Stream Clustering

Extension of DGIM to More Complex Problems
Clustering a Stream

- Assume points enter in a stream.
- Maintain a sliding window of points.
- Queries ask for clusters of points within some suffix of the window.
- **Important issue:** where are the cluster centroids?
BDMO Approach

- BDMO = Babcock, Datar, Motwani, O’Callaghan.
- $k$-means based.
- Can use less than $O(N)$ space for windows of size $N$.
- Generalizes trick of DGIM: buckets of increasing “weight.”
Recall DGIM

- Maintains a sequence of buckets $B_1, B_2, \ldots$
- Buckets have timestamps (most recent stream element in bucket).
- Sizes of buckets nondecreasing.
  - In DGIM size = power of 2.
- Either 1 or 2 of each size.
Alternative Combining Rule

♦ Instead of “combine the 2\text{nd} and 3\text{rd} of any one size” we could say:
♦ “Combine $B_{i+1}$ and $B_i$ if $\text{size}(B_{i+1} \cup B_i) < \text{size}(B_{i-1} \cup B_{i-2} \cup \ldots \cup B_1)$.”
  ♦ If $B_{i+1}$, $B_i$, and $B_{i-1}$ are the same size, inequality must hold (almost).
  ♦ If $B_{i-1}$ is smaller, it cannot hold.
Buckets for Clustering

- In place of “size” (number of 1’s) we use (an approximation to) the sum of the distances from all points to the centroid of their cluster.

- Merge consecutive buckets if the “size” of the merged bucket is less than the sum of the sizes of all later buckets.
Consequence of Merge Rule

- In a stable list of buckets, any two consecutive buckets are “bigger” than all smaller buckets.
- Thus, “sizes” grow exponentially.
- If there is a limit on total “size,” then the number of buckets is $O(\log N)$.
  - $N = \text{window size}$.
  - E.g., all points are in a fixed hypercube.
Outline of Algorithm

1. What do buckets look like?
   - Clusters at various levels, represented by centroids.

2. How do we merge buckets?
   - Keep # of clusters at each level small.

3. What happens when we query?
   - Final clustering of all clusters of all relevant buckets.
Organization of Buckets

◆ Each bucket consists of clusters at some number of levels.
  ♦ 4 levels in our examples.
◆ Clusters represented by:
  1. Location of centroid.
  2. Weight = number of points in the cluster.
  3. Cost = upper bound on sum of distances from member points to centroid.
Processing Buckets --- (1)

◆ Actions determined by $N$ (window size) and $k$ (desired number of clusters).
◆ Also uses a tuning parameter $\tau$ for which we use $1/4$ to simplify.
  ◆ $1/\tau$ is the number of levels of clusters.
Processing Buckets --- (2)

- Initialize a new bucket with $k$ new points.
  - Each is a cluster at level 0.
- If the timestamp of the oldest bucket is outside the window, delete that bucket.
Level-0 Clusters

◆ A single point \( p \) is represented by \((p, 1, 0)\).
◆ That is:
  1. A point is its own centroid.
  2. The cluster has one point.
  3. The sum of distances to the centroid is 0.
Merging Buckets --- (1)

_needed in two situations:

1. We have to process a query, which requires us to (temporarily) merge some tail of the bucket sequence.
2. We have just added a new (most recent) bucket and we need to check the rule about two consecutive buckets being “bigger” than all that follow.
Merging Buckets --- (2)

◆ **Step 1:** Take the union of the clusters at each level.

◆ **Step 2:** If the number of clusters (points) at level 0 is now more than $N^{1/4}$, cluster them into $k$ clusters.
  - These become clusters at level 1.

◆ **Steps 3, ...:** Repeat, going up the levels, if needed.
Representing New Clusters

- **Centroid** = weighted average of centroids of component clusters.
- **Weight** = sum of weights.
- **Cost** = sum over all component clusters of:
  1. Cost of component cluster.
  2. Weight of component times distance from its centroid to new centroid.
Example: New Centroid

weights

centroids

10 + (3,3)

new centroid

15 + (18,-2)

5 + (12,12)

+ (12,2)
Example: New Costs

10
+ (3,3)

old cost

+ (12,2)

true cost

5
+ (12,12)

added

15
+ (18,-2)

true cost
Queries

◆ Find all the buckets within the range of the query.
  ♦ The last bucket may be only partially within the range.
◆ Cluster all clusters at all levels into $k$ clusters.
◆ Return the $k$ centroids.
Error in Estimation

◆ Goal is to pick the $k$ centroids that minimize the $true$ cost (sum of distances from each point to its centroid).
◆ Since recorded “costs” are inexact, there can be a factor of 2 error at each level.
◆ Additional error because some of last bucket may not belong.
  ✷ But fraction of spurious points is small (why?).
Effect of Cost-Errors

1. May alter when buckets get combined.
   - Not really important.

2. Produce suboptimal clustering at any stage of the algorithm.
   - The real measure of how bad the output is.
Speedup of Algorithm

- As given, algorithm is slow.
  - Each new bucket causes $O(\log N)$ bucket-merger problems.

- A faster version allows the first bucket to have not $k$, but $N^{1/2}$ (or in general $N^{2^\tau}$) points.
  - A number of consequences, including slower queries, more space.