Stream Clustering

Stream Clustering Algorithm

- Proposed by BDMO(B. Babcock, M. Datar, R. Motwani, and L. O'Callaghan)
- The BDMO Algorithm builds on the methodology for counting ones in a stream

key similarities and differences:

- **Similarity:** The points of the stream are partitioned into, and summarized by, buckets whose sizes are a power of two. **Difference:** Here, the size of a bucket is the number of points it represents, rather than the number of stream elements that are 1.
- **Similarity:** The sizes of buckets obey the restriction that there are one or two of each size, up to some limit. **Difference:** Here it is not assumed that the sequence of allowable bucket sizes starts with 1. Rather, they are required only to form a sequence where each size is twice the previous size, e.g., 3, 6, 12, 24, ...

key similarities and differences:

- Bucket sizes are again restrained to be non decreasing as we go back in time. we can conclude that there will be O(logN) buckets.

contents of a bucket consists of:

1. The size of the bucket.
2. The timestamp of the bucket, that is, the most recent point that contributes to the bucket. timestamps can be recorded modulo N.
3. A collection of records that represent the clusters into which the points of that bucket have been partitioned. These records contain:
   a) The number of points in the cluster.
   b) The centroid or clustroid of the cluster.
   c) Any other parameters necessary to enable us to merge clusters and maintain approximations to the full set of parameters for the merged cluster.

Initializing Buckets

- Our smallest bucket size will be p, a power of 2.
- Thus, every p stream elements, we create a new bucket, with the most recent p points.
- The timestamp for this bucket is the timestamp of the most recent point in the bucket.
- We may leave each point in a cluster by itself, or we may perform a clustering of these points according to whatever clustering strategy we have chosen. For instance, if we choose a k-means algorithm, then (assuming $k < p$) we cluster the points into k clusters by some algorithm.

Merging Buckets

- whenever we create a new bucket, we need to review the sequence of buckets.
- First, if some bucket has a timestamp that is more than N time units prior to the current time, then nothing of that bucket is in the window, and we may drop it from the list.
- Second, we may have created three buckets of size p, in which case we must merge the oldest two of the three. The merger may create two buckets of size 2p, in which case we may have to merge buckets of increasing sizes, recursively,

To merge two consecutive buckets

- The size of the bucket is twice the sizes of the two buckets being merged.
- The timestamp for the merged bucket is the timestamp of the more recent of the two consecutive buckets.
- We must consider whether to merge clusters, and if so, we need to compute the parameters of the merged clusters.

Answering Queries

- Having selected the desired buckets, we pool all their clusters. We then use some methodology for deciding which clusters to merge.
- if we are required to produce exactly k clusters, then we can merge the clusters with the closest centroids until we are left with only k clusters

Map- Reduce

- Begin by creating many Map tasks. Each task is assigned a subset of the points. The Map function's job is to cluster the points it is given. Its output is a set of key-value pairs with a fixed key 1, and a value that is the description of one cluster.
- Since all key-value pairs have the same key, there can be only one Reduce task. This task gets descriptions of the clusters produced by each of the Map tasks, and must merge them appropriately.