CS 596: Machine Learning

unsupervised k-means

Program Approach:

There are three place holders in this assignment.

Place Holder #1: *calc\_euclidean\_dist\_vector*(vector1, vector2): ): calculate the Euclidean distance between two vectors.

from scipy.spatial import distance

def calc\_euclidean\_dist\_vector(vector1, vector2):

result = distance.euclidean(vector1, vector2)

return result

Place Holder #2: *get\_centroids*(dimensions, xy\_groups): calculate centroid for each group

length = len(xy\_groups)

centroid\_points = np.zeros((length, dimensions))

for i in range(length):

centroid\_points[i, :] = get\_point\_mean(xy\_groups[i])

Place Holder #3: *get\_sum\_of\_squares*(dimensions, center, samples): calculate the sum of square errors for the current clustering result.

def get\_sum\_of\_squares(dimensions, center, samples):

for i in range(0,len(samples)):

for j in range(0,9):

holder = samples[i,j]-center[j]

sse = sse + math.pow(holder,2)

return sse

Initialization of centroids:

# pick 2 random sets of x,y coordinates to be centroids to start off

centroid\_points = np.random.randint(samples\_size, size=(k\_value, data\_dimensions))

The program calculates the sum of squared errors (SSE) as a quantitative metric of the clustering result.

I explored 4 k-values (K=3,6,8,10)

When K = 3:

[ 34 252 35 73 70 293 237 148 218]

[217 14 254 290 18 201 289 62 99]

[235 200 178 211 102 262 106 325 197]]

Found due to 0 minimum changes

Found after 11 iterations

When K=6:

[[231 282 327 243 221 39 207 179 280]

[232 325 249 33 234 190 263 84 61]

[ 85 42 76 140 275 19 251 37 67]

[289 250 146 162 23 51 227 123 76]

[323 187 152 185 44 277 33 2 149]

[ 4 248 234 117 300 288 310 215 58]]

Found due to 0 minimum changes

Found after 8 iterations

When K=8:

[281 240 235 148 38 236 127 115 57]

[249 65 42 65 41 85 82 180 323]

[ 52 66 170 237 207 4 5 296 190]

[301 192 210 294 96 314 216 155 67]

[261 17 193 265 147 312 269 139 116]

[ 73 227 193 204 83 116 43 36 19]

[151 98 32 210 179 225 76 297 108]

[ 43 229 51 46 143 53 161 183 117]]

Found due to 0 minimum changes

Found after 35 iterations

When K = 10:

[[164 14 313 106 43 168 273 67 100]

[ 39 127 182 144 328 158 237 3 239]

[ 78 104 17 266 62 264 267 256 72]

[158 47 271 13 197 82 225 307 327]

[ 36 6 162 204 118 127 175 277 224]

[109 263 48 13 25 188 194 207 248]

[295 194 90 217 92 308 287 251 197]

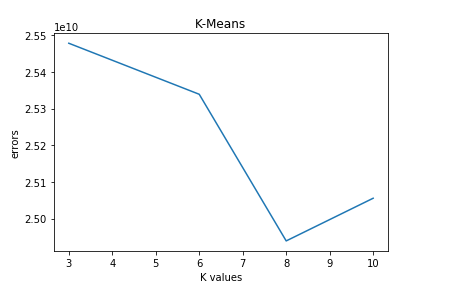
[ 29 170 182 216 130 220 159 192 73]

[197 237 50 259 92 96 72 73 49]

[ 42 117 316 308 265 320 209 306 36]]

Found due to 0 minimum changes

Found after 22 iterations



K = 3) SSE = 25478494577.0)

K = 6) SSE = 25339170274.0)

K = 8) SSE = 24938814365.0)

K = 10) SSE = 25055332559.0)

In this particular example, when K=8, SSE reaches the minimum, and it took most iterations (35) to converge vs when K=3 (11 iteration), 6 (8 iterations), and 10(22 iterations). Most often, the most optimal K-values will have most iterations as it converges but not always the case.