K-Medoids Clustering Implementation

DWDM project

BY,

Arjun Chaudhary - 100911119

**Abstract**

This project centers on the discussion of k-medoid-style clustering algorithms for supervised summary generation. This task requires clustering techniques that identify class-uniform clusters. This project investigates such a novel clustering technique we term supervised clustering. Our work focuses on the generalization of k-medoid-style. clustering algorithms. PAM is a classical partitioning technique of clustering that the data set of n objects into k clusters. It is a partitioning algorithm and attempts to minimize the distance between points labeled to be in a cluster and a point designated as the center of the cluster. The algorithm PAM first computes representative objects, called medoids, where the number of medoids depends on number of clusters desired by the user. A medoid is that object of a cluster considered to be its “center”. A medoid is the most centrally located data object in a cluster. Here, k data objects are selected randomly as medoids to represent k cluster and remaining all data objects are placed in a cluster having medoid nearest (or most similar) to that data object. After processing all data objects, new medoid is determined which can represent cluster in a better way and the entire process is repeated. Again all data objects are bound to the clusters based on the new medoids. In each iteration, medoids change their location step by step. Or in other words, medoids move in each iteration. This process is continued until no any medoid move. As a result, k clusters are found representing a set of n data objects.

We would like to thank Mr **Keshavamurthy B N**(Asst. Professor - Sr. Scale MIT, manipal) without his help it would not have been possible for us to finish the project on time.

**Introduction**

* **Input** : ‘k’, the number of clusters to be partitioned;

‘n’, the number of objects.

* **Output**: A set of ‘k’ clusters
* The algorithm proceeds in the following steps:
* Initialize: randomly select *k* of the *n* data points as the medoids
* Associate each data point to the closest medoid.

(The distance metric is calculated using Manhattan

Distance metric)

3. For each medoid *m*

For each non-medoid data point *o*

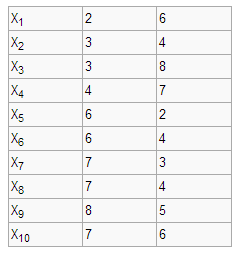
Swap *m* and *o* and compute the total cost of the

configuration

* Select the configuration with the lowest cost.
* Repeat steps 2 to 4 until there is no change in the

medoid.

Consider a data set of ten objects as follows: *k* = 2



**Step 1**

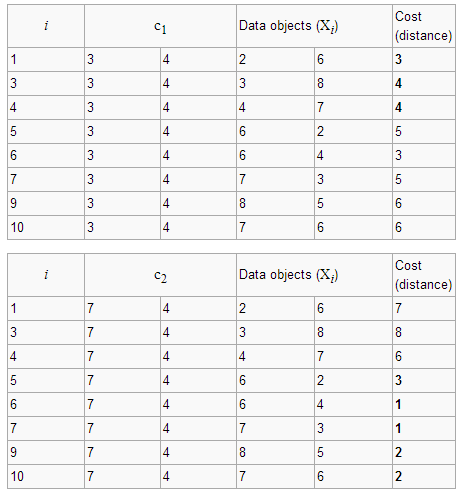
Initialize *k* centers.

Let us assume c1 = (3,4) and c2 = (7,4)

So here c1 and c2 are selected as medoids.

Calculate distance so as to associate each data object to its nearest medoid. Cost is calculated using

Manhattan distance. Costs to the nearest medoid are shown bold in the table.



Then the clusters become:

Cluster1 = {(3,4)(2,6)(3,8)(4,7)}

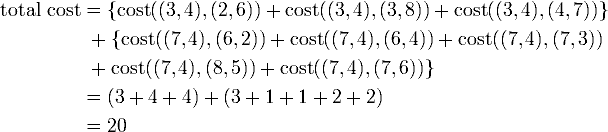
Cluster2 = {(7,4)(6,2)(6,4)(7,3)(8,5)(7,6)}

Since the points (2,6) (3,8) and (4,7) are closer to c1 hence they form one cluster whilst remaining points from another cluster.So the total cost involved is 20. Where cost between any two points is found using formula where *x* is any data object, *c* is the medoid, and *d* is the dimension of the object which

in this case is 2.

Total cost is the summation of the cost of data object from its medoid in its cluster so here:





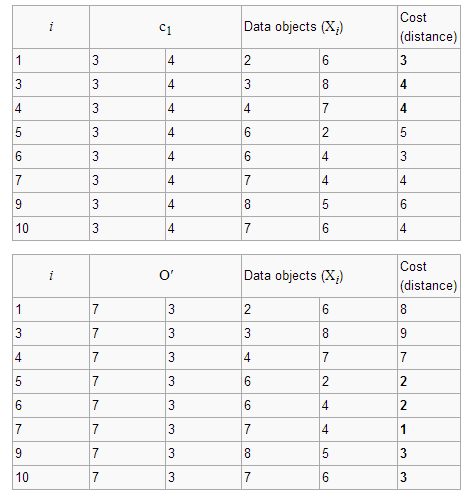
Step 2

Select one of the nonmedoids  O′

Let us assume O′ = (7,3)

So now the medoids are c1(3,4) and O′(7,3)

If c1 and O′ are new medoids, calculate the total cost involved



So cost of swapping medoid from c2 to O′ is So moving to O′ Total cost=3+4+4+2+2+1+3+3

S=current total cost – past total cost

=22-20

=2>0

So moving to O′ would be a bad idea, so the previous choice was good. So we try other non medoids and found that our first choice was the best. So the configuration does not change.

**References:-**

1. Datawarehousing and Datamining by Han and Kamber

2. Research8.net

3. Wikibooks.org

4. Class notes by Mr. Keshavmurthy B N(Asst. professor - sr. scale MIT,Manipal)

**Code**

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

packagepamimplement;

/\*\*

\* k=no of clusters

\* size=size of data set

\* dataset[][]=the dataset

\* c1 is the data object of class Clustering

\*/

importjava.util.\*;

import java.io.\*;

class Clustering {

publicint i = 0, j = 0, l = 0, k = 0, m = 0, r = 0, noofmedoids = 0, size = 0;

intc\_id = 1, tempx = 0, tempy = 0, medoid\_id\_index = 0;

inttotalcost, prevcost,presentcost;

int x=0,y=0,z=0;

intmedoid\_id[] = new int[100];

int[][] dataset = new int[500][2];

int[][] medoidarr = null;

int[][] nonmedoidarr = null;

int[][] prevmedoidarr = null;

int[][] prevnonmedoidarr = null;

Clustering(int k) {

this.noofmedoids = k;

}

public void read(String str) throws FileNotFoundException

{

Scanner s = new Scanner(new File(str));

while (s.hasNextInt()) {

dataset[size][0] = s.nextInt();

dataset[size++][1] = s.nextInt();

}

medoidarr = new int[size][3];

nonmedoidarr = new int[size - k][3];

}

public void generate(String str,intlen) throws IOException

{

PrintWriterpr = new PrintWriter(str);

int temp1,temp2,arsize=0,flag=0,ab=0;

intwritearr[][]=new int[len][2];

temp1 = (int) Math.floor(100 \* Math.random());

writearr[arsize][0]=temp1;

temp2 = (int) Math.floor(100 \* Math.random());

writearr[arsize][1]=temp2;

for(arsize=1;arsize<len;arsize++)

{

while(true)

{

flag=0;

temp1 = (int) Math.floor(100 \* Math.random());

temp2 = (int) Math.floor(100 \* Math.random());

for(ab=0;ab<arsize;ab++)

{

if(writearr[ab][0]==temp1&&writearr[ab][1]==temp2)

{

flag=1;

break;

}

}

if(flag==0)

{

writearr[arsize][0]=temp1;

writearr[arsize][1]=temp2;

break;

}

}

}

for (intij=0; ij<len ; ij++){

pr.println(writearr[ij][0]+" " + writearr[ij][1]);

}

pr.close();

}

public void randomele() {

inttemprand, flag;

while (true) {

temprand = (int) Math.floor(100 \* Math.random());

if (temprand>= size) {

continue;

} else {

medoid\_id[medoid\_id\_index++] = temprand;

break;

}

}

for (i = 1; i <noofmedoids; i++) {

while (true) {

flag = 0;

temprand = (int) Math.floor(100 \* Math.random());

if (temprand< size) {

for (j = 0; j < i; j++) {

if (temprand == medoid\_id[j]) {

break;

} else {

flag++;

}

}

}

if (flag == i) {

medoid\_id[medoid\_id\_index++] = temprand;

break;

}

}

}

for (l = 0, j = 0; l <noofmedoids; l++, j++) {

medoidarr[l][0] = c\_id++;

medoidarr[l][1] = dataset[medoid\_id[j]][0];

medoidarr[l][2] = dataset[medoid\_id[j]][1];

}

}

public void cluster() {

m=0;c\_id = 0;

for (i = 0; i < size; i++) {

int flag1 = 0;

intuits = 0;

interred = 999999;

// c\_id = 0;

for (j = 0; j <noofmedoids; j++) {

if (medoidarr[j][1] == dataset[i][0] &&medoidarr[j][2] == dataset[i][1]) {

flag1 = 1;

break;

}

}

if (flag1 == 1) {

continue;

} else {

for (j = 0; j <noofmedoids; j++) {

dist = (Math.abs(medoidarr[j][1] - dataset[i][0]) + Math.abs(medoidarr[j][2] - dataset[i][1]));

if (prep>dist) {

prep = dist;

c\_id = j + 1;

}

}

//System.out.println("value of c\_id:"+c\_id+"m "+m);

nonmedoidarr[m][0] = c\_id;

nonmedoidarr[m][1] = dataset[i][0];

nonmedoidarr[m++][2] = dataset[i][1];

// System.out.println("after value of c\_id:"+c\_id+"m "+m);

}

}//cost();

}

publicint cost() {

// provost = totalcost;

totalcost = 0;

for (i = 0; i <noofmedoids; i++) {

for (j = 0; j < (size - noofmedoids); j++) {

if (medoidarr[i][0] == nonmedoidarr[j][0]) {

totalcost += (Math.abs(nonmedoidarr[j][1] - medoidarr[i][1]) + Math.abs(nonmedoidarr[j][2] - medoidarr[i][2]));

}

}

}

returntotalcost;

}

public void checkandswap()

{

tempx=tempy=0;

for(x=0;x<noofmedoids;x++)

{

for(y=0;y<size;y++)

{

for(z=0;z<size-noofmedoids;z++)

if(dataset[y][0]==nonmedoidarr[z][1]&&dataset[y][1]==nonmedoidarr[z][2]){

tempx=medoidarr[x][1];

tempy=medoidarr[x][2];

provost=cost();

System.out.println("cost:"+provost);

medoidarr[x][1]=nonmedoidarr[z][1];

medoidarr[x][2]=nonmedoidarr[z][2];

cluster();

presentcost=cost();

System.out.println("after swap cost:"+presentcost);

if(presentcost>=provost)

{

medoidarr[x][1]=tempx;

medoidarr[x][2]=tempy;

cluster();

System.out.println("Old is better");

}

else

{

z=0;

System.out.println("New is better");

}

}

}

}

}

public void display()

{

displaymedoids();

displaynonmedoids();

displayclusters();

}

public void displaymedoids() {

System.out.println("The medoids are:");

System.out.println("------------------------------------------");

System.out.println("Cluster Id\tX\t\t Y");

for (i = 0; i <noofmedoids; i++) {

System.out.print(medoidarr[i][0] + "\t\t" + medoidarr[i][1] + "\t\t" + medoidarr[i][2] + "\n");

}

System.out.println("------------------------------------------");

System.out.println("------------------------------------------");

}

public void displaynonmedoids()

{

System.out.println("The Non medoids are:");

System.out.println("------------------------------------------");

System.out.println("Cluster Id\tX\t\t Y");

for (i = 0; i < (size - noofmedoids); i++)

System.out.print(nonmedoidarr[i][0] + "\t\t" + nonmedoidarr[i][1] + "\t\t" + nonmedoidarr[i][2] + "\n");

System.out.println("------------------------------------------");

System.out.println("------------------------------------------");

}

public void displayclusters()

{

System.out.println("The clusters are:");

System.out.println("Cluster Id\tX\t\t Y");

System.out.println("------------------------------------------");

for (i = 1; i <= noofmedoids; i++) {

for(j=0;j<noofmedoids;j++)

if(i==medoidarr[j][0])

System.out.print(medoidarr[j][0] + "\t\t" + medoidarr[j][1] + "\t\t" + medoidarr[j][2] + "\n");

for(j=0;j<(size-noofmedoids);j++)

if(i==nonmedoidarr[j][0])

System.out.print(nonmedoidarr[j][0] + "\t\t" + nonmedoidarr[j][1] + "\t\t" + nonmedoidarr[j][2] + "\n");

}

System.out.println("------------------------------------------");

System.out.println("------------------------------------------");

}

}

public class Pam {

public static void main(String[] args) throws FileNotFoundException,IOException {

intnoofmedoids = 15, i = 0, j = 0, size = 0;

int dataset[][] = new int[100][2];

String file="E:\\ICT\\sem 6\\Dwdm Project\\dwdmdataset.txt";

Clustering c1 = new Clustering(noofmedoids);

c1.generate(file,50);

c1.read(file);

c1.randomele();

c1.cluster();

System.out.println("init cost:"+c1.cost());

c1.display();

c1.checkandswap();

c1.display();

System.out.println("after swap cost:"+c1.cost());

}

}