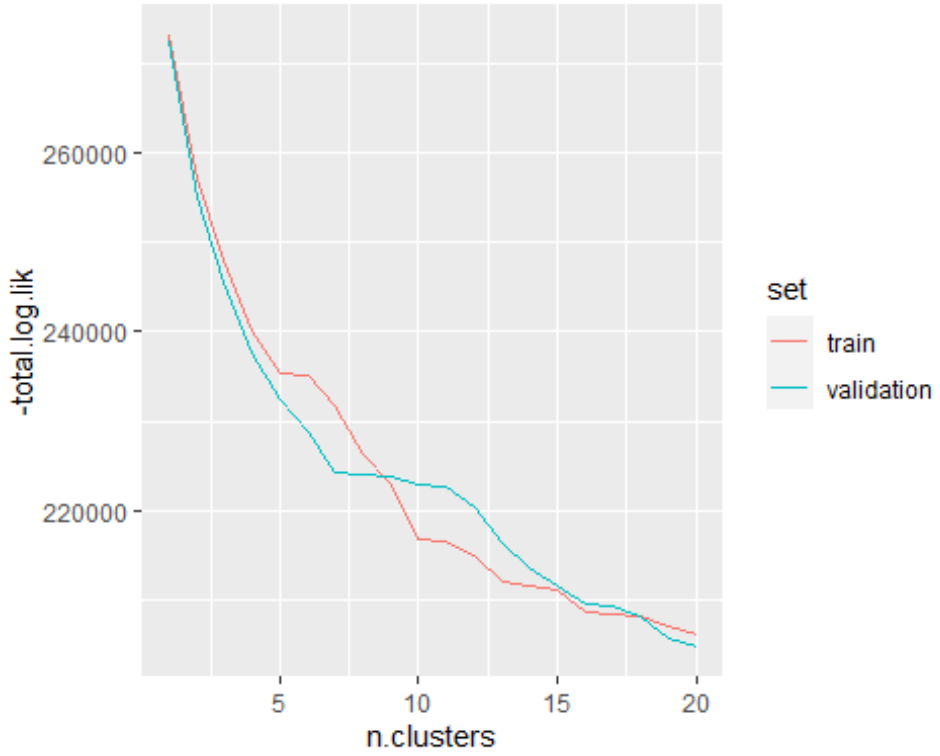
```
s, SOS,set)
    }

}


result.dt <- do.call(rbind, result.dt.list)

ggplot()+
  geom_line(aes(n.clusters, SOS, color=set),data=result.dt) +
  labs(x = "the number of clusters", y ="within-cluster sum of squares")
```



3.Use mclust::Mclust on the train data, for 1 to 20 clusters, and for a single value of modelNames (e.g. VEV for ellipsoidal, equal shape). Remember you can use a subset of data for the initialization to reduce computation time. Use mclust::dens to compute a vector of log likelihood values, one for each observation (in both train/validation sets). Compute and plot y=mean negative log likelihood as a function of x=number of clusters, using different colored lines for different sets (e.g. train=red, validation=black). Does the train negative log likelihood always decrease as expected? Does the validation negative log likelihood decrease and then increase (U shape) as expected?

```
clusters <- 20

lik.dt.list <- list()
```

```
for(n.clusters in 1:clusters){
    for (set in names(set.prop.vec)) {
        data.set <- df[shuffled.sets == set,]
        mclust <- Mclust(data.set, n.clusters, initialization=list(subset=1:5
00),verbose = 0, modelNames="EII")
        log.lik.vec <- dens(
                            modelName = mclust[["modelName"]],
                            data = data.set,
                            parameters = mclust[["parameters"]],
                            logarithm=TRUE)
       total.log.lik <- sum(log.lik.vec)
       rbind(my=total.log.lik, mclust=mclust[["loglik"]])
       lik.dt.list[[paste(n.clusters, set)]] <- data.table(n.clusters, set, t
otal.log.lik)
    }

}


lik.dt <- do.call(rbind, lik.dt.list)

ggplot()+
  geom_line(aes(
    n.clusters, -total.log.lik, color=set),
    data=lik.dt)
```



The goal is to implement and plot the gap statistic described in ESL-14.3.11, using kmeans and the whole zip.test data. The main idea is that if there are real clusters in the data, then the kmeans sum of squares on the zip.test data should decrease more rapidly than the kmeans sum of squares on random uniform data. 1.You will need to use the stats::runif function to generate uniform random numbers between -1 and 1, in a matrix the same size as the zip.test data matrix you cluster.

```
generate_uniform_numbers = function(data) {

    r = nrow(data)
    c = ncol(data)

    return(matrix(runif(r*c, min = -1, max = 1),r,c))

}
```

2.Generate 20 random matrices, as they did in ESL Figure 14.11.

```
generate_uniform_matrices = function(data,N) {
    m.list <- list()

    for (i in 1:N) {

         m.list[[i]] <- generate_uniform_numbers(data)

    }

    return(m.list)

}
```

3.Compute kmeans for 1 to 20 clusters, for each of the 21 data sets (zip.test + 20 random uniform). For clarity use a for loop over a list with 21 elements.

```
kclu_matrices = function(data, clusters,N) {
  m.list <- generate_uniform_matrices(data,N)

    kclu.list <- list()

  for (i in 1:N) {
      temp <- c()
      for (n.clusters in 1:clusters) {
           kclu<- kmeans(m.list[[i]],n.clusters,iter.max = clusters)
           temp <- c(temp,kclu$tot.withinss)
      }
      kclu.list[[i]] <- temp
  }
```

```
  return(kclu.list)
}
```

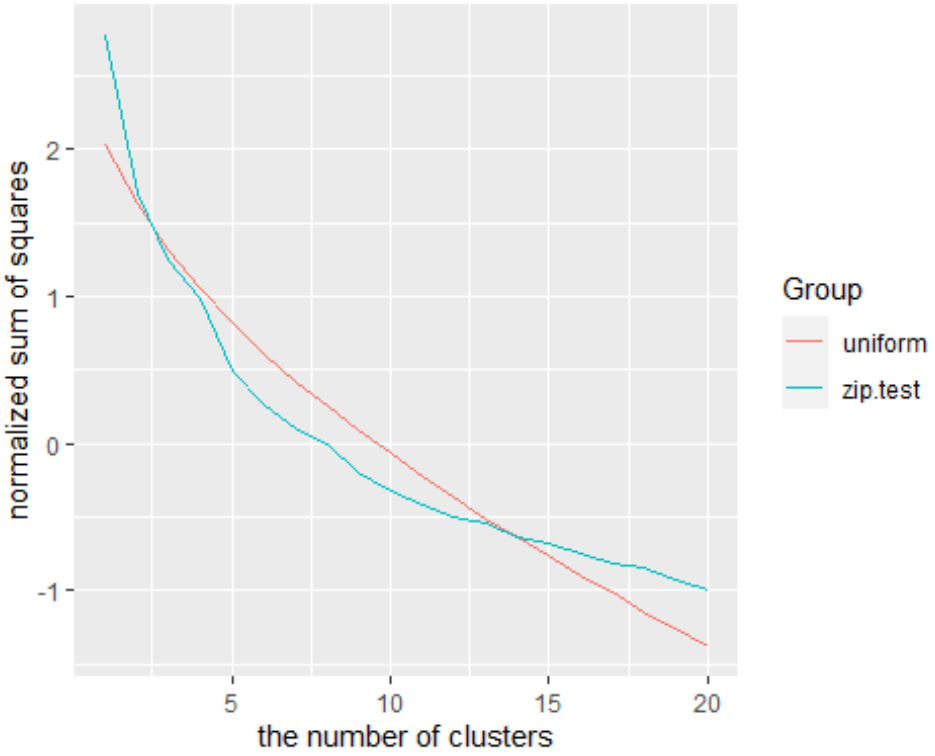
normalized function

```
normalized <- function(data){
      data <- (data-mean(data))/sd(data)


   return(data)
}
```

4. The first plot is y=normalized sum of squares as a function of x=number of clusters, zip.test line in green, mean uniform random line in blue. The second plot is y=gap statistic with error bars/bands as a function of x=number of clusters (use geom_line and geom_ribbon).Compute the quantities shown in ESL Figure 14.11, and make two ggplots.

```
kclu.list <- kclu_matrices(df,20,20)

zip.test.SOS <- rep(0,20)
for (index in 1:20) {
   clu<-kmeans(df, index)
   zip.test.SOS[index] <- clu$tot.withinss
}

normalized.zip.test.SOS <- normalized(zip.test.SOS)

total.SOS <- rep(0,20)
for (index in 1:20) {
   total.SOS= total.SOS + kclu.list[[index]]
}

uniform.SOS <- total.SOS/20

normalized.uniform.SOS <- normalized(uniform.SOS)



SOS.list  <- list()

for (index in 1:20){
    clu<-kmeans(df, index)
    SOS.list[[index]] = data.table("SOSE" = normalized.zip.test.SOS[index], "
Ncluster" = index, "Group" = "zip.test")
}


for (index in 21:40){

    SOS.list[[index]] = data.table("SOSE" = normalized.uniform.SOS[index-20],
```

```
 "Ncluster" = index-20, "Group" = "uniform")
}

SOS.dt <- do.call(rbind,SOS.list)


ggplot()+
  geom_line(aes(Ncluster,SOSE, color=Group),data= SOS.dt) +
  labs(x = "the number of clusters", y ="normalized sum of squares")
```



The second plot is y=gap statistic with error bars/bands as a function of x=number of clusters (use geom_line and geom_ribbon).

```
normalized.zip.test.gap <- mean(normalized.zip.test.SOS) - normalized.zip.tes
t.SOS
normalized.zip.test.sd <- sd(normalized.zip.test.SOS)/1.1
normalized.uniform.gap <- mean(normalized.uniform.SOS) - normalized.uniform.S
OS
normalized.uniform.sd <- sd(normalized.uniform.SOS)/1.1


gap.data.dt <- list()

for (index in 1:20){

    gap.data.dt[[index]] = data.table("GAP" = normalized.zip.test.gap[index],
```

```
"sd"=normalized.zip.test.sd, "Ncluster" = index, "Group" = "zip.test")
}

for (index in 21:40){

    gap.data.dt[[index]] = data.table("GAP" = normalized.uniform.gap[index-20
],"sd"=normalized.uniform.sd, "Ncluster" = index-20, "Group" = "uniform")
}

gap.dt <- do.call(rbind,gap.data.dt)



ggplot(gap.dt) +
  geom_errorbar(aes(x=Ncluster, ymin = GAP-sd, ymax = GAP+sd,color=Group))+
  geom_line(aes(Ncluster,GAP,color=Group))  +
  labs(x = "the number of clusters", y ="GAP")
```