CS 493: Algorithms for Massive Data Sets | **Streaming Algorithms**
March 26, 2002 | **Scribe: Yitzhak Mandelbaum**

1 Review

The last class ended with a discussion of clustering algorithms. The focus was comparing the clusters produced by these algorithms to the optimal clusters.

1.1 k-Center Problem

Start with a collection of points. Divide them into k clusters. Then, choose a single point in each cluster as the center of the cluster. The goal is to minimize the maximum distance of any given point to its cluster center. The number of clusters, k, is given in advance.

1.2 k-Median Problem

A similar problem is the k-median problem. Here the goal is to minimize the sum of the distances between all points and the cluster center.

Suppose that you know R^*, the minimum possible distance of a point to its cluster center. Then, we can find a solution with maximum radius $2R^*$. Pick a point to be a cluster center and then add all points within a distance of $2R^*$. Repeat until all of the points belong to clusters.

Claim 1.1 *This algorithm will not produce more than k clusters.*

Proof. The pairwise distance between any 2 cluster centers is greater then $2R^*$. Therefore, in the optimal solution, each cluster center must appear in distinct clusters. So, if this algorithm can find $k+1$ clusters, then the optimal solution must have $k+1$ points in distinct clusters, and, therefore, at least $k+1$ clusters, which is a contradiction. ■

Sometimes, R^* is not provided. Instead, we "find" it by checking all possible R^*s. We can derive the different possibilities for R^* from the graph itself, by considering the distances between all pairs of points.

2 Streaming Algorithms

Notice that the above algorithm requires the entire point set to be known at the outset. However, there arise situations in which this assumption is not valid. Instead, we need to consider *streaming* algorithms, which do not make this assumption.

Definition 2.1 (Streaming Model) *In the streaming model of clustering, we assume that the points arrive in a data stream:*

$$p_1, p_2, \dots p_n$$

Having seen

$$p_1, p_2, \dots p_i$$

we maintain a clustering for the whole set.

The question, then, is how to effectively update this clustering upon receipt of p_{i+1} while using little space.

Let us assume that we can find the distance of p_{i+1} from any previously known point. We want to find an algorithm that represents the clustering by the k centers. Then, when we receive point p_{i+1}, we want to update the set of k centers in some way.

This representation of clustering is implicit. Given each center, we can find which points belong to which center by mapping all points to the nearest center.

The performance of the algorithm is measured with the following quantity:

$$\frac{\text{maximum radius of algorithm's clustering}}{\text{maximum radius of optimal clustering}}$$

(similar to the *competitive ratio*)

Now, supposing that we know the maximum radius of the optimal clustering, we can find a new clustering in an incremental fashion, as follows. Maintain clusters of radius $2R^*$. For each new point, check if the point lies in any of the existing clusters. If not, create a new cluster with the point as the center. If we don't know the optimal radius, then we can make a conservative guess: optimal $\geq$ guess. An incorrect guess can be detected, as the algorithm will produce too many clusters. Then, we revise our guess and merge clusters based on the new guess.

Details:

At the end of phase i:

1. Have $k+1$ points with pairwise distance $\geq r_i$.

2. Each cluster has radius $\leq 2r_i$.

Transition from phase $i \rightarrow i+1$:

```
$r_{i+1} = 2r_i$
merge clusters:
    Define a graph on the cluster centers by connecting any two centers
    whose distance is $\leq r_{i+1}$.
    Pick a point in this graph and merge all points (i.e. cluster centers)
    directly connected to this point.
```

Notice that the maximum cluster radius at the beginning of phase $i+1 \leq 2r_{i+1}$. The optimal radius $> \frac{r_i}{2}$.
Running phase $i+1$:

Start with set of centers, with each cluster's maximum radius $2r_{i+1}$. Given a new point, check the distance of the point from each cluster center. If the point is close enough to any center, then it belongs to the associated cluster. Otherwise, start a new cluster with the point as its center.
Invariants:

1. cluster centers have pointwise distance $\geq r_{i+1}$
2. radius of each cluster $\leq 2r_{i+1}$.

Performance Ratio:

$$\frac{\text{maximum radius of algorithm in phase } i+1}{\text{optimal max. radius}} = \frac{2r_{i+1}}{\frac{r_i}{2}} = 8$$

Notice that the algorithm above only needs to remember k centers. Also, it only starts new clusters and merges old ones. It never reassigns points from one cluster to another.

3 Streaming Algorithms and Frequency Moments

Definition 3.1 *Given a set of numbers $n_1, n_2, \ldots n_m$,*

$$m_i = \textit{the number of items of type } i$$

The k^{th} Frequency Moment is defined as:

$$F_k = \sum m_i^k$$

Then, for

$k = 0$, $F_0 =$ the number distinct items,
$k = 1$, $F_1 =$ the size of the list (i.e. n),
$k = 2$, $F_2 =$ a measure of skewness of the list.

In general, for $j > 1$, F_j is a measure of skew of items in the list.

Perhaps surprisingly, we can compute these values using a small amount of space. In the next lecture, we will examine estimating F_0 within a constant factor using a small amount of space. Now, we will examine F_1 and F_2.

Example 1 (Estimating F_1)

Naive approach: keep a counter. Uses $O(\log m)$ space, where m is the size of the stream.
Better: counter keeps only powers of 2, i.e.:

$$1, 2, 4, 8, 16, \ldots$$

When the counter has value 2^n it has some probability of incremementing to 2^{n+1}. In this way, we can use log of counter bits.

represented values:	$1 \ldots 2^{\log m}$
counter values:	$1 \ldots \log m$
counter size:	$O(\log \log m)$ bits

Note: X_i represents the value 2^{X_i}

$$X_0 = 0$$
$$X_{i+1} = \begin{cases} X_i + 1 & \text{with probability} \frac{1}{2^{X_i}} \\ X_i & \text{with probability} 1 - \frac{1}{2^{X_i}} \end{cases}$$

For the random variable X_N,

$$E[2^{X_N}] = N + 1$$
$$var[2^{X_N}] = \frac{N(N+1)}{2}$$

Now, keep a number of these counters. Then divide them into groups and take the average of each group. Finally, take the median of the averages as the value of F_1. Given c_1, c_2, the final estimate of F_1 is the median of $Y_1, Y_2, \ldots Y_{C_1}$, with Y_i the average of $Z_{i_1}, Z_{i_2}, \ldots Z_{i_{c_2}}$, with each Z_{i_j} one of the above counters.

Goal: Given ϵ, δ, compute Y such that $|Y - N| < \epsilon N$ with probability $1 - \delta$. In english: with high probability, estimate is within fraction ϵ of N.)

$$c_1 = 2 \lg \left(\frac{1}{\delta} \right)$$
$$c_2 = \frac{4}{\epsilon^2}$$

Analysis of Average:

$$E[2^{X_N}] = N + 1 = \mu$$
$$var[2^{X_N}] = \frac{N(N+1)}{2} \leq \frac{\mu^2}{2}$$

Taking c_2 copies of the estimator (counter) and averaging them, the variance is

$$var[\text{mean}] = \frac{\mu^2}{2c_2}$$

Then,

$$Pr[|\text{mean} - \mu| > \epsilon\mu] \le \frac{\mu^2}{2c_2\epsilon^2\mu^2} = \frac{1}{2c_2\epsilon^2}$$

substituting for c_2:

$$= \frac{1}{8}$$

The median serves as a good estimate because if the median is not in the range then at least $\frac{1}{2}$ of the estimators are "bad"(i.e. not in range). Each mean is "bad" with probability $\le \frac{1}{8}$
Chernoff Bound:

$$X = \sum x_i, \text{ where } E[X] = \mu$$
$$Pr[X > (1+\epsilon)\mu] \le \left(\frac{e^\epsilon}{(1+\epsilon)^{(1+\epsilon)}}\right)^\mu$$

Take x_i as an indicator variable for Y_i being bad: $x_i = \begin{cases} 0 & \text{if } Y_i \text{ bad} \\ 1 & \text{if } Y_i \text{ good} \end{cases}$

$$E[X] \le \frac{1}{8} \cdot \overbrace{2\lg\left(\frac{1}{\delta}\right)}^{c_1}$$

resulting in the bound

$$Pr[X > (1+\epsilon)\mu] \le \delta$$

Example 2 (Estimating F_2)

To review, $F_2 = \sum m_i^2$. The naive approach would use an estimator for each m_i. However, we can do better than that. Consider a hash function $h : i \to \{+1, -1\}$ with equal probability of $\frac{1}{2}$. h is random, yet consistent for any given i.

$$x_i = h(i)$$
$$X = \sum x_i m_i$$
$$Y = X^2$$
$$E[Y] = E[X^2] = F_2$$

$$
\begin{aligned}
E[Y] &= E[X^2] \\
&= E[(\sum x_i m_i)^2] \\
&= E[\sum x_i^2 m_i^2 + 2 \sum_{i<j} x_i x_j m_i m_j]
\end{aligned}
$$

Now, x_i^2 is always 1. Also, the second term is 0 because the expectance of x_i is 0, so

$$
= \sum m_i^2
$$

Now we compute $var(Y)$:

$$
\begin{aligned}
E[Y^2] &= E[X^4] \\
&= \sum x_i^4 m_i^4 + 6 \sum x_i^2 x_j^2 m_i^2 m_j^2 \qquad \text{(all other terms disappear)} \\
&= F_4 + 6F_2 \\
var(Y) &= E[Y^2] - (E[Y])^2 \\
&= 4 \sum m_i^2 m_j^2 \\
&= 2F_2^2
\end{aligned}
$$

Now we can use the median of means machinery from the previous example to obtain an accurate estimate of F_2.