2. Figures

2.1

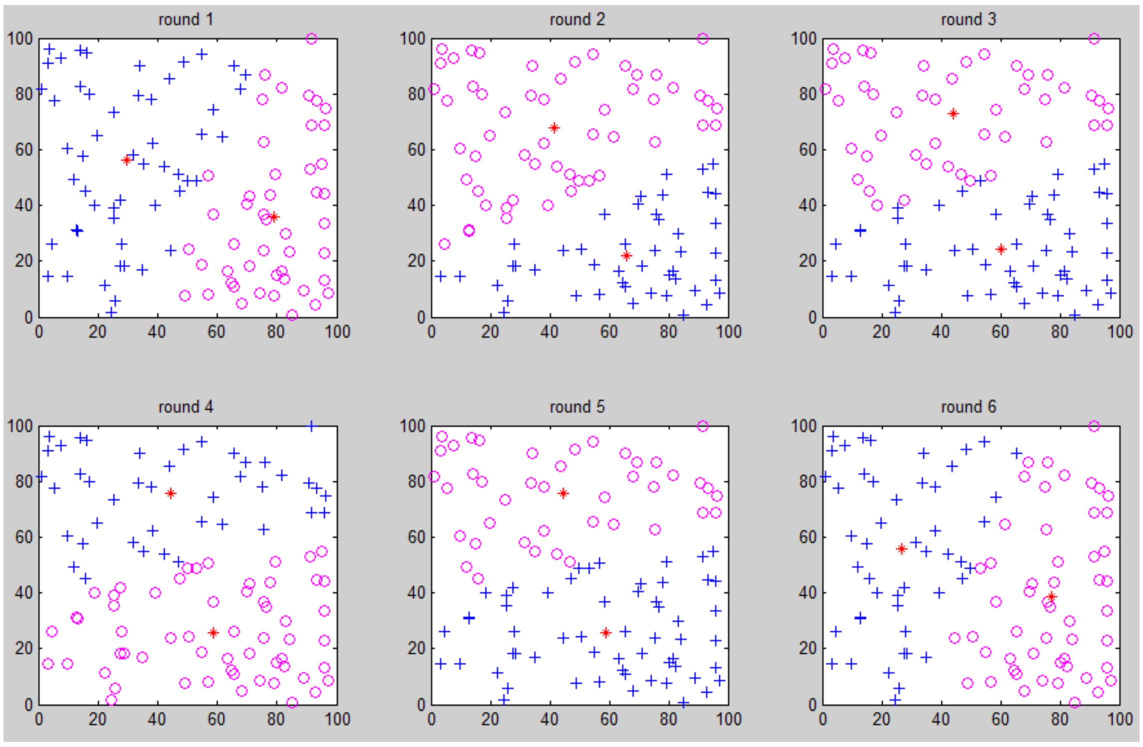

Figure.1

2.2

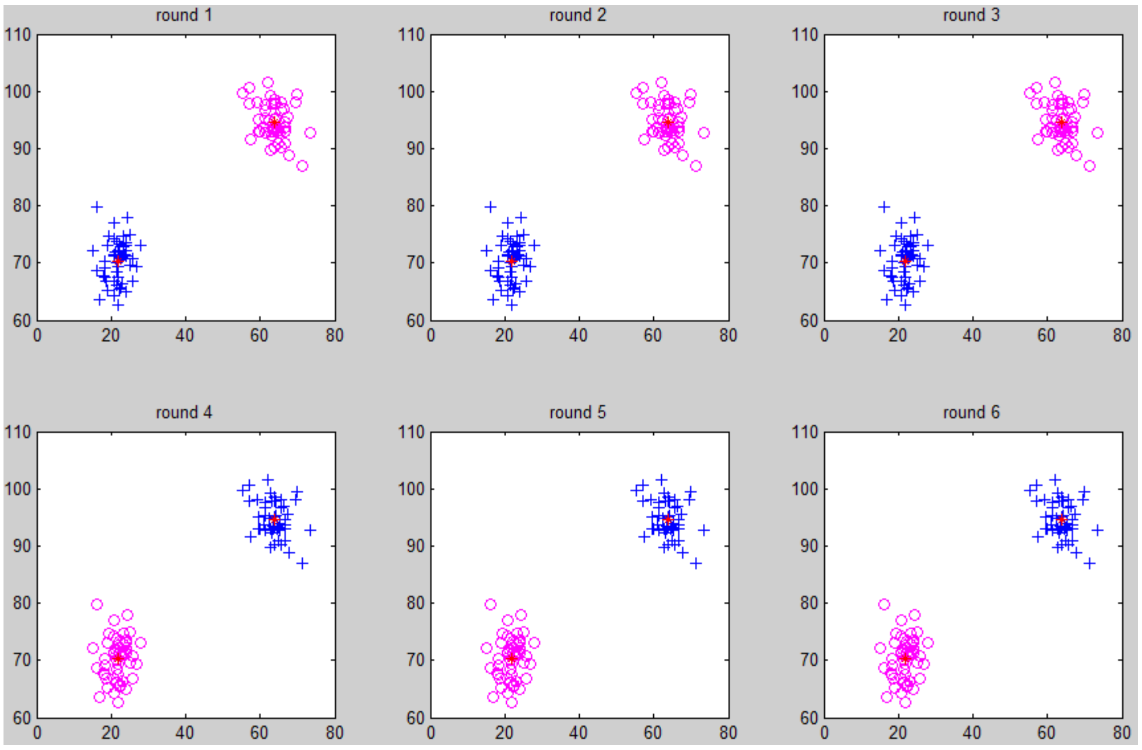

Figure.2

2.3 figures（2 experiment results）

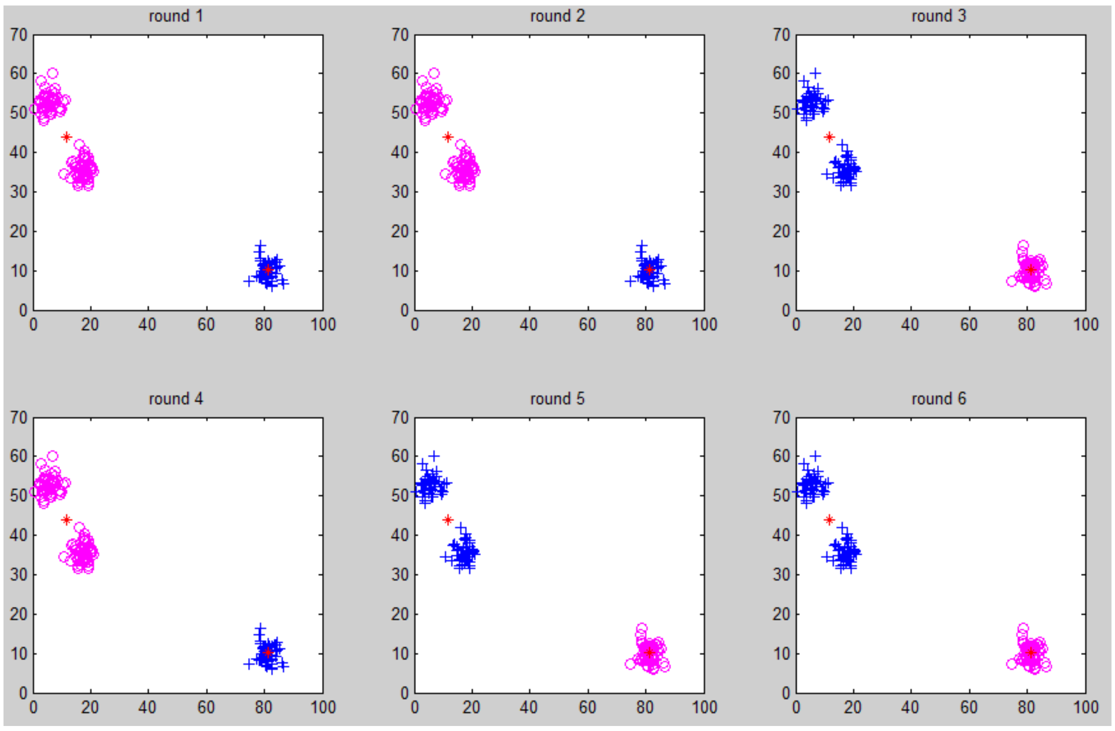

Figure.3

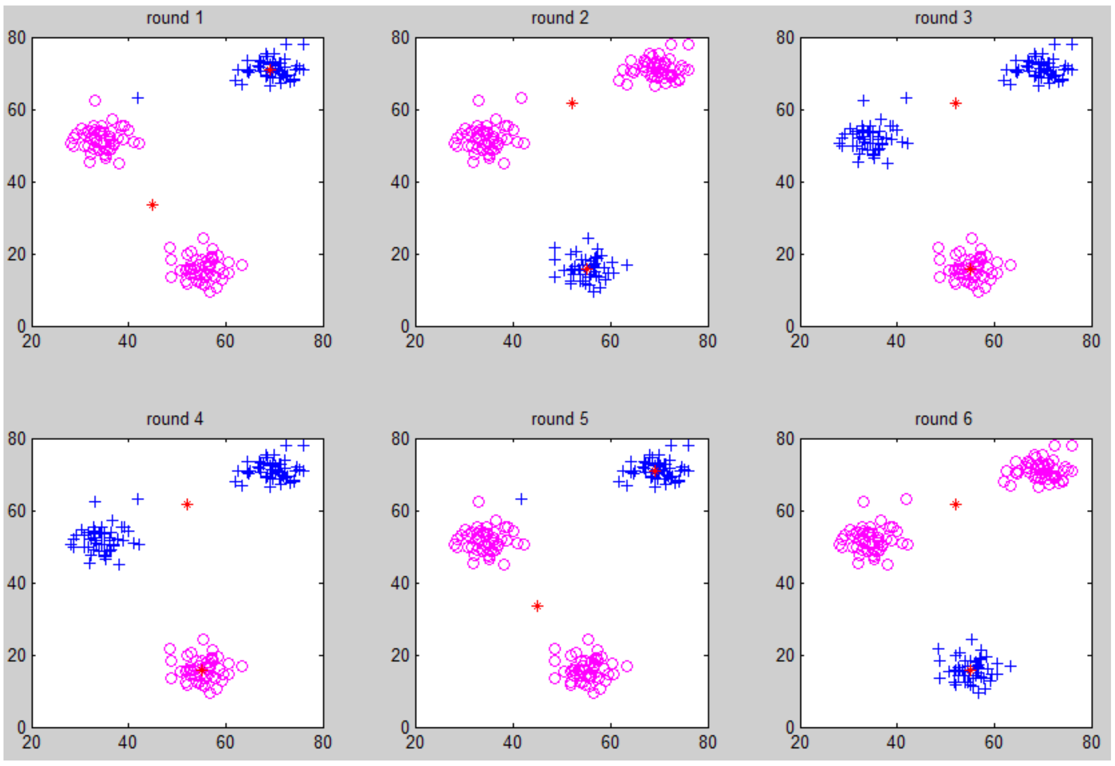

Figure.4

2.4

From figure1, we can infer that the result of kmeans is sensitive to the starting points especially when the data set has no extremely separated clusters.

Comparing figure2 with figure1, we can find that, no matter how to choose the starting points (The different color assignments of round1 and round4 can tell the roughly difference between the initial centroids choosing of two rounds.), kmeans does quite well when k is correctly chosen and the k clusters are separated far away to each other.

Under the similar situation as problem2.2 (i.e. if clusters of the data set are separated far away to each other), the results of problem2.3 reveal that, when k is less than the real amount of clusters, the result of kmeans also make sense, because the points which are in the same cluster (in real) will still in the same clusters (by kmeans).

In addition, when k is less than the real amount of clusters, comparing figure3 with figure4, we may find that: if some of the clusters are much closer to each other than to the others (see in figure3), no matter how to choose the starting points, these clusters will be assigned to the same clusters by kmeans with high probability; however, if the clusters have similar distance to each other (see in figure4), the result of clustering will be different depending on the starting points.

3. Kmeans in MapReduce

(mapper/reducer finish **one iteration, which is the problem requires**; main function does all iterations.)

```
main () {
    init_centroids_list = randomly pick k points from input data;
    old_centroids_list = // old centroid list after t-1 iterations
    cur_centroids_list = null; //current centroid list after t iterations
    while( !isequal(pre_centroids_list , cur_centroids_list)){
        runJob();
        old_centroids_list = cur_centroids_list;
    }
    Output(cur_centriods_list);
}
Map (key, value) //assume global variables can be passed to mapper/reducer
{
    //key: document name
    //value: document content
    For each input data point p in value do{
        distance_list = distances between p with centroids in old_centroids_list;
        clusterIdx=IndexOf(Min(distance_list));
        EmitIntermediate(clusterIdx, p);// the w here is after ith iteration
    }
}

Reduce (key, value)
{
     //key: clusterIdx
     //value: a list of p
     NewCentroid=mean(value);
     cur_centroids_list[clusterIdx]=NewCentroid;
   Emit(clusterIdx, NewCentroid);
}
```

4. Proof

4.1

(a) Prove that , where

To prove it, we first consider the following situation: suppose we are required to divide one cluster C into two parts C_1 and C_2. It is easy to compute that, if we pick one of its boundary points out of this cluster as C_1 and the rest points become C_2, thus we will have:

Therefore, given optimal k clusters, we can always pick one cluster and divide it into two clusters which satisfies the previous condition, and then get k+1 clusters. This k+1 clustering may not be the optimal solution, but it always gets . Since , we can always get .

(b) We cannot prove for any clustering, because the result of kmeans is local optima which depends on the starting points. Suppose we have a local optima for k clustering that then we may get another local optima for k+1 clustering that which violates the hypothesis.

4.2 Prove the convergence of kmeans:

Let's go through the process of kmeans:

At step0, pick initial centroids and then compute the total distance between each object and its nearest centroid. Call this distance . Then we recalculate the centroids at step1, and then compute . At step1 we are minimizing the total distance of each cluster's objects from the respective centroid, so the new distance D[1] either decreases or remains the same. That is,. In the subsequent assignment step, if an object object is reassigned to a different centroid then it must be that this object is closer to its new centroid than to the previously one. If an object is not reassigned, then there is obviously no change in its distance. Therefore, we have in th iteration.

Since and the goal of kmeans is to find the minimum of D which exists, so there must exists an subject to , which means no point at that moment will be assigned to different cluster and the centroids are stable from then on.