Pokhara University

Everest Engineering Collage

Sanapa-2, Lalitpur

Lab Report on “K Clustering”

Submitted to
Shailesh Pandey
of
IT Department

Submitted By
Bhupendra Prasad Bhat-[20120010]

Class Roll No :-04

Submitted Date

2023/07/2

kmeans.java

```java
import java.text.DecimalFormat;
import java.util.Arrays;
import java.util.Random;

public class kmeans {
    int x[], y[];                // data points
    int num;                     // number of data points (supplied by the user)
    int k;                       // number of clusters (supplied by the user)
    double meanX[],meanY[];    // cluster centres
    double oldX[],oldY[];        // backup old cluster centres
    int cAssign[];               // cluster assignment

    private kmeans(int num,int clusters){
        this.num = num;
        x= new int[num];
        y= new int[num];
        k=clusters;
        meanX = new double[k];
        meanY = new double[k];
        oldX = new double[k];
        oldY = new double[k];
        cAssign = new int[num];

    }

    void randomMean() {
        //Initialize meanX and meanY with random values between 0 and 500 for all k centres
        //using the nextInt() method in the java.util.Random class
        for (int i = 0; i < k; i++) {
            Random rand = new Random();
            int first  = rand.nextInt(499)+1;
            int second = rand.nextInt(499)+1;

            this.meanX[i]=first*1.0;
            this.meanY[i]=second*1.0;


        }

    }


    void randomData(){
        // for random x and y data
        for (int i = 0; i < num; i++) {
            Random rand = new Random();
            int first  = rand.nextInt(499)+1;
            int second = rand.nextInt(499)+1;
            x[i]=first;
            y[i]=second;

        }

    }
```

```
55
56      void assignCluster() {
57          // Calculate the distance between the point and the cluster centre
58          // The Euclidean distance between the jth data point and ith cluster centre is:
59          double distance[]=new double[k];// stores the k distances
60          for (int i = 0; i < num; i++) {
61              for (int j = 0; j < k; j++) {
62                  distance[j] = Math.sqrt(Math.pow(x[i]-meanX[j],2) + Math.pow(y[i]-meanY[j],2));
63              }
64
65              double minValue = distance[0];
66              int minIndex =0;
67              for (int j = 0; j < k; j++) {
68                  if (distance[j] < minValue) {
69                      minValue = distance[j];
70                      minIndex = j;
71                  }
72              }
73              cAssign[i]=minIndex;
74
75          }
76
77          // Calculate distance for all k clusters and assign j to whichever i has the smallest
    distance
78          // Assign this i to cAssign
79      }
80
81      void updateMeans() {
82          // Before updating the centres, backup meanX and meanY (copy to oldX and oldY)
83
84          for (int j = 0; j < k; j++) {
85              oldX[j]=meanX[j];
86              oldY[j]=meanY[j];
87          }
88
89          // Calculate meanX and meanY for each cluster
90
91          //this stores total sum in meanX and meanY
92          for (int j = 0; j < k; j++) {
93              for (int i = 0; i < num; i++) {
94                  if(cAssign[i]==j){
95                      meanX[j]+=x[i];
96                      meanY[j]+=y[i];
97                  }
98              }
99
100         }
101         //updatindg meanX and meanY by dividing it with num ie 100 [which eventually the mean]
102         for (int j = 0; j < k; j++) {
103             meanX[j]=meanX[j]/num;
104             meanY[j]=meanY[j]/num;
105         }
106
107     }
108
109     boolean isDifferent() {
110         //return true if meanX≠oldX or meanY≠oldY for one or more clusters
111         //Otherwise return false
```

```
112
113
114         if(meanX[0]!=oldX[0]){
115             return true;    //returns true
116         }else{
117             return false;
118         }
119     }
120
121
122
123     void doClustering() {
124         //This is where you implement the clustering algorithm. Simple isn't it.
125         randomMean();
126         randomData();
127         do {
128             assignCluster();
129             updateMeans();
130         } while(isDifferent());
131     }
132
133     public static void main(String[] args){
134
135         kmeans km = new kmeans(100,2);
136         km.doClustering();
137         System.out.println("The X and Y values are");
138         System.out.println(Arrays.toString(km.x));
139         System.out.println(Arrays.toString(km.y));
140         System.out.println("The value after k clustering");
141         System.out.println(Arrays.toString(km.cAssign));
142 //        System.out.println(Arrays.toString(km.meanX));
143 //        System.out.println(Arrays.toString(km.meanY));
144 //        System.out.println(Arrays.toString(km.oldX));
145 //        System.out.println(Arrays.toString(km.oldY));
146
147
148
149     }
150
151 }
152
```

output.txt

```
run:
The X and Y values are
[20, 396, 76, 207, 476, 455, 374, 172, 215, 268, 248, 27, 379, 262, 235, 451, 74, 4, 473, 23, 473, 466, 280, 4
[146, 466, 61, 184, 339, 328, 328, 94, 370, 330, 281, 252, 446, 320, 70, 386, 109, 95, 150, 68, 200, 65, 388,
The value after k clustering
[0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1
BUILD SUCCESSFUL (total time: 0 seconds)
```