Section 4: CH Index and K-Means++

Estevan Gonzales, Erik Honore, Sean Kessel

August 3, 2016

Review on basic K-means - opening questions from previous class (Module 7)

K-means is considered a 'first crack' at clustering in that it would be the simplest 'go-to' option to start a clustering problem. This is a good method to use as a first pass over a data set, and from there you may consider other more sophisticated methods for clustering.

Q: "K-Means seems to work best with a small number of observations, what about large?"
A: K-Means has a reputation for speed, every iteration has complexity that is linear with the number of data points and number of means you're asking (per iteration)

The difficulty of finding global minimizers in K-means

The metric we use to evaluate a basic k-means clustering scheme is SSE (sum of squared errors):

$$\sum_{j=1}^{k} \sum i : a_i = j(\, x_i - \mu_j)^2$$

We want to minimize this function, but getting to the global minimum is really difficult.

How do we go about picking the best k, or number of clusters in order to minimize SSE?

Consider this:

Define $A(n, k)$ to equal number of ways to divide n points into k groups.

Note how:

$A(10,4)$ = 34105

$A(25,4)$ = $\sim 5 * 10^{13}$

As shown, this function grows rapidly and becomes unwieldy. Clearly, we can see that trying to find the absolute optimal number for picking k and grouping the points correctly to minimize SSE is tough. The amount of combinations for 10 data points into 4 clusters is 34105 as shown above. The computational complexity is massive for this problem.

Solution:

Two assessments for a k-means implementation

1. Fit

- A highly fitted model has a low sum of squared errors (SSE).
  - $SSE = \sum_{j=1}^{k} \sum \quad i: a_i = j(x_i - \mu_j)^2$
- Note that for K-Means, the model can always be fitted better by adding another cluster.
  - This produces a model that is more specified, but not necessarily better. The model can be over-fitted. In the extreme case, every data point is basically a cluster with itself as the center.

2. Simplicity

- A simple model has fewer groups than a complex model. A simpler model may also have more interpretability than the complex one.

CH Index: The Occam's Razor for K-Means

We use a metric that seeks to balance Fit and Simplicity for an optimal k. That will be the CH Index.

The Yin and the Yang

- High k leads to high fit.
- Low k leads to high simplicity.
- We need to find balance.

The CH Index Method

For k as the number of centers.

Let's define $w(k)$ to be the sum of the squares of the points (x_i) within the group to the groups mean(μ_i)

Note, $w(k) = \sum_{j=1}^{k} \quad \sum \quad i: a_i = jd\,(x_i, \mu_j)^2$. Where $d()$ is the Euclidean distance between two points.

Lets define $B(k)$ to equal the sum of square of distance clusters are from the mean of the entire sample, weighted by number of points in each sample.

Note, $B(k) = \sum_{j=1}^{k} \quad n_j * d(m_j, \overline{x})$ where n_j is the number of points in the cluster, m_j is the location of the cluster, and $\overline{x}$ is the overall sample mean.

Finally, lets define a $CHIndex$ by the equation: $CH(k) = \frac{B(k)/(k-1)}{w(k)/(n-k)} = \frac{B(k)(n-k)}{w(k)(k-1)}$

The idea is to choose a k that maximizes $CH(k)$.

In other words, find $\hat{k}$ such that $\hat{k} = argmax_{k \in 2,3,\dots} CH(k)$

Ideally, we want $B(k)$ to be bigger, and $w(k)$ to be smaller. $B(k)$ and $w(k)$ could be thought of as simplicity bonuses whereas $(n-k)$ and $(k-1)$ can be thought of as complexity penalties.

Initializing K-Means via K-Means++

The Method

Given a set of points indexed by i:

1. Choose 1 center at random, m_1

2. Compute d_i, where d_i = minimum distance of x_i to any existing cluster center.
- Repeat this step for $j = 2, \ldots, k$.
3. Choose a new center with probability proportional to d_i^2.
- This creates a probability distribution that favors further points.
  - Note that this is still a random probability. A point very far from a center will have a higher probability of being selected as a new center.
- Go back to step 2 to repeat.

Question: Why not simply choose the furthest point?

Answer: Choosing the furthest point would not be optimal for K-Means.

Question: After the first iteration when the first center is created, there will be multiple distances from each data point to multiple centers as the iterative process continues. Which distance is used in K-Means ++ ?

Answer: The distance that is shortest would be selected. Say we have centers μ_1 and μ_2 and we are determining the distance for x_3, then $d_3 = min(d(x_3, \mu_1), d(x_3, \mu_2))$.

K-Means++ Example in we8there.r Script

Load the necessary libraries

library(wordcloud) is for visualizations.

library(textir) contains functions for fast sparse multinomial logistic regression for phrase counts.

library(flexclust) contains have K-means ++ routine for initialization.

Script High Points

- This data set contains text data from restaurant reviews. We are interested in clustering the reviews to draw thematic insights from them. We need to first construct a term matrix, where the values are counts for the number of times a phrase or word was used. Then we need to scale and normalize it.

- The K-Means++ method is contained cclust() function in flexclust package
  - control=list(initcent="kmeanspp") is the parameter to use K-means++
- The model scheme for K-Means++ uses different access syntax than the regular k-means
  - "model@clusinfo" method Shows the size average distance, maximum distance, and separation metrics for all of the clusters.

Annotated we8there.r

```
## Loading required package: RColorBrewer

## Loading required package: distrom

## Loading required package: Matrix

## Loading required package: gamlr

## Loading required package: parallel

## Loading required package: grid

## Loading required package: lattice

## Loading required package: modeltools

## Loading required package: stats4

#not sure if this one is necessary

data(we8there) #a set with text data from restaurant reviews
#interested in clustering reviews to see thematics in the text
#document term matrix, terms are phrases words or characters, entries are
counts


#normalize and scale the term matrix
X_freq = we8thereCounts/rowSums(we8thereCounts)
Z = scale(X_freq)

# Run k means
kmeans_we8there <- kmeans(Z, 4)  #z score matrix from above, with 4 cluster
centers

# The first centroid
head(sort(kmeans_we8there$centers[1,], decreasing=TRUE), 10) #last argument
means it will do 10 random restarts in order to find the best clustering set
that it can
```

```
##   food great   great food  great place  food servic   food excel
##   0.04331350   0.04144518   0.03997633   0.03826684   0.03709514
## great servic servic great    place eat  veri reason    good food
##   0.03642959   0.03599196   0.03523063   0.03283218   0.03251269

# Prints all the centroids for each of the 10 restarts
print(apply(kmeans_we8there$centers,1,function(x) colnames(Z)[order(x,
decreasing=TRUE)[1:10]]))

##       1               2                3                4
##  [1,] "food great"    "out bag"        "main cours"     "serv here"
##  [2,] "great food"    "just over"      "san francisco" "after read"
##  [3,] "great place"   "sandwich came" "best kept"      "get order"
##  [4,] "food servic"   "waitress came" "worth everi"    "now know"
##  [5,] "food excel"    "came over"      "blue chees"     "sandwich burger"
##  [6,] "great servic" "right away"     "crab cake"      "food network"
##  [7,] "servic great" "took order"     "kept secret"    "stir fri"
##  [8,] "place eat"     "came back"      "open daili"     "onli serv"
##  [9,] "veri reason"   "minut befor"    "larg enough"    "went breakfast"
## [10,] "good food"     "minut later"    "melt mouth"     "again again"

#good for rapidly understanding what the data is saying

#A word cloud - usage of the wordcloud package to see which words are most
significant

wordcloud(colnames(Z), kmeans_we8there$centers[2,], min.freq=0,
max.words=100)

#This block is basically accessing the different metrics for the cluster

# The different sums of squares: not great
kmeans_we8there$totss   # total sums of squares

## [1] 16275600

kmeans_we8there$withinss  # sums of squares within each cluster

## [1] 11147760  1844398  2818796   422751

kmeans_we8there$tot.withinss # sum of sums of squares of all clusters

## [1] 16233704

kmeans_we8there$betweenss  # sum of square distances of the cluster centers

## [1] 41895.8

# Sums of squares obey the "variance decomposition"
# aka Pythagorean theorem of statistics
kmeans_we8there$betweenss + kmeans_we8there$tot.withinss

## [1] 16275600
```

```
kmeans_we8there$totss

## [1] 16275600

# USING Kmeans++ for initialization of the clustering

kmeansPP_we8there = cclust(Z, k=4, control=list(initcent="kmeanspp"))
#cclust() is in the flexclust package
#This line is asking for 4 centers, using kmeans++ (kmeanspp initialization
scheme)


#less intuitive way to access summary stats with kmeans++
# This package has a different interface for accessing model output

# parameters(kmeansPP_we8there)

kmeansPP_we8there@clusinfo #shows a variety of metrics associated the
clustering result

##   size  av_dist  max_dist separation
## 1    5 55.69458  76.44172   48.02404
## 2 6146 49.31027 116.66151   29.22352
## 3    9 39.82740  66.68002   35.73917
## 4    6 38.27795  61.58663   40.80632

print(apply(parameters(kmeansPP_we8there),1,function(x) colnames(Z)[order(x,
decreasing=TRUE)[1:10]]))

##       [,1]            [,2]            [,3]                [,4]
##  [1,] "first thing"   "go back"       "chef owner"        "excel veri"
##  [2,] "attent staff"  "food great"    "wonder food"       "go food"
##  [3,] "time time"     "great place"   "never got"         "atmospher excel"
##  [4,] "littl extra"   "great food"    "look more"         "san diego"
##  [5,] "eat good"      "veri friend"   "restaur owner"     "fair price"
##  [6,] "cream chees"   "servic great"  "menu full"         "whip cream"
##  [7,] "thing notic"   "realli good"   "servic except"     "servic wonder"
##  [8,] "menu consist"  "dine room"     "amaz food"         "food realli"
##  [9,] "experi food"   "reason price"  "small restaur"     "realli enjoy"
## [10,] "walk door"     "veri nice"     "recommend friend"  "everi week"

#Roll our own function
#This function pulls the necessary information do a kmeans++ sum of squared
errors calculation
centers = parameters(kmeansPP_we8there)
kpp_residualss = foreach(i=1:nrow(Z), .combine='c') %do% {
    x = Z[i,]
    a = kmeansPP_we8there@cluster[i]
    m = centers[a,]
    sum((x-m)^2)
}
```

```r
sum(kpp_residualss)
kmeans_we8there$tot.withinss
```