K-means Clustering

Jim Harner

5/31/2020

7.2 K-means Clustering

7.2.1 K-means Basics

Linear regression and logistic regression are *supervised* learning methods. By that we mean that the the values of the outcome variable, e.g., the labels, are known, at least for the training set. k-means is an *unsupervised* learning methods. By that we mean that the values of the outcome variable are unknown or that there is no outcome variable, e.g., the labels are unknown.

The objective of k-means is to group or cluster similar objects together. For example, suppose you have users and you know the age, gender, income, state, and household size for each user. We then want to segment, stratify, group, of cluster the data. You may or may not have an *a priori* notion concerning the number of groups k.

The k-means algorithm works as follows:

1. Initially, you randomly pick k centroids (or points that will be the center of your clusters) in d-space. Try to make them near the data but different from one another.

2. Then assign each data point to the closest centroid.

3. Move the centroids to the average location of the data points (which correspond to users in this example) assigned to it.

4. Repeat the preceding two steps until the assignments don't change, or change very little.

More formally, let $x_1, x_2, \cdots, x_n$ be the n observed data points. Let $s_1, s_2, \cdots, s_k$ be initial seed points—perhaps chosen randomly.The seed points form the nuclei of the clusters $C_1, C_2, \cdots, C_k$. The data point x_i is put into cluster C_j if

$$\|x_i - s_j\| = \min_{a=1,\cdots,k} \|x_i - s_a\|,$$

i.e., if x_i is closest to the j^{th} seed point. At the end of the first step, we have k clusters: $C_1, C_2, \cdots, C_k$. It is possible that some clusters are empty and thus there can be fewer than k clusters. The choice of the initial seed points is critical in determining clusterings not only in the first stage, but also in the final stage. Hierarchical clustering is often used to get the initial seed points, but other choices are possible.

For each cluster, e.g., C_r, compute the cluster centroid $\bar{x}_r$. The $\bar{x}_r$ become the new seed points and the observations are formed into clusters using the above spherical (Euclidean) distances. This process is iterated until the cluster means do not change.

8.2.2 K-means on the State Crime Data

Read in the crime data for the 50 states:

```
state_crime_df <- read_csv(
  "/home/rstudio/rspark-tutorial/data/state_crime.csv")
```

```
## Parsed with column specification:
## cols(
```

```
##   State = col_character(),
##   Abbr = col_character(),
##   Division = col_character(),
##   Region = col_character(),
##   Murder = col_double(),
##   Rape = col_double(),
##   Robbery = col_double(),
##   Assault = col_double(),
##   Burglary = col_double(),
##   Larceny = col_double(),
##   Auto = col_double(),
##   Unemploy = col_double(),
##   Police = col_double(),
##   InSchool = col_double()
## )
```

We select the variables of interest and standardize them as in Section 8.1.2.

```
state_crime_std_df <- state_crime_df %>%
  select(-State, -Abbr, -Division, -Region, -Unemploy, -Police, -InSchool) %>%
  lapply(function(e) scale(e)) %>%
  as.data.frame()
```

Our objective is to compute the PCA scores on the standardized variables for the first two components. Once we perform k-means clustering we plot the points in this 2-dimensional PCA space.

```
state_crime_pca <- state_crime_std_df %>%
  princomp()
summary(state_crime_pca)
```

```
## Importance of components:
##                           Comp.1    Comp.2     Comp.3     Comp.4     Comp.5
## Standard deviation     2.0056558 1.0360906 0.82097344 0.71310558 0.49365277
## Proportion of Variance 0.5863929 0.1564845 0.09825035 0.07412822 0.03552377
## Cumulative Proportion  0.5863929 0.7428774 0.84112776 0.91525597 0.95077974
##                            Comp.6     Comp.7
## Standard deviation     0.48374626 0.32193249
## Proportion of Variance 0.03411231 0.01510795
## Cumulative Proportion  0.98489205 1.00000000
```

The PCA loadings and scores (projections into PCA space) are given by:

```
state_crime_pca$loadings
```

```
##
## Loadings:
##          Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7
## Murder    0.392  0.259  0.410  0.423  0.433  0.350  0.350
## Rape      0.288 -0.476  0.599 -0.556         0.113
## Robbery   0.404  0.429 -0.138 -0.239  0.386 -0.165 -0.633
## Assault   0.435         0.196  0.222 -0.346 -0.757  0.168
## Burglary  0.420 -0.223 -0.171  0.393 -0.489  0.433 -0.405
## Larceny   0.291 -0.617 -0.493         0.480 -0.154  0.179
## Auto      0.388  0.299 -0.379 -0.495 -0.272  0.231  0.493
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000  1.000
```

```
## Proportion Var  0.143  0.143  0.143  0.143  0.143  0.143  0.143
## Cumulative Var  0.143  0.286  0.429  0.571  0.714  0.857  1.000
```

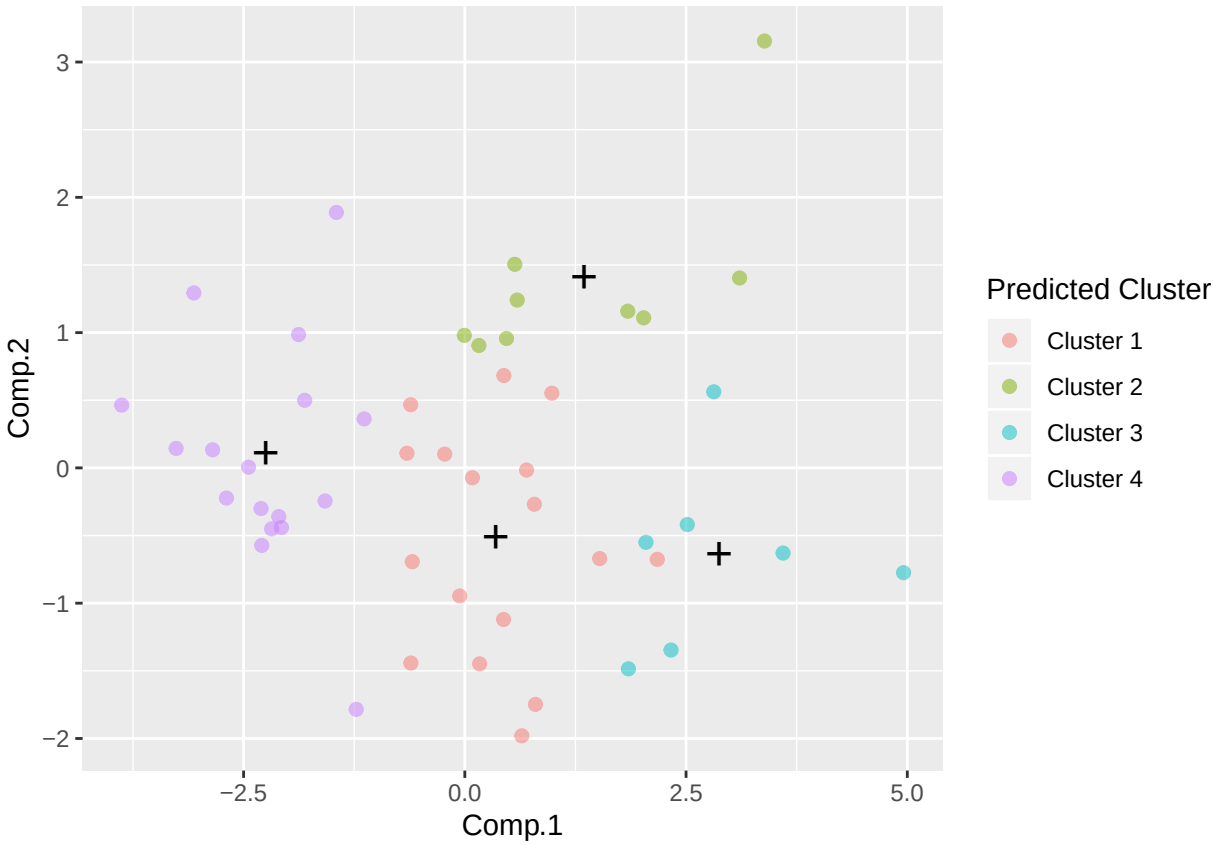
```
state_crime_pca_scores <- predict(state_crime_pca) %>%
  as.data.frame()
```

The clusters are now computed and identified by projecting into the first two PCA components.

```
state_crime_kmeans <- kmeans(state_crime_std_df, centers = 4)
state_crime_pca_centers <- predict(state_crime_pca, state_crime_kmeans$centers) %>%
  as.data.frame()
state_crime_pca_centers
```

```
##       Comp.1      Comp.2     Comp.3      Comp.4      Comp.5      Comp.6
## 1  0.3521886 -0.53947200  0.4619079 -0.16153591 -0.03679657  0.15271063
## 2  1.3483402  1.37924888 -0.5387322 -0.44026297 -0.11979458 -0.18239953
## 3  2.8724444 -0.66313870 -0.1587708  0.72671362 -0.09914512  0.01527615
## 4 -2.2487870  0.08233853 -0.1113205  0.09538128  0.14104169 -0.06243635
##         Comp.7
## 1 -0.052807038
## 2  0.020940712
## 3  0.088805512
## 4  0.005153803
```

```
state_crime_pca_scores %>%
  select(Comp.1, Comp.2) %>%
  ggplot(aes(Comp.1, Comp.2)) +
  geom_point(aes(Comp.1, Comp.2, col = factor(state_crime_kmeans$cluster)),
             size = 2, alpha = 0.5) +
  geom_point(data = state_crime_pca_centers[, 1:2], aes(Comp.1, Comp.2),
             pch = '+', size = 6) +
  scale_color_discrete(name = "Predicted Cluster", labels = paste("Cluster", 1:4))
```



7.2.3 Spark K-means on the State Crime Data

Load `state_crime.csv` into Spark with `spark_read_csv` from the local filesystem.

```
state_crime_sdf <- spark_read_csv(sc, "state_crime_sdf",
    path =  "file:///home/rstudio/rspark-tutorial/data/state_crime.csv")
```

The crime rates per 100,000 are extracted and scaled for each state.

```
state_crime_std_sdf <- state_crime_sdf %>%
  select(-State, -Abbr, -Division, -Region, -Unemploy, -Police, -InSchool) %>%
  spark_apply(function(e) scale(e))
class(state_crime_std_sdf)
```

```
## [1] "tbl_spark" "tbl_sql"   "tbl_lazy"  "tbl"
```

The Spark K-means clustering is performed. The k-means centers are computed in the original feature space.

```
state_crime_kmeans_model <- state_crime_std_sdf %>%
  ml_kmeans(~ Murder + Rape + Robbery + Assault + Burglary + Larceny + Auto,
            k = 4L)
```

```
## Warning in spark_param_deprecated("compute_cost"): The 'compute_cost' parameter
## is deprecated in Spark 3.x

## Warning in spark_param_deprecated("compute_cost"): The 'compute_cost' parameter
## is deprecated in Spark 3.x
```

```
class(state_crime_kmeans_model)
```

```
## [1] "ml_model_kmeans"     "ml_model_clustering" "ml_model"
```

```
state_crime_kmeans_model$center
```

```
##       Murder       Rape    Robbery     Assault   Burglary    Larceny
## 1  1.1883723  0.6379006  0.6274535  1.38760952  1.6859339  1.3426086
## 2 -0.6848691 -0.7922050 -0.8054411 -0.97145866 -0.9423250 -0.7542186
## 3  0.5254655 -0.2867302  0.9784305  0.64914728  0.2517314 -0.3022851
## 4 -0.2129584  0.8887817 -0.1743458 -0.07058834  0.0855151  0.5036341
##          Auto
## 1  0.69187098
## 2 -0.91085053
## 3  0.95145304
## 4 -0.05543388
```

The predicted group memberships are computed.

```
state_crime_kmeans_predict_sdf <- state_crime_std_sdf %>%
  sdf_predict(state_crime_kmeans_model) %>%
  select(prediction)
```

```
## Warning: 'sdf_predict' is deprecated.
## Use 'ml_predict' instead.
## See help("Deprecated")
```

```
state_crime_kmeans_predict_df <- state_crime_kmeans_predict_sdf %>%
  collect()
```

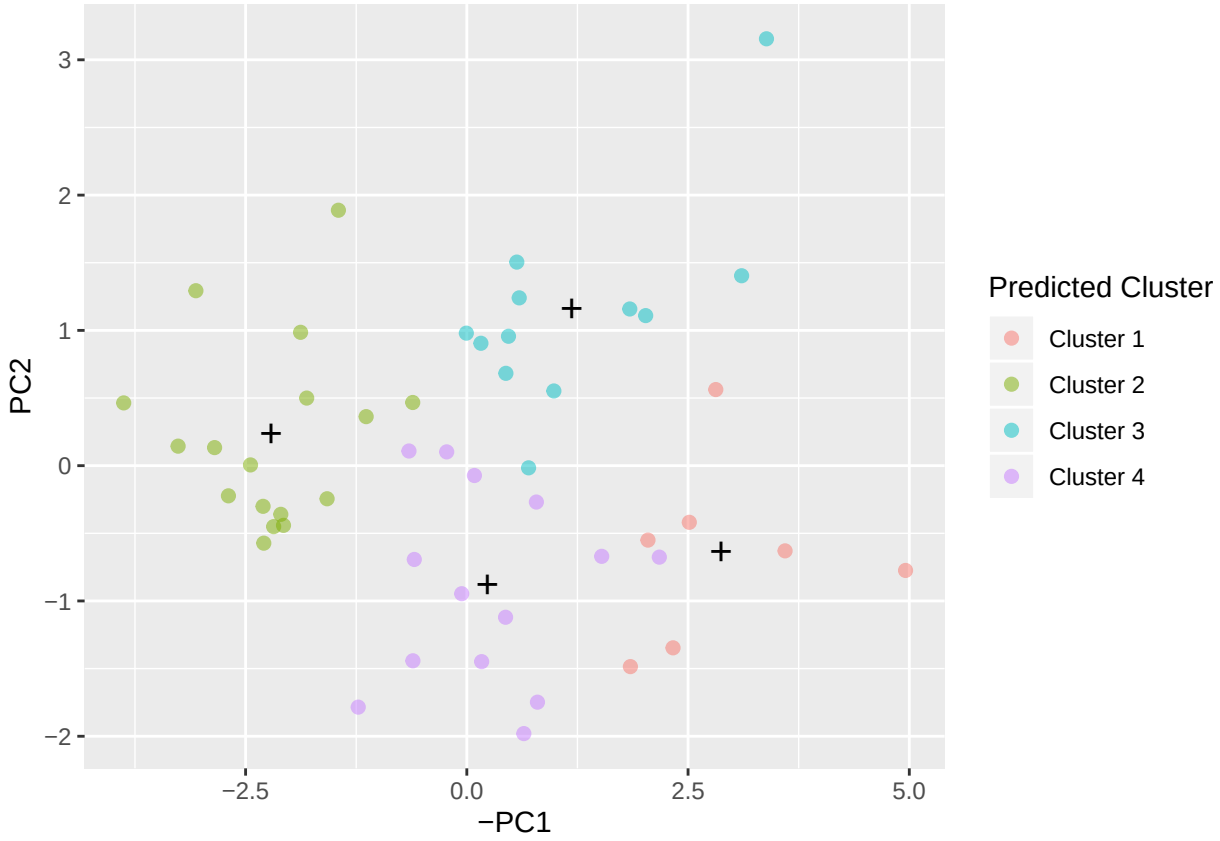
We now plot the data in the 2-dim PCA space of the first two principal variables. Recall that the first two principal variables explain about 74.3% of the variation. The points are colorized according to their group membership found by k-means. The centers of the four groups are also plotted in PCA space.

```
state_crime_pca_proj <- ml_pca(state_crime_std_sdf, k = 2) %>%
  sdf_project(state_crime_std_sdf)
state_crime_kmeans_centers <- state_crime_pca_proj %>%
  sdf_bind_cols(state_crime_kmeans_predict_sdf) %>%
  group_by(prediction) %>%
  summarise(
    PC1 = mean(PC1, na.rm = TRUE),
    PC2 = mean(PC2, na.rm = TRUE)
  ) %>%
  collect()
state_crime_kmeans_centers[order(state_crime_kmeans_centers$prediction),]
```

```
## # A tibble: 4 x 3
##   prediction    PC1    PC2
##        <int>  <dbl>  <dbl>
## 1          0 -2.87  -0.663
## 2          1  2.21   0.215
## 3          2 -1.19   1.14
## 4          3 -0.232 -0.903
```

```
state_crime_pca_proj %>%
  collect() %>%
  ggplot(aes(-PC1, PC2)) +
  geom_point(aes(-PC1, PC2,
```

```
              col = factor(state_crime_kmeans_predict_df$prediction + 1)),
           size = 2, alpha = 0.5) +
geom_point(data = state_crime_kmeans_centers,
           aes(-PC1, PC2),
           pch = '+', size = 5) +
scale_color_discrete(name = "Predicted Cluster", labels = paste("Cluster", 1:4))
```



This plot agrees with the analysis in Section 8.2.2 except that we reversed the scale of PC2 and the clusters are labeled by different colors. These changes are not relevant to cluster identification.

```
spark_disconnect(sc)
```