**TWISTER K-MEANS REPORT**

Doga Tuncay, Fei Tang

*Indiana University School of Informatics and Computing*

*B649 Cloud Computing Class Project*

What is K-means, why do we need Objective Function : (from an online resource)

K-means is a data mining approach to find out the clusters by doing Euclidean Distance sequentially.[1]

In sequential approach, we have to have a step to do a global reduction. We have to take each data point as a consideration. That is why when we finish this task we have the do a reduction and then combine the other data points that might belong the current data center. Then we pool all things together into the reduction and that will be our answer of whole dataset.

Because of this algorithm is iterative, not every step we have the correct centroid. In each step we try to refine it and stabilize and converge to say this is the answer. Then we go back the beginning of the operation and say we have a current global centroid vector and we broadcast the each of subsets and re-calculate again. So we loop through all the dataset calculate the Euclidian Distance to say which data point belongs to which centroid. Then we go back to the broadcast step recalculate and do this again sequentially. This version of the problem has two gigantic nested loops that loops through data points and data centroids and solves the problem sequentially.

Rather than this approach, we plan to solve this in a parallel system by dividing into small pieces. Within one small piece we do the same process sequentially than combine it. Also we know that K-means converges very fast. However, the problem is, it is very likely to get the local minimum result which gives us a wrong answer.

Sequential : O(nk)

We can partition the data equally in this problem, but if we look at one partition what is the computationally complexity of the mapper task is a problem?

One partition in Map : O((n/p).k)

k : centers n : data points (evenly partitioned) p : partition n/p: evenly partitioned data points

Complexity is the same as in the sequential except we have a subset of data points, because we do parallelism with partitions of data points.

What is the complexity of the Reducer? Let’s say we have one reducer, which can be more. When we do global reduction after we do the subs’ reduce. In the reducer we want to do some computations, to compare the results, because we have to have a convergence and we have to find where to stop this comparing. However, we have a complication here. If we compare the difference between the centroids, in other words we already have the vectors all we need to do is to calculate the vectors versus the previous ones. The main case is when we are comparing the centroids, we should give a decision to stop and give an answer by defining a threshold.

So, in reducer we are comparing two iterations (vectors and the previous vectors) to find the difference between the centroids and the compare the difference. So the problem computation is depends on the centroids which is O(k) where k is a dimensional vector.

When we run the algorithm we get different answers, however we want to find the best solution. To solve this problem, we needed to design an objective function. This algorithm aims at minimizing an objective function, in this case a squared error function. The objective function,

http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/images/image009.gif,

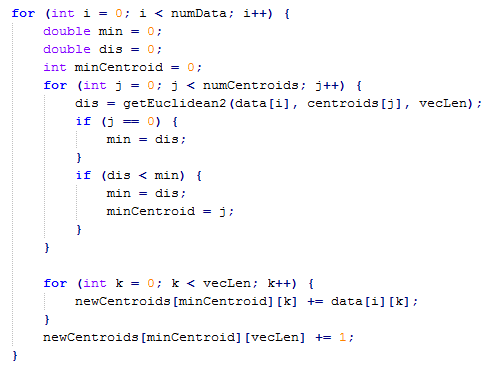
where http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/images/image011.gif is a chosen distance measure between a data point http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/images/image013.gif and the cluster centroid http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/images/image015.gif, is an indicator of the distance of the n data points from their respective cluster centroids.[1]

Consider that we have 10 centers of clusters. If we use each data point and calculate the distance the center of the clusters that are already decided and add all the Euclidian distance together then we can get the minimum result. When we add the Euclidian distances we get a summation of the distances then we try to get the best result by using objective function by using the minimum of the summation. There are some other approaches but this is the most straight forward and naïve approach.

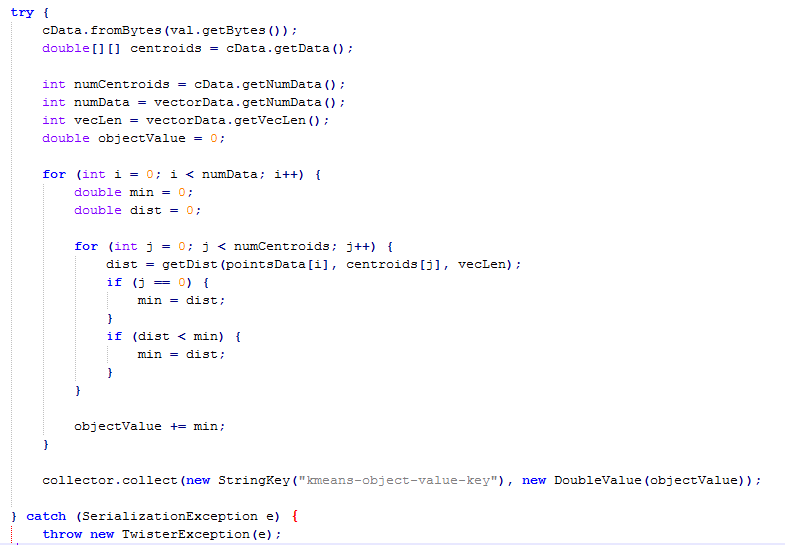
The most computation is needed in the mapper (Euclidian distance) and then the stop condition comes which is the minimum function. In mapper we run the algorithm through the data points and in the reducer we calculate the global centroids and then we calculate the difference (reduction). Which means when we run this algorithm the size of the reduction depends on the problem. 10 vector centroids could be a 1000 but it is much smaller than the whole data points size N. So the most costing computation when we do the mapper (Euclidian Distance). We know that the complexity is order of nk (O(nk)) so the problem size goes up, because the main computation is in the mapper task. Fortunately, if our data centroids are very small, communication between mapper and reducer will go down, but it depends on the program (as the class example of the image processing problem that has a lot of communication even we reduce the centroids). So, the answer for question C is not very obvious based on what the problem is.

In our solution we have the input scale of 80x1000=80000 data points. There are 80 mappers and 1 reducer to run the program.

To calculate the objective value, we had to add an extra map-reduce as what you can see files. Here are the two loops we use to do kmeans clustering sequentially:

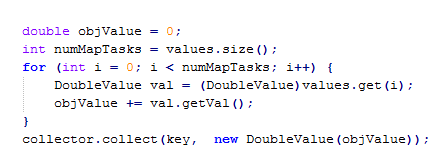


And here is the Objective Function in the Mapper Task :



In here, we try to find the minimum distances and if we have a larger minimum we add it into the ObjectValue which is the <value> and we have one <key> because just one reducer is required here.

Here is the objective function implementation in the reducer part:



where we define an initial ObjValue again and while we have a number of Map Tasks we run through the number of the mappers and we do the same computation from Mapper and get the final objValue and collect it.

Logic is simple, it re-broadcasts all the points data to mappers and use the final centroids as the static data, so that each mapper can calculate to value of J, and then send the value to reducer. Reducer adds all the J from mappers to get the final answer.

Here is the basic data flow representation of the project:

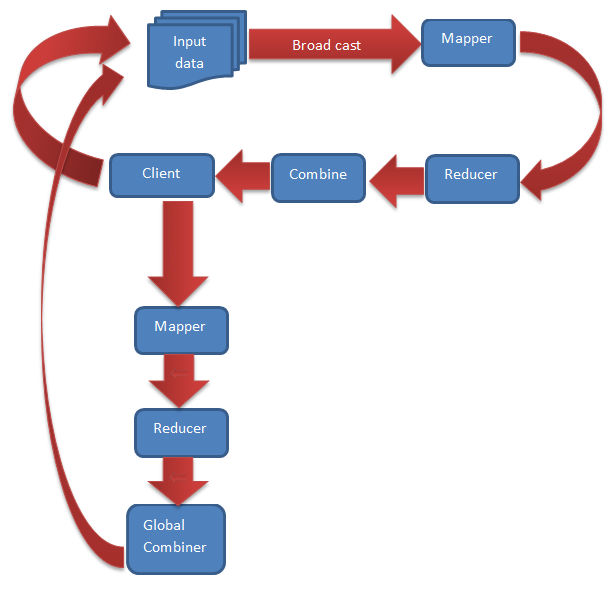


Chart 1. Data Flow of The Twister K-Means

|  |  |
| --- | --- |
| runs | Objective Function Result |
| 1 | 3.7366832148726835E9 |
| 2 | 3.7368089283604774E9 |
| 3 | 3.7365993082278466E9 |
| 4 | 3.73690864637493E9 |
| 5 | 3.736636430974697E9 |
| 6 | 3.736666911394709E9 |
| 7 | 3.73671887878718E9 |
| 8 | 3.743448436517685E9 |
| 9 | 3.7373550855888844E9 |
| 10 | 3.7367490615668874E9 |

Table 1. Results of the Objective Function

We observed that the objective function’s result is different because we use different initial centroids as input. We think it’s a natural result because different initial centroids have different distance from the optimal clustering centroids. So, the objective value will be smaller if the initial centroid is relatively close to the optimal centroids and vice versa.

REFERENCES

1. <http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html>