Clustering

Rui Wang

10/19/2018

K-Means

Theory

- $\text{cost}(cluster_1, cluster_2, ..., cluster_k, c_1, ..., c_k) = \sum_k \sum_{i:x_i \in cluster_k} dist(x_i, c_k)$
- Input:number of clusters K, randomly initialize center c_k
- Until converged: Assign each point to the cloest cluster center:

  $$\min_{cluster_1,...,cluster_k} \text{cost}(cluster_1, cluster_2, ..., cluster_k, c_1, ..., c_k)$$

  Change each cluster center to be in the middle of its point:

  $$\min_{c1,...,c_k} \text{cost}(cluster_1, cluster_2, ..., cluster_k, c_1, ..., c_k)$$

- Pros and Cons:
  - Computationally efficient
  - Can use cost function to choose the number of clusters
  - Does not always fully minimize the cost function
  - Does not work well for highly non-spherical data

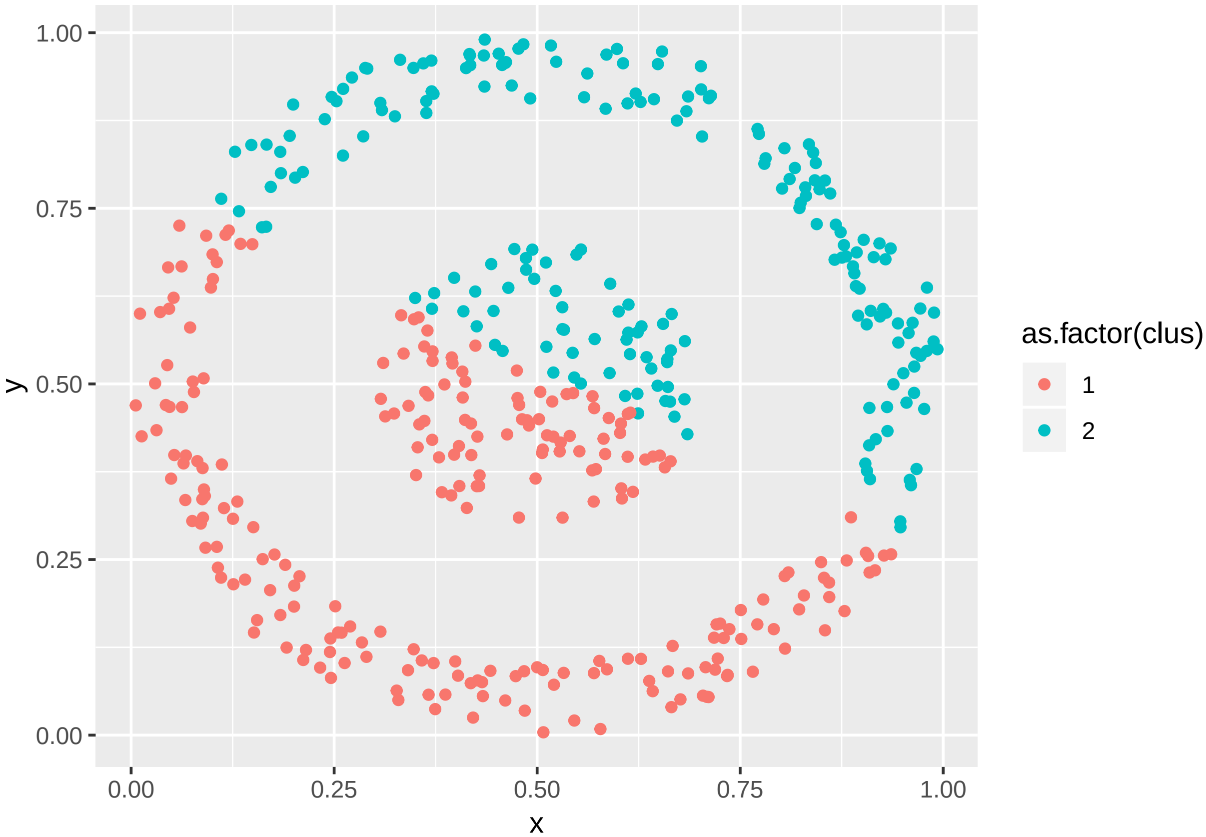
Algorithm

```
k_means = function(data,nclus){
  N = nrow(data)
  data = data %>% mutate(clus = rep(0,N))
  center = sample(N,nclus,replace = F)
  xcen = data[center,1]
  ycen = data[center,2]
  cluster = data.frame(k = 1:nclus,xcen,ycen)
  stop = FALSE
  while(stop == FALSE){
    for (i in 1:N){
      dist = sqrt((data$x[i]-cluster$xcen)^2 + (data$y[i]-cluster$ycen)^2)
      data$clus[i] = which.min(dist)
    }

    xcen_old = cluster$xcen
    ycen_old = cluster$ycen

    for (i in 1:nclus){
      cluster[i,"xcen"] = mean(subset(data$x, data$clus == i))
      cluster[i,"ycen"] = mean(subset(data$y, data$clus == i))
    }

    if(identical(xcen_old, cluster$xcen) & identical(ycen_old, cluster$ycen))
      stop = TRUE
```

```
  }
  return(list(data=data,cluster=cluster))
}
```

Implementation

```
circle = read.csv("data.csv")
colnames(circle) = c("x","y")
circle_k = k_means(circle,2)
ggplot(circle_k$data,aes(x=x,y=y,colour = as.factor(clus)))+
  geom_point()
```



Hierarchical Agglomerative Clustering

Theory

- start with each point in its own cluster
- repeatedly merge the clusters of the cloest two points(choose when to stop merging clusters)
- Useful when clusters are well-separated.

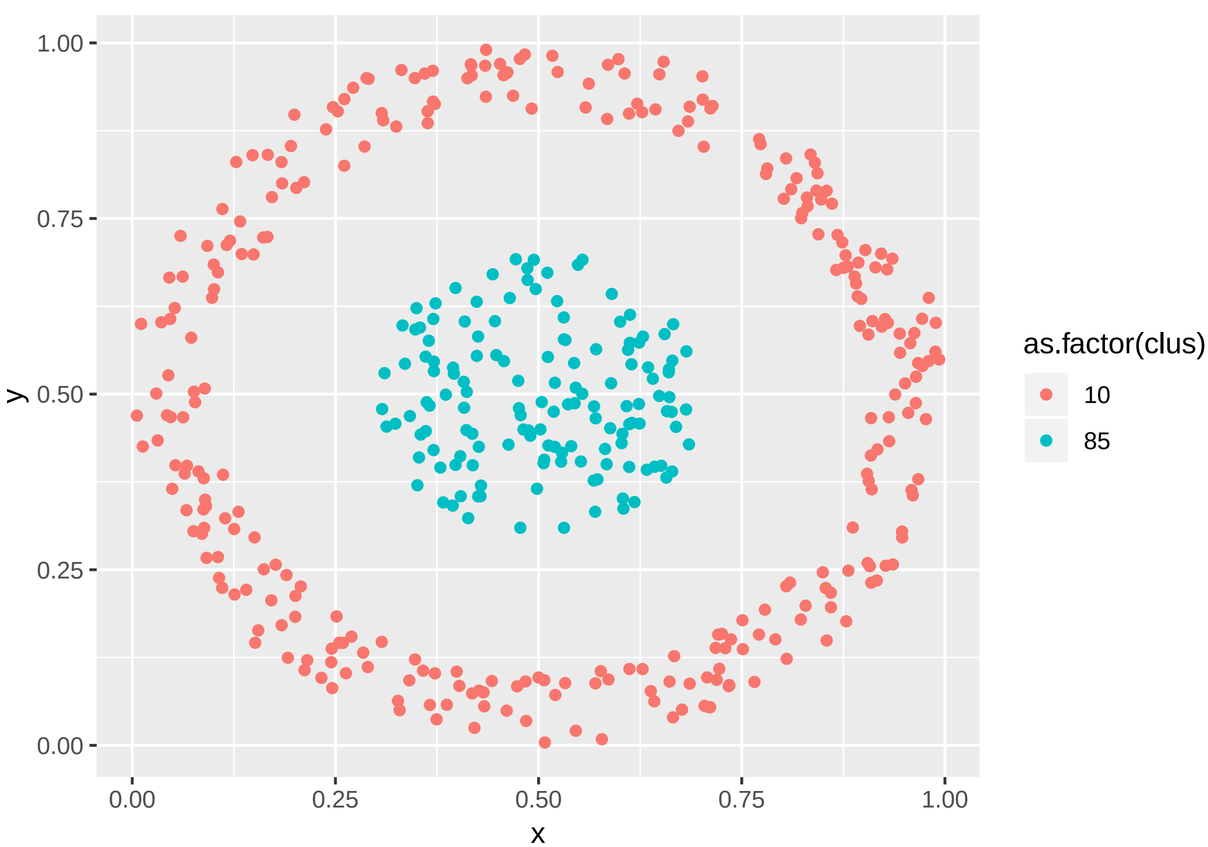
Algorithm

```
hac = function(data,nclus){
  d = as.matrix(dist(data))
  d[lower.tri(d)] = Inf
  diag(d)=Inf
```

```
  N = nrow(data)
  clus = -(1:N)
  k = 0
  while(length(unique(clus))>nclus){
    h = min(d)
    i = which(d - h == 0, arr.ind=TRUE)
    i = i[1,,drop=FALSE]
    d[i] = Inf
    if (clus[i[1]]<0&clus[i[2]]<0){
      k = k+1
      clus[i[1]]=clus[i[2]]=k
    }else{
      cluster_keep =  clus[i][clus[i]>0][1] #record one cluster
      cluster_delete = clus[i][clus[i]!=cluster_keep]
      clus[clus==cluster_delete] = cluster_keep
    }
  }
  return(clus)
}
```

Implementation

```
circle_hac = hac(circle,2)
circle_hclust = cbind(circle,clus = circle_hac)
ggplot(circle_hclust,aes(x=x,y=y,colour = as.factor(clus)))+
  geom_point()
```



- For this dataset, hierarchical agglomerative clustering performs better. The possible reasons are that hierarchical agglomerative clustering performs better when clusters are well seperated while K means works well for spherical data. In order to boost the performance of the weaker algorithm, which is k-means in our case, we can map the original dataset into a new feature space where k-means algorithm works well.