Streaming Euclidean k-median and k-means with $o(\log n)$ Space

Vincent Cohen-Addad

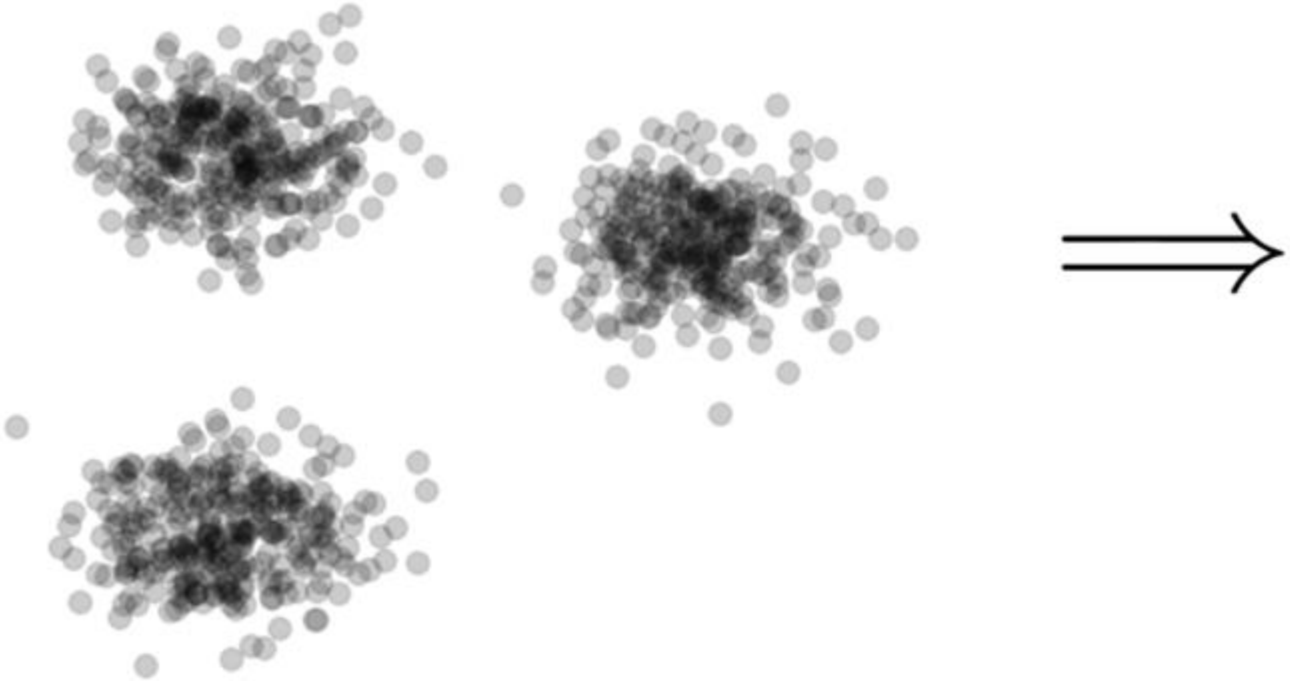
David P. Woodruff

Samson Zhou

Goal: Cluster a stream of n points using $o(\log n)$ space

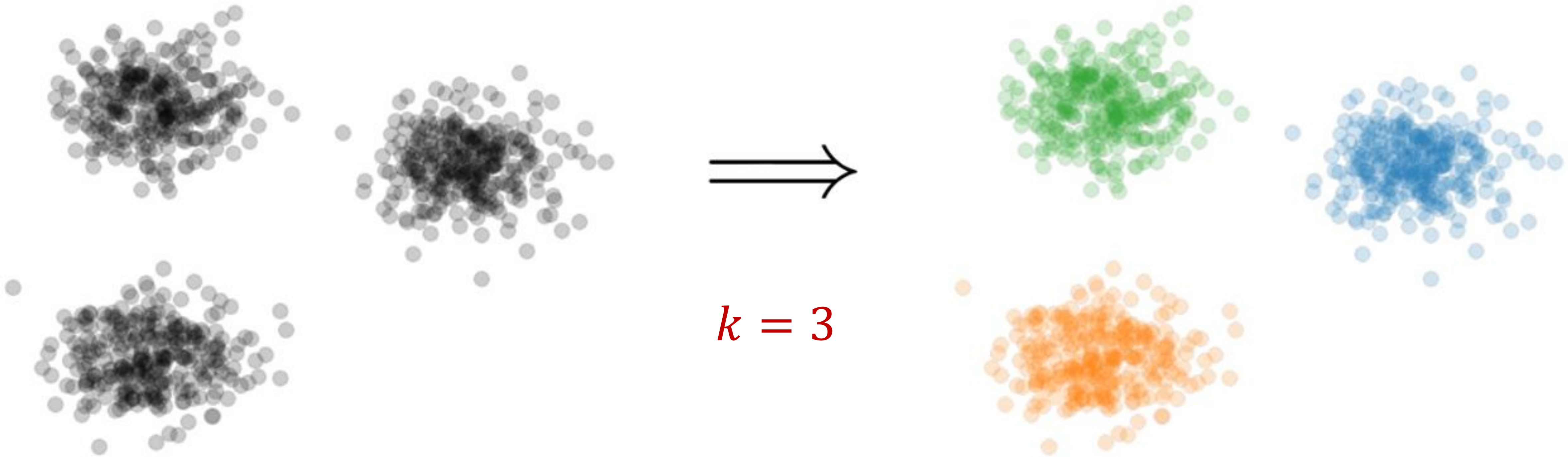
Clustering

- Goal: Given input dataset X, partition X so that “similar” points are in the same cluster and “different” points are in different clusters



k-Clustering

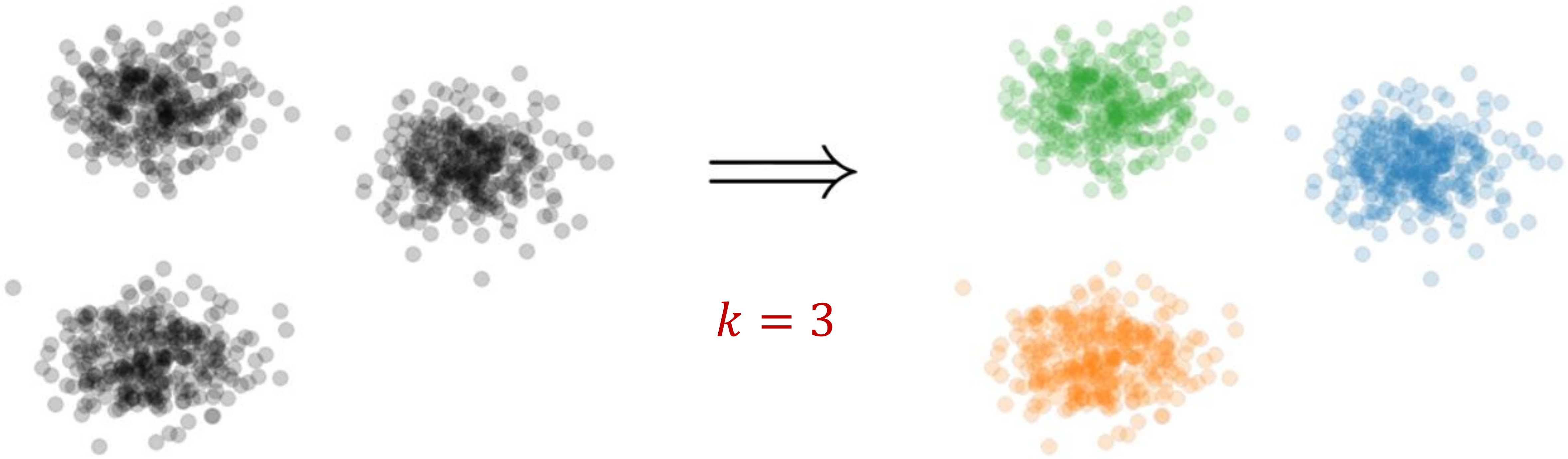
- Goal: Given input dataset X, partition X so that "similar" points are in the same cluster and "different" points are in different clusters
- There can be at most k different clusters



$k = 3$

k-Clustering

- Question: How do we measure the "quality" of each clustering?



k-Clustering

- Question: How do we measure the "quality" of each clustering?
- Assign a "center" c_i to each cluster
- Have a cost function induced by c_i for all of the points P_i assigned to cluster i

k-Clustering

- Question: How do we measure the "quality" of each clustering?
- Assign a "center" c_i to each cluster
- Have a cost function induced by c_i for all of the points P_i assigned to cluster i
  - Assume points are in metric space with distance function $\mathrm{dist}(\cdot,\cdot)$
  - Define $\mathrm{Cost}(P_i, c_i)$ to be a function of $\{\mathrm{dist}(x, c_i)\}_{x \in P_i}$

k-Clustering

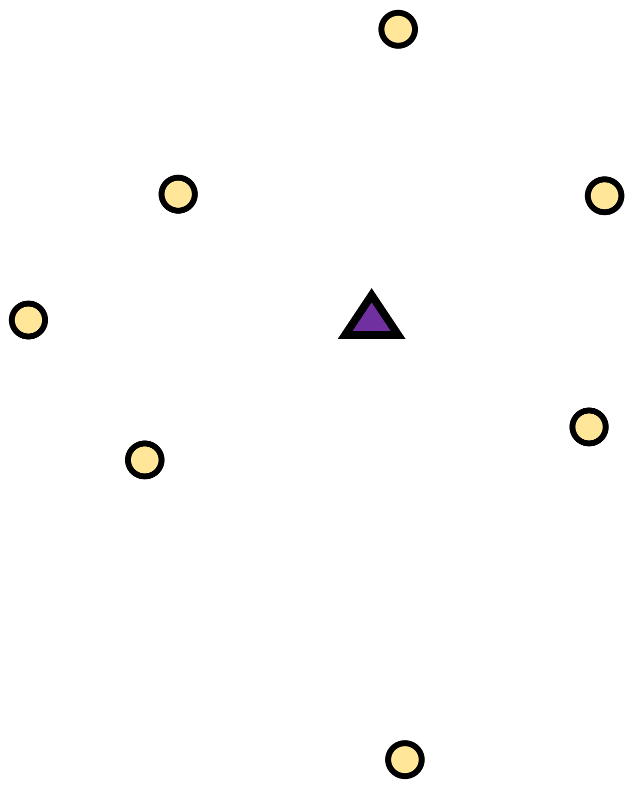
- Question: How do we measure the "quality" of each clustering?

- Have a cost function induced by c_i for all of the points P_i assigned to cluster i
  - Define $\mathrm{Cost}(P_i, c_i)$ to be a function of $\{\mathrm{dist}(x, c_i)\}_{x \in P_i}$

- Suppose the set of centers is $C = \{c_1, \ldots, c_k\}$
  - Define clustering cost $\mathrm{Cost}(X, C)$ to be a function of $\{\mathrm{dist}(x, C)\}_{x \in C}$

k-Clustering

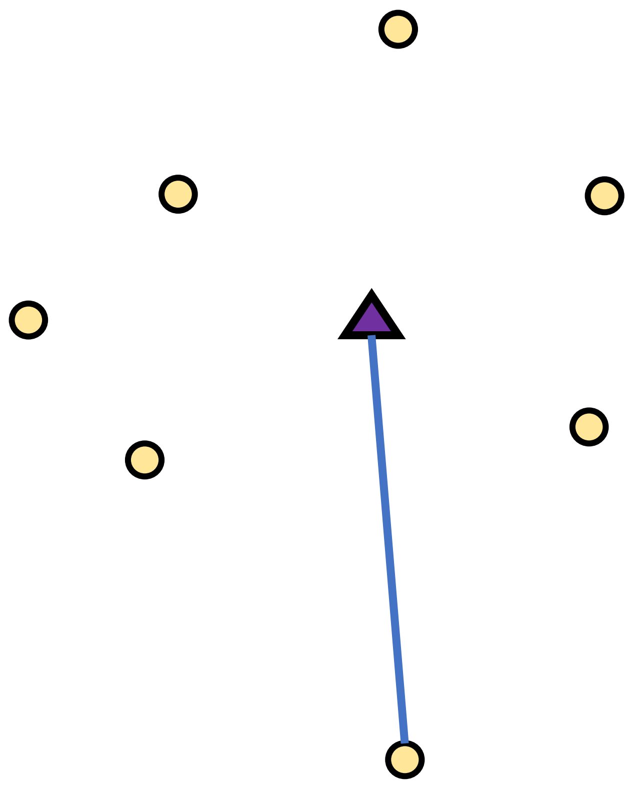
- Define clustering cost $\mathrm{Cost}(X, C)$ to be a function of $\{\mathrm{dist}(x, C)\}_{x \in C}$



k-Clustering

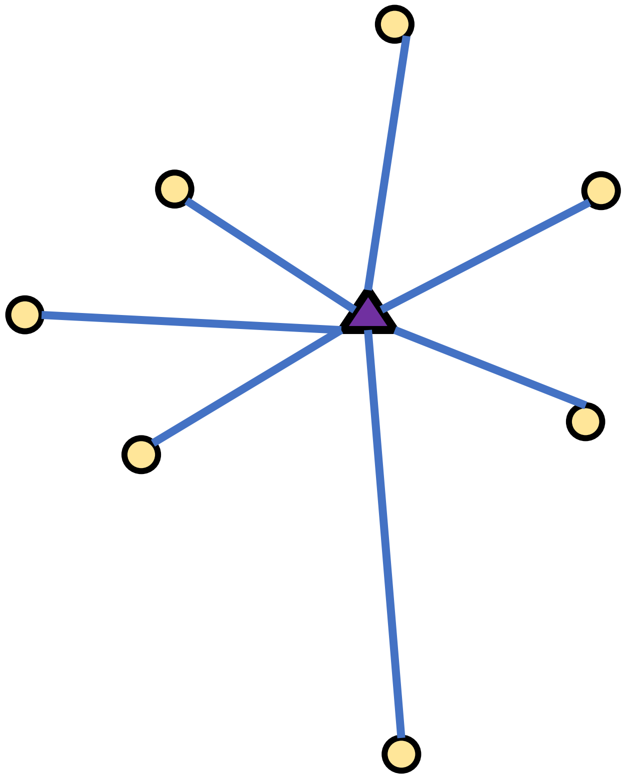
- Define clustering cost $\mathrm{Cost}(X, C)$ to be a function of $\{\mathrm{dist}(x, C)\}_{x \in C}$

- k-center: $\mathrm{Cost}(X, C) = \max_{x \in X} \mathrm{dist}(x, C)$



k-Clustering

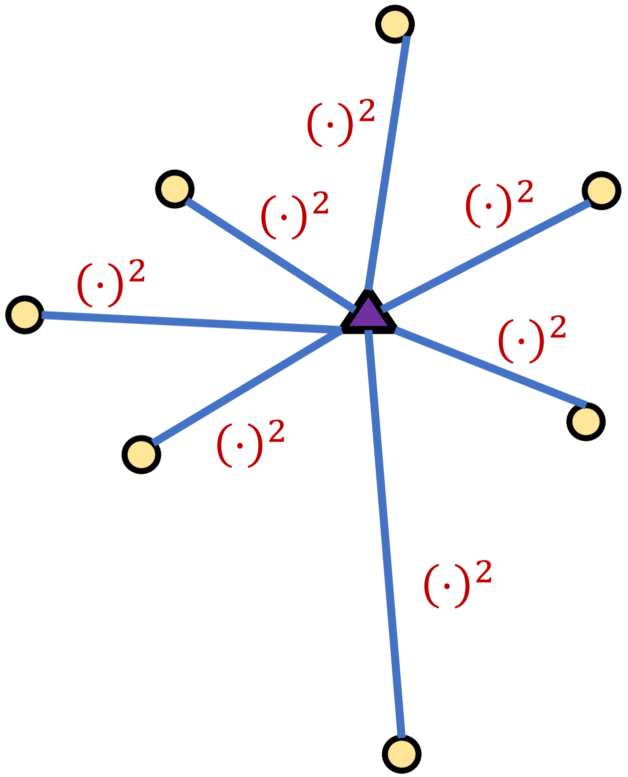
- Define clustering cost $\mathrm{Cost}(X, C)$ to be a function of $\{\mathrm{dist}(x, C)\}_{x \in C}$



- k-center: $\mathrm{Cost}(X, C) = \max_{x \in X} \mathrm{dist}(x, C)$
- k-median: $\mathrm{Cost}(X, C) = \sum_{x \in X} \mathrm{dist}(x, C)$

k-Clustering

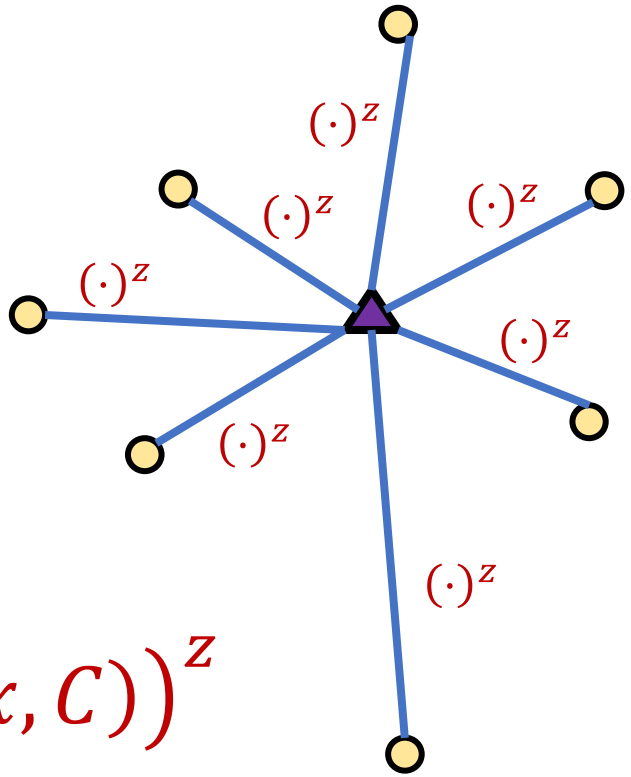
- Define clustering cost $\mathrm{Cost}(X, C)$ to be a function of $\{\mathrm{dist}(x, C)\}_{x \in C}$
- k-center: $\mathrm{Cost}(X, C) = \max_{x \in X} \mathrm{dist}(x, C)$
- k-median: $\mathrm{Cost}(X, C) = \sum_{x \in X} \mathrm{dist}(x, C)$
- k-means: $\mathrm{Cost}(X, C) = \sum_{x \in X} \left(\mathrm{dist}(x, C)\right)^2$



k-Clustering

- Define clustering cost $\mathrm{Cost}(X, C)$ to be a function of $\{\mathrm{dist}(x, C)\}_{x \in C}$

- k-center: $\mathrm{Cost}(X, C) = \max_{x \in X} \mathrm{dist}(x, C)$
- k-median: $\mathrm{Cost}(X, C) = \sum_{x \in X} \mathrm{dist}(x, C)$
- k-means: $\mathrm{Cost}(X, C) = \sum_{x \in X} \left(\mathrm{dist}(x, C)\right)^2$
- (k, z)-clustering: $\mathrm{Cost}(X, C) = \sum_{x \in X} \left(\mathrm{dist}(x, C)\right)^z$



Euclidean k-Clustering

- For Euclidean k-clustering, input points $X = x_1, \ldots, x_n$ are in $\mathbb{R}^d$ (for us, they will be in $[\Delta]^d := \{1,2, \ldots, \Delta\}^d$)

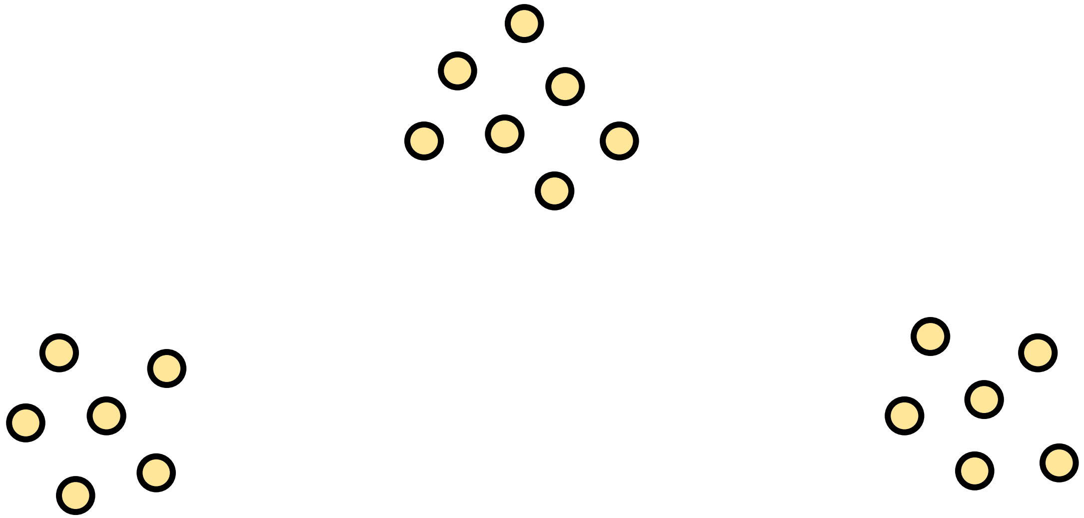
- $\mathrm{dist}(x, y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_d - y_d)^2}$ is the Euclidean distance

- (k, z)-clustering problem:

$$\min_{C:|C| \leq k} \mathrm{Cost}(X, C) = \min_{C:|C| \leq k} \Sigma_{x \in X} \left(\mathrm{dist}(x, C)\right)^z$$

The Streaming Model

- Input: Updates to an underlying data set X that arrive sequentially
- Output: Evaluation (or approximation) of a given function
- Goal: Use space *sublinear* in the size n of the input X



Goal: Cluster a stream of n points using $o(\log n)$ space

Our Results (Insertion-Only)

- There exists a one-pass algorithm on insertion-only streams that outputs $(1+\varepsilon)$-approximation for (k,z)-clustering for *all times in the stream* and uses $\tilde{O}\left(\frac{dk}{\varepsilon^2}\right)\cdot\min\left(k,\frac{1}{\varepsilon^z}\right)\cdot\text{poly}(\log\log n\Delta)$ words of space

- Our algorithm outputs $(1+\varepsilon)$-coreset constructions for (k,z)-clustering for *all times in the stream*

Our Results (Insertion-Deletion Impossibility)

- Any one-pass algorithm on insertion-deletion streams that outputs a 2-approximation to the (k, z)-clustering cost *at all times* in the stream with $d = \Omega(\log n)$ must use $\Omega(\log^2 n)$ bits of space

- Any one-pass algorithm on insertion-deletion streams that outputs a 2-approximation to the (k, z)-clustering cost *from a weighted subset of the input* must use $\Omega(\log^2 n)$ bits of space

Our Results (Insertion-Deletion Two-Pass)

- There exists a two-pass algorithm on insertion-deletion streams that outputs a $(1+\varepsilon)$-coreset construction for k-median and k-means clustering that uses $\tilde{O}\left(\frac{1}{\varepsilon^2}\right) \cdot \text{poly}(d, k, \log\log n\Delta)$ words of space

- Result generalizes to $z \in [1,2]$

Our Results (Sum of the Online Sensitivities)

- Sum of the online sensitivities of a set of n points in $\mathbb{R}^d$ for (k, z)-clustering is at most $O(k \log^2(nd\Delta))$

Coreset

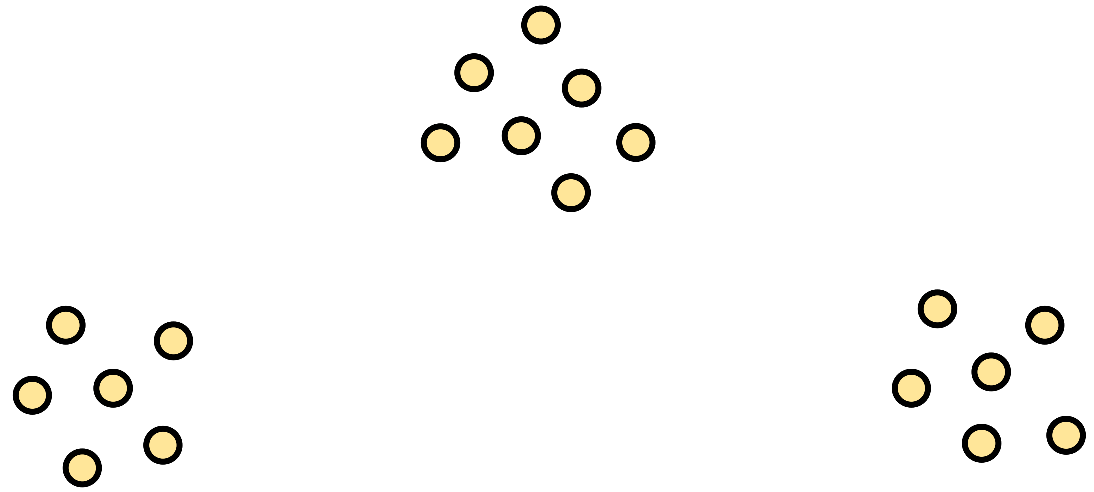
- Subset X' of representative points of X for a specific clustering objective

- $\text{Cost}(X, C) \approx \text{Cost}(X', C)$ for all sets C with $|C| = k$

Coreset

- Subset X' of representative points of X for a specific clustering objective

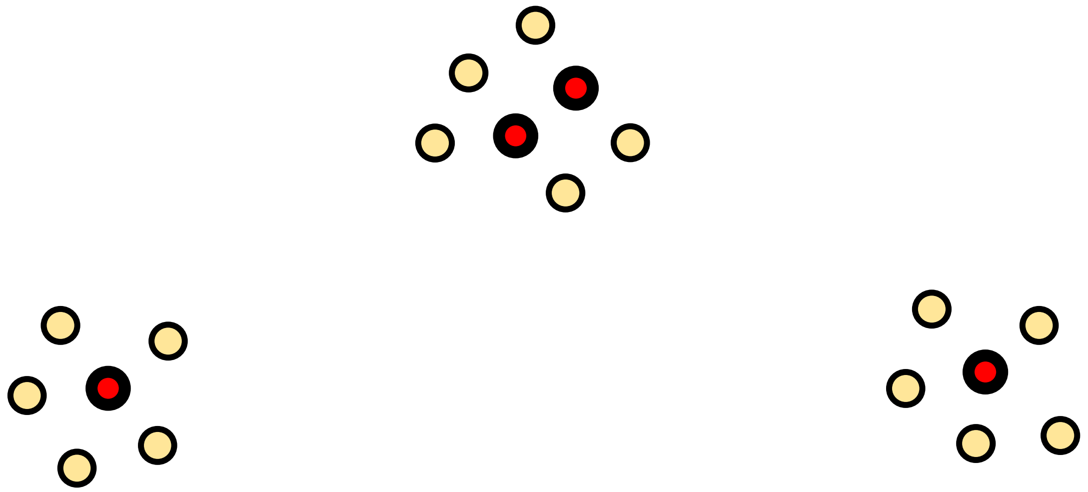
- $\text{Cost}(X, C) \approx \text{Cost}(X', C)$ for all sets C with $|C| = k$



Coreset

- Subset X' of representative points of X for a specific clustering objective

- $\text{Cost}(X, C) \approx \text{Cost}(X', C)$ for all sets C with $|C| = k$



Coreset (Formal Definition)

- Given a set X and an accuracy parameter $\varepsilon > 0$, we say a set X' with weight function w is an $(1+\varepsilon)$-multiplicative coreset for a cost function Cost, if for all queries C with $|C| \leq k$, we have

$$(1-\varepsilon)\mathrm{Cost}(X,C) \leq \mathrm{Cost}(X',C,w) \leq (1+\varepsilon)\mathrm{Cost}(X,C)$$

↑

(k,z)-clustering: $\mathrm{Cost}(X',C,w) = \sum_{x\in X'} w(x) \cdot \left(\mathrm{dist}(x,C)\right)^z$

Coreset Constructions

- Let $\tilde{O}(f)$ denote $f \cdot \text{polylog}(f)$
- For (k, z)-clustering, there exist coreset constructions that only require $\tilde{O}\left(\frac{k}{\varepsilon^2}\right) \cdot \min\left(k, \frac{1}{\varepsilon^z}\right)$ weighted points of the input [Cohen-AddadLarsenSaulpicSchweighelshohn22]
- *Independent* of input size n

(k, z)-Clustering in the Streaming Model

- Merge-and-reduce framework

- Suppose there exists a $(1+\varepsilon)$-coreset construction for (k, z)-clustering that uses $f\left(k, \frac{1}{\varepsilon}\right)$ weighted input points

  $\tilde{O}\left(\frac{k^2}{\varepsilon^2}\right)$

- Partition the stream into blocks containing $f\left(k, \frac{\log n}{\varepsilon}\right)$ points

(k, z)-Clustering in the Streaming Model

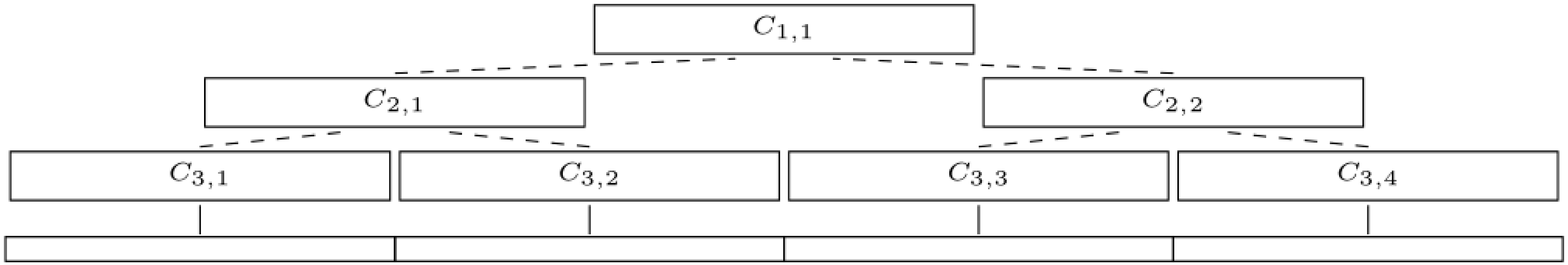
- Partition the stream into blocks containing $f\left(k, \frac{\log n}{\varepsilon}\right)$ points
- Create a $\left(1 + \frac{\varepsilon}{\log n}\right)$-coreset for each block
- Create a $\left(1 + \frac{\varepsilon}{\log n}\right)$-coreset for the set of points formed by the union of two coresets for each block

Reduce

Merge

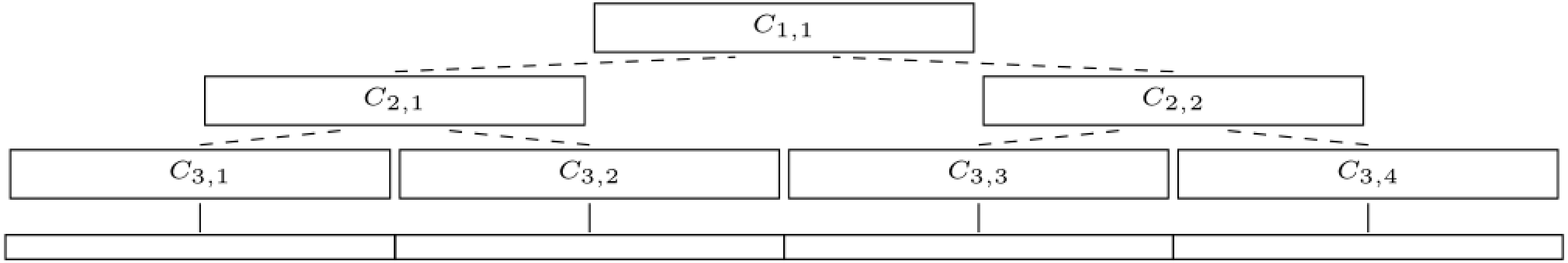
(k, z)-Clustering in the Streaming Model

- Partition the stream into blocks containing $f\left(k, \frac{\log n}{\varepsilon}\right)$ points
- Create a $\left(1+\frac{\varepsilon}{\log n}\right)$-coreset for each block
- Create a $\left(1+\frac{\varepsilon}{\log n}\right)$-coreset for the set of points formed by the union of two coresets for each block



(k, z)-Clustering in the Streaming Model

- There are $O(\log n)$ levels
- Each coreset is a $\left(1 + \frac{\varepsilon}{\log n}\right)$-coreset of two coresets
- Total approximation is $\left(1 + \frac{\varepsilon}{\log n}\right)^{\log n} = (1 + O(\varepsilon))$



(k, z)-Clustering in the Streaming Model

- Suppose there exists a $(1+\varepsilon)$-coreset construction for (k, z)-clustering that uses $f\left(k, \frac{1}{\varepsilon}\right)$ weighted input points
- Partition the stream into blocks containing $f\left(k, \frac{\log n}{\varepsilon}\right)$ points
- Total space is $f\left(k, \frac{\log n}{\varepsilon}\right) \cdot O(\log n)$ points

For k-means clustering, this is $\tilde{O}\left(\frac{k^2}{\varepsilon^2} \cdot \log^3 n\right)$ points

(k,z)-Clustering in the Streaming Model

- Suppose there exists a $(1+\varepsilon)$-coreset construction for (k,z)-clustering that uses $f\left(k,\frac{1}{\varepsilon}\right)$ weighted input points
- Partition the stream into blocks containing $f\left(k,\frac{\log n}{\varepsilon}\right)$ points
- Total space is $f\left(k,\frac{\log n}{\varepsilon}\right)\cdot O(\log n)$ points

> - For (k,z)-clustering, there exist coreset constructions that only require $\tilde{O}\left(\frac{k}{\varepsilon^2}\right)\cdot\min\left(k,\frac{1}{\varepsilon^z}\right)$ weighted points of the input [Cohen-AddadLarsenSaulpicSchweighelshohn22]

(k,z)-Clustering in the Streaming Model

- Suppose there exists a $(1+\varepsilon)$-coreset construction for (k,z)-clustering that uses $f\left(k,\frac{1}{\varepsilon}\right)$ weighted input points
- Partition the stream into blocks containing $f\left(k,\frac{\log n}{\varepsilon}\right)$ points
- Total space is $f\left(k,\frac{\log n}{\varepsilon}\right)\cdot O(\log n)$ points

Do there exist streaming algorithms for (k,z)-clustering that use $o(\log n)$ words of space?

Streaming algorithm	Words of Memory
[HK07], $z \in \{1, 2\}$	$\tilde{\mathcal{O}}\left(\frac{dk^{1+z}}{\varepsilon^{\mathcal{O}(d)}} \log^{d+z} n\right)$
[HM04], $z \in \{1, 2\}$	$\tilde{\mathcal{O}}\left(\frac{dk}{\varepsilon^d} \log^{2d+2} n\right)$
[Che09], $z \in \{1, 2\}$	$\tilde{\mathcal{O}}\left(\frac{d^2k^2}{\varepsilon^2} \log^8 n\right)$
[FL11], $z \in \{1, 2\}$	$\tilde{\mathcal{O}}\left(\frac{d^2k}{\varepsilon^{2z}} \log^{1+2z} n\right)$
Sensitivity and rejection sampling [BFLR19]	$\tilde{\mathcal{O}}\left(\frac{d^2k^2}{\varepsilon^2} \log n\right)$
Online sensitivity sampling, i.e., Theorem 3.5	$\tilde{\mathcal{O}}\left(\frac{d^2k^2}{\varepsilon^2} \log n\right)$
Merge-and-reduce with coreset of [CLSS22]	$\tilde{\mathcal{O}}\left(\frac{dk}{\varepsilon^2} \log^4 n\right) \cdot \min\left(\frac{1}{\varepsilon^z}, k\right)$
This work, i.e., Theorem 1.1	$\tilde{\mathcal{O}}\left(\frac{dk}{\varepsilon^2}\right) \cdot \min\left(\frac{1}{\varepsilon^z}, k\right) \cdot \text{poly}(\log\log n)$

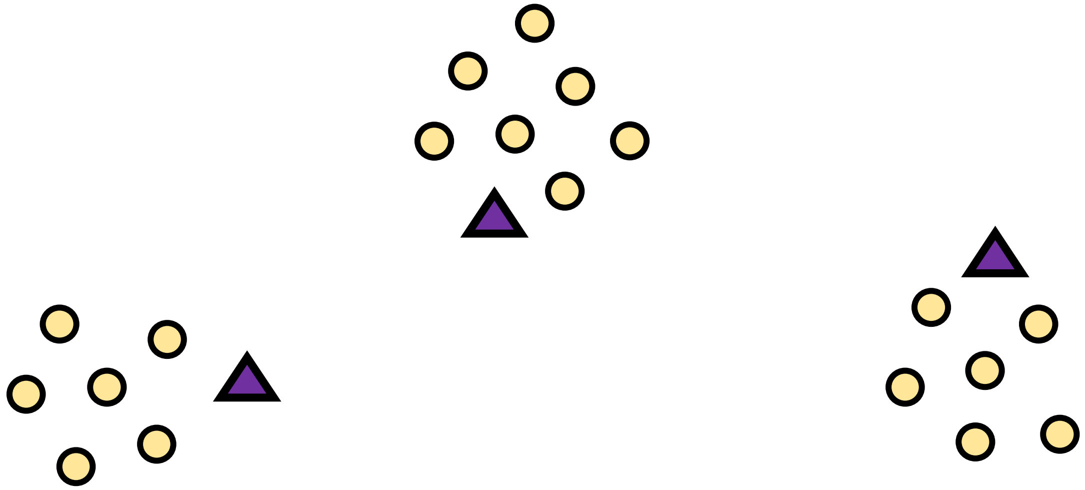
Format

- Part 1: Background
- Part 2: Insertion-Only Streams
- Part 3: k-Median on Dynamic Streams
- Part 4: (k, z)-Clustering on Dynamic Streams

Questions?

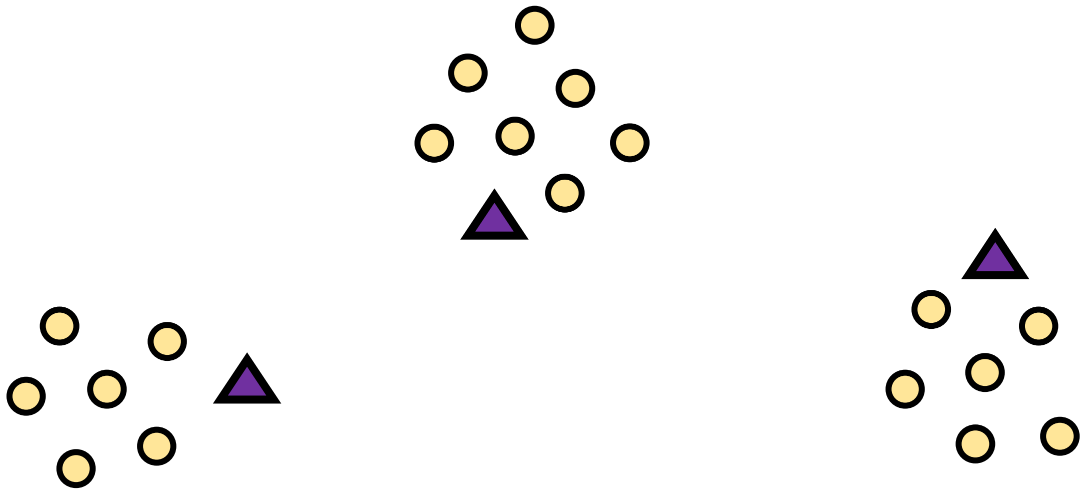
Coreset Construction and Sampling

- Consider a fixed set X and a fixed set C of k centers, which induces a fixed cost $\mathrm{Cost}(X, C)$



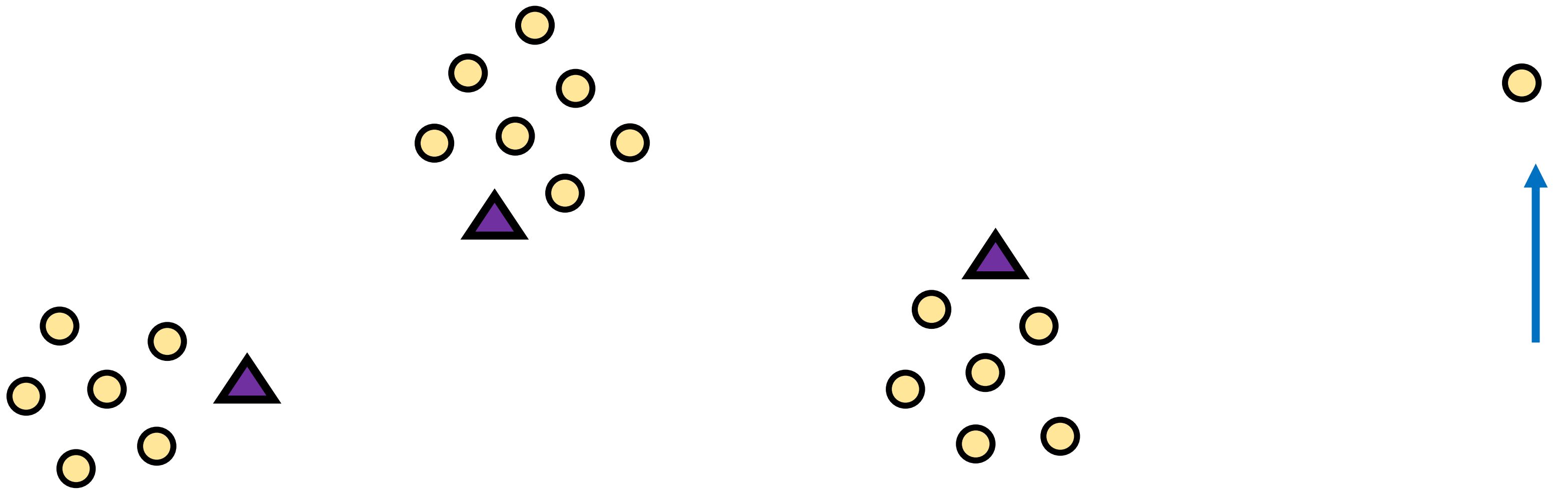
Coreset Construction and Sampling

- Consider a fixed set X and a fixed set C of k centers, which induces a fixed cost $\mathrm{Cost}(X, C)$
- A simple way to obtain X' with $\mathrm{Cost}(X', C) \approx \mathrm{Cost}(X, C)$ is to uniformly sample points of X into X'



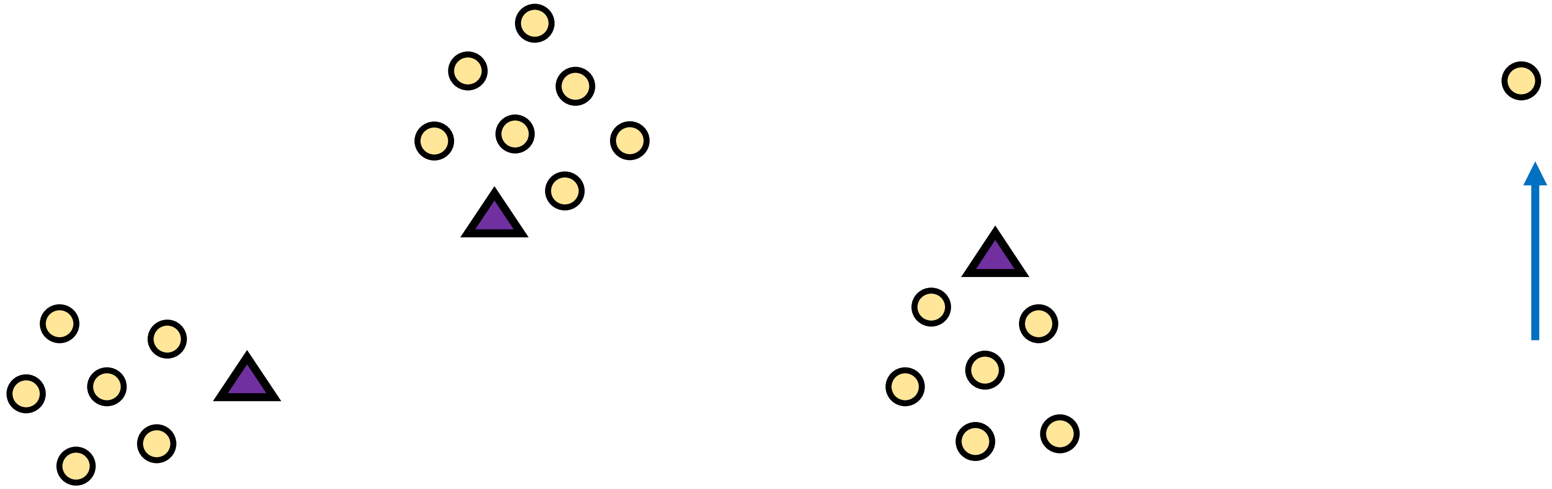
Coreset Construction and Sampling

- Consider a fixed set X and a fixed set C of k centers, which induces a fixed cost $\mathrm{Cost}(X, C)$
- Uniform sampling needs a lot of samples if there is a single point that greatly contributes to $\mathrm{Cost}(X, C)$



Coreset Construction and Sampling

- Fix: Importance sampling, sample each point $x \in X$ into X' with probability proportional $\mathrm{Cost}(x, C)$, i.e., $\mathrm{Cost}(x, C)/\mathrm{Cost}(X, C)$



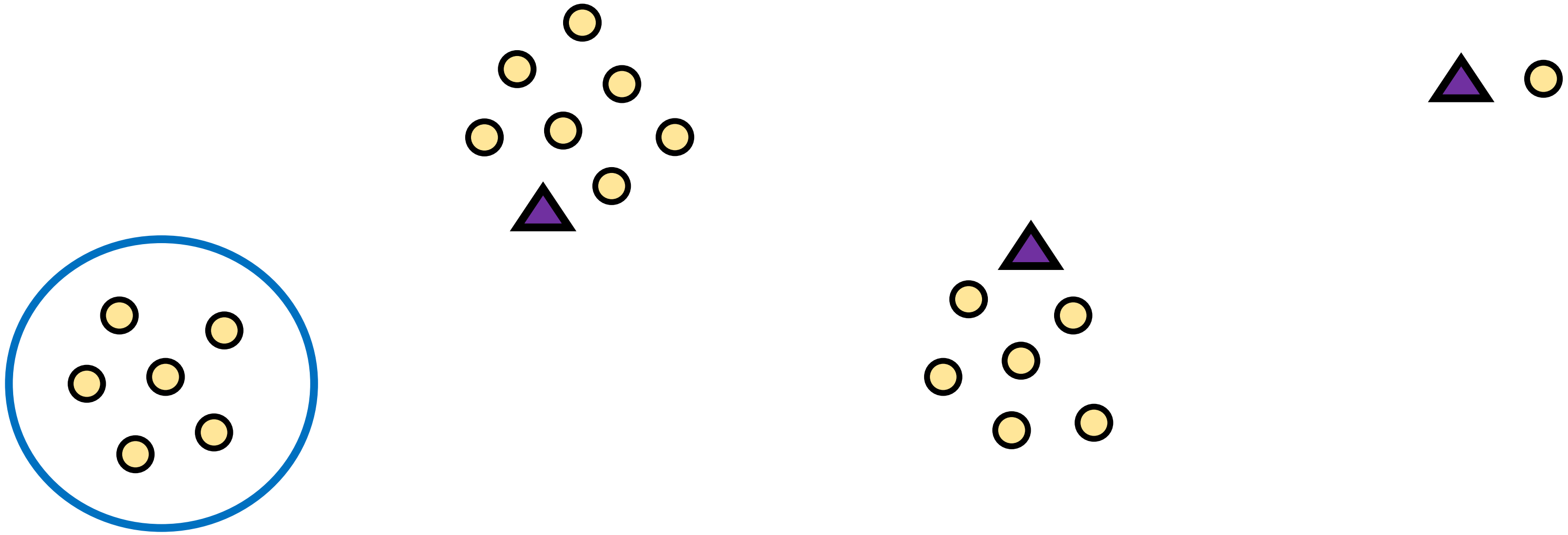
Coreset Construction and Sampling

- Fix: Importance sampling, sample each point $x \in X$ into X' with probability proportional $\mathrm{Cost}(x, C)$, i.e., $\mathrm{Cost}(x, C)/\mathrm{Cost}(X, C)$

- Importance sampling only needs X' to have size $O\left(\frac{1}{\varepsilon^2}\right)$ to achieve $(1 + \varepsilon)$-approximation to $\mathrm{Cost}(X, C)$

Coreset Construction and Sampling

- Importance sampling only needs X' to have size $O\left(\frac{1}{\varepsilon^2}\right)$ to achieve $(1+\varepsilon)$-approximation to $\text{Cost}(X, C)$
- What about a different choice C of k centers?



Coreset Construction and Sampling

- Importance sampling only needs X' to have size $O\left(\frac{1}{\varepsilon^2}\right)$ to achieve $(1+\varepsilon)$-approximation to $\text{Cost}(X, C)$
- To handle all possible sets of k centers:
  - Need to sample each point x with probability $\max_C \frac{\text{Cost}(x,C)}{\text{Cost}(X,C)}$ instead of $\frac{\text{Cost}(x,C)}{\text{Cost}(X,C)}$
  - Need to union bound over a net of all possible sets of k centers

Coreset Construction and Sampling

- Importance sampling only needs X' to have size $O\left(\frac{1}{\varepsilon^2}\right)$ to achieve $(1+\varepsilon)$-approximation to $\text{Cost}(X, C)$
- To handle all possible sets of k centers:
  - Need to sample each point x with probability $\max_C \frac{\text{Cost}(x,C)}{\text{Cost}(X,C)}$ instead of $\frac{\text{Cost}(x,C)}{\text{Cost}(X,C)}$
  - Need to union bound over a net of all possible sets of k centers

Net with size $\left(\frac{n\Delta}{\varepsilon}\right)^{O(kd)}$

Sensitivity Sampling

- The quantity $s(x) = \max_C \frac{\text{Cost}(x,C)}{\text{Cost}(X,C)}$ is called the *sensitivity* of x and intuitively measures how "important" the point x is

- The *total sensitivity* of X is $\sum_{x \in X} s(x)$ and quantifies how many points will be sampled into X' through importance/sensitivity sampling (before the union bound)

Online Sensitivity

- In a data stream, computing/approximating sensitivity $s(x) = \max_C \frac{\text{Cost}(x,C)}{\text{Cost}(X,C)}$ requires seeing the entire dataset X, but then it is too late to sample x

- We define the *online sensitivity* of x_t with respect to a stream $x_1, \dots, x_n$ to be $\varphi(x_t) = \max_C \frac{\text{Cost}(x_t,C)}{\text{Cost}(X_t,C)}$, where $X_t = x_1, \dots, x_t$, which intuitively measures how "important" the point x is *SO FAR*

Online Sensitivity

- Streaming algorithm: sample each point x_t with probability $p(x_t) = \min\left(1, \frac{kd}{\varepsilon^2} \cdot \text{polylog}(n\Delta) \cdot \varphi(x_t)\right)$

- How to compute (or approximate) $\varphi(x_t)$?

Online Sensitivity

- Observation: we can use a $(1+\varepsilon)$-coreset to obtain a $(1+\varepsilon)$-approximation to $\varphi(x_t)$
- Use samples obtained from online sensitivity sampling at each time $t-1$ to obtain a $(1+\varepsilon)$-approximation to $\varphi(x_t)$
- Can then perform online sensitivity sampling at time t and by induction, at all times in the stream

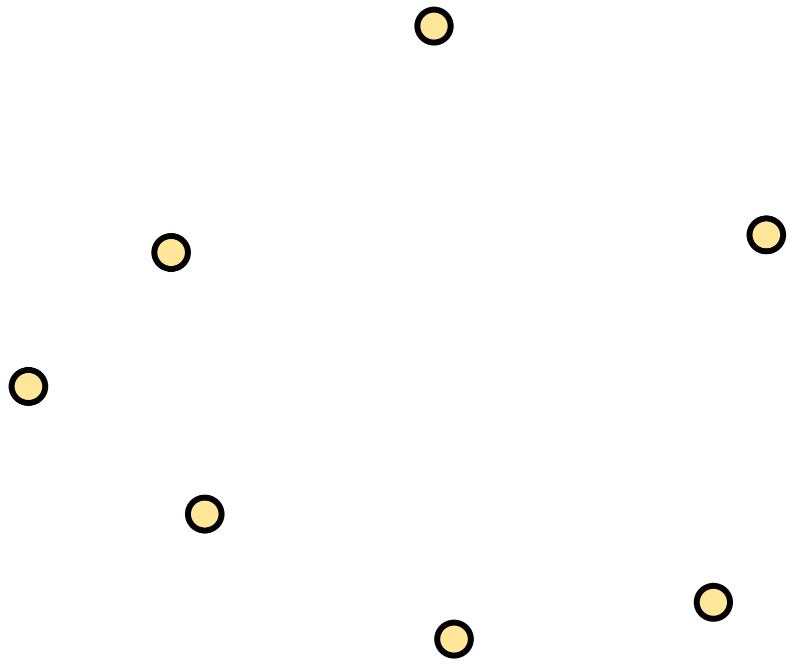
Online Sensitivity

- Streaming algorithm: sample each point x_t with probability $p(x_t) = \min\left(1, \frac{kd}{\varepsilon^2} \cdot \text{polylog}(n\Delta) \cdot \varphi(x_t)\right)$

- Given our new bounds on total sensitivity, we get a coreset of size $\sum_t p(x_t) = \frac{k^2 d}{\varepsilon^2} \cdot \text{polylog}(n\Delta)$

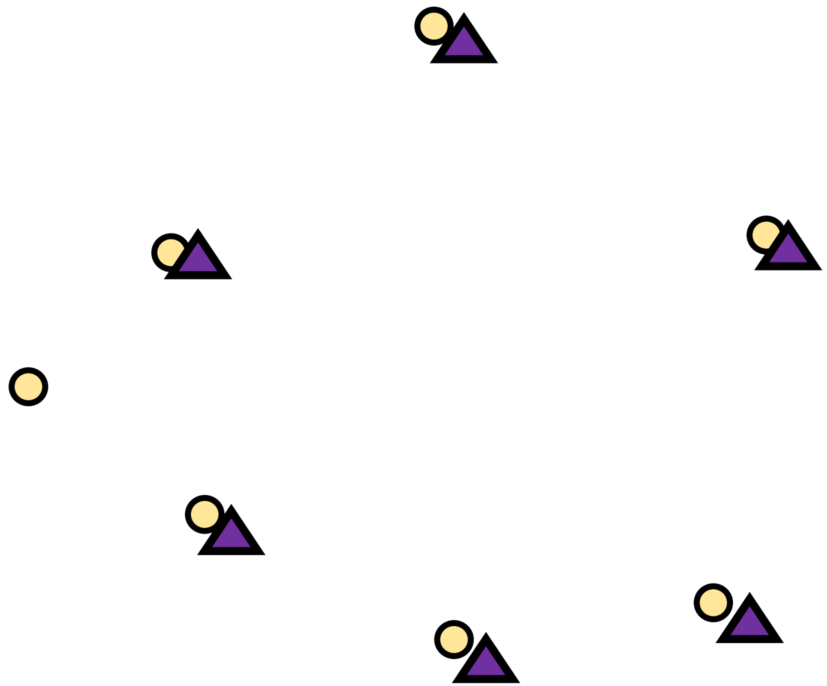
- Sampling is done online, can view as a new stream X'

$$\varphi(x_t) = \max_{C:|C|\leq k} \frac{\text{Cost}(x_t, C)}{\text{Cost}(X_t, C)} = \max_{C:|C|\leq k} \frac{\text{Cost}(x_t, C)}{\sum_{i=1}^{t} \text{Cost}(x_i, C)}$$

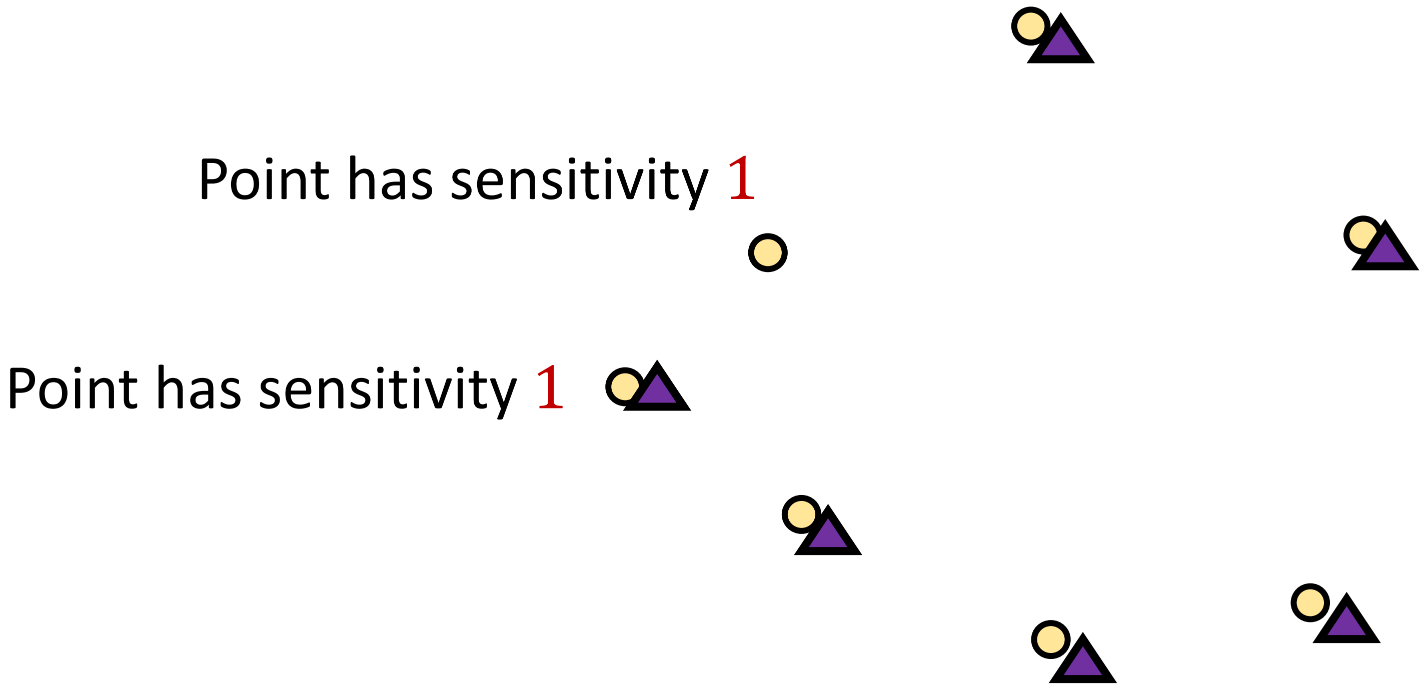


$$\varphi(x_t) = \max_{C:|C|\le k} \frac{\text{Cost}(x_t, C)}{\text{Cost}(X_t, C)} = \max_{C:|C|\le k} \frac{\text{Cost}(x_t, C)}{\sum_{i=1}^{t} \text{Cost}(x_i, C)}$$

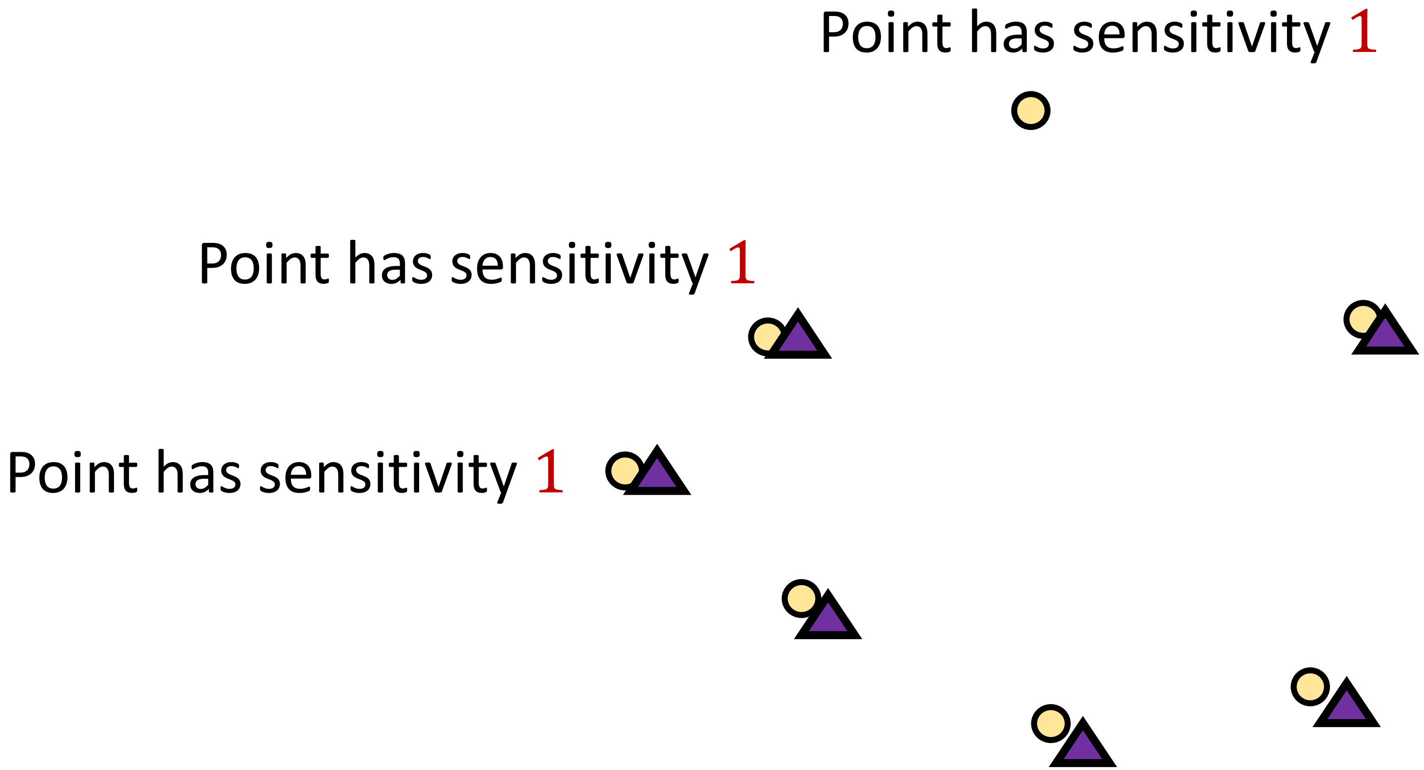
Point has sensitivity 1



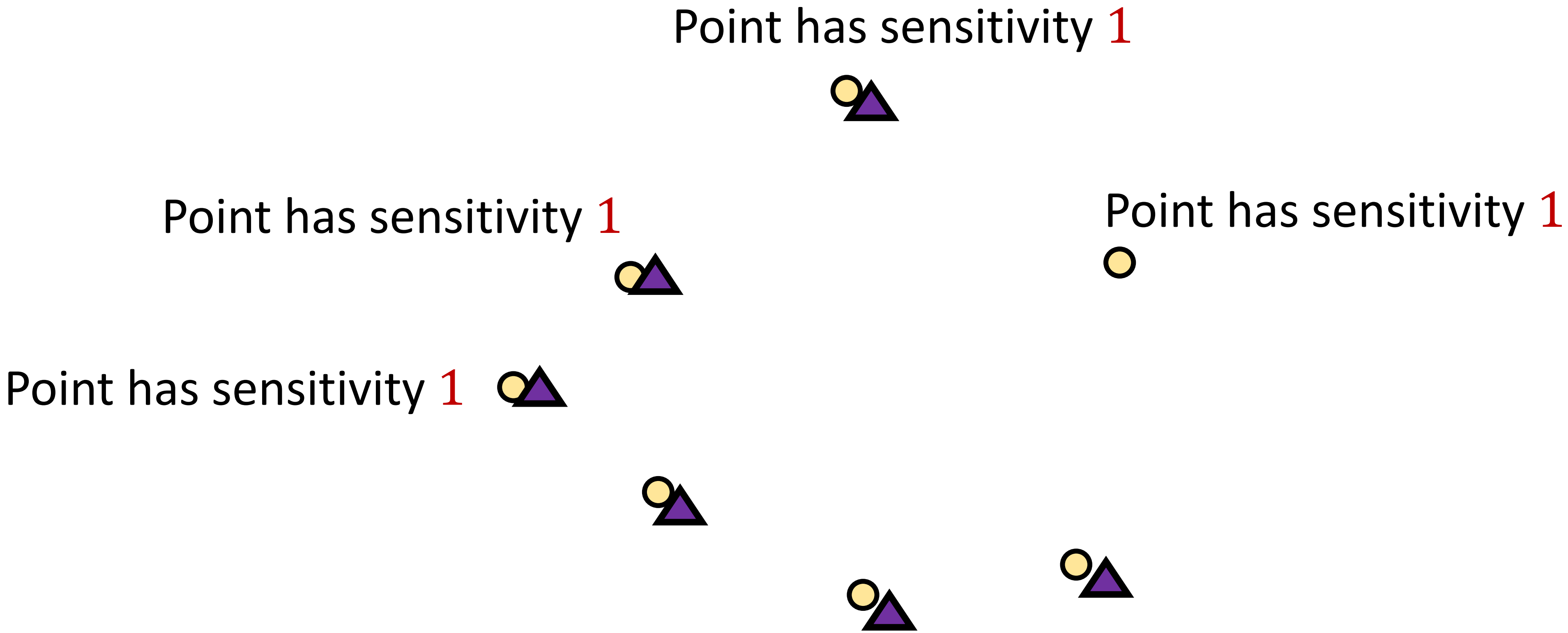
$$\varphi(x_t) = \max_{C:|C|\leq k} \frac{\text{Cost}(x_t, C)}{\text{Cost}(X_t, C)} = \max_{C:|C|\leq k} \frac{\text{Cost}(x_t, C)}{\sum_{i=1}^{t} \text{Cost}(x_i, C)}$$



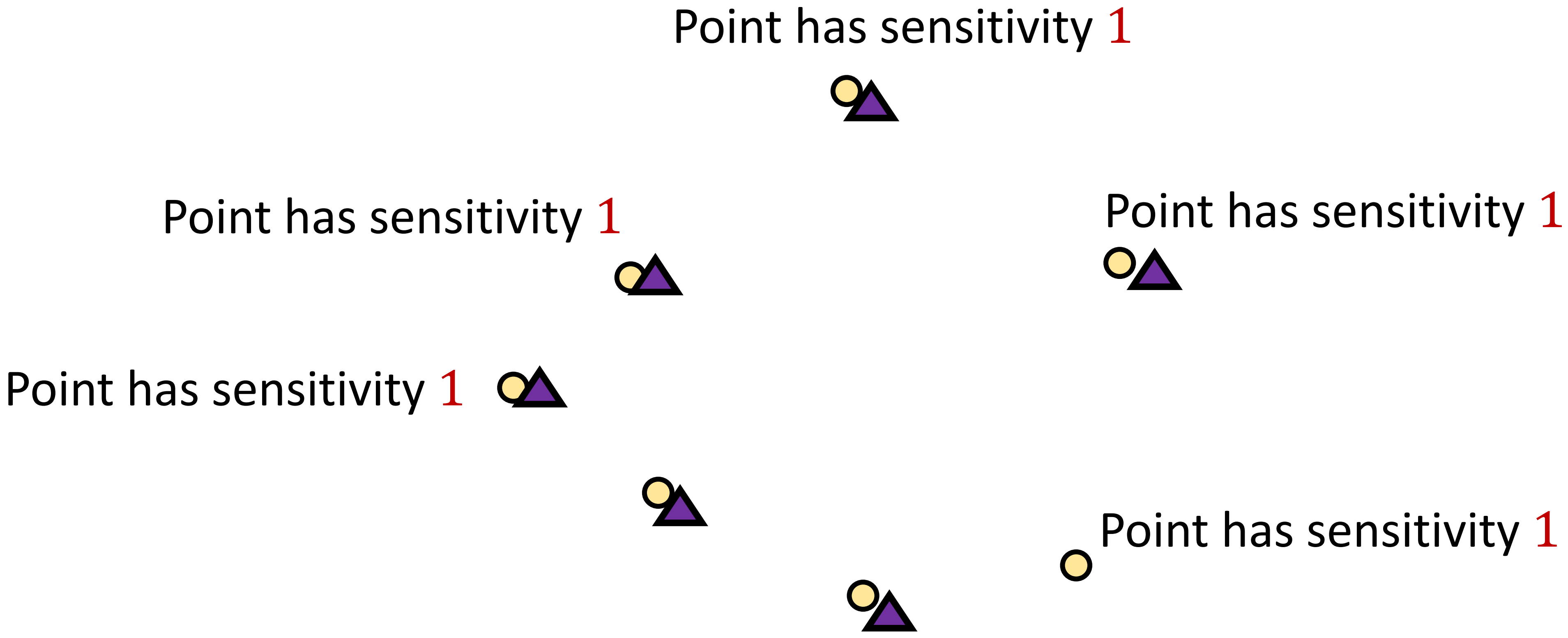
$$\varphi(x_t) = \max_{C:|C|\leq k} \frac{\mathrm{Cost}(x_t, C)}{\mathrm{Cost}(X_t, C)} = \max_{C:|C|\leq k} \frac{\mathrm{Cost}(x_t, C)}{\sum_{i=1}^{t} \mathrm{Cost}(x_i, C)}$$



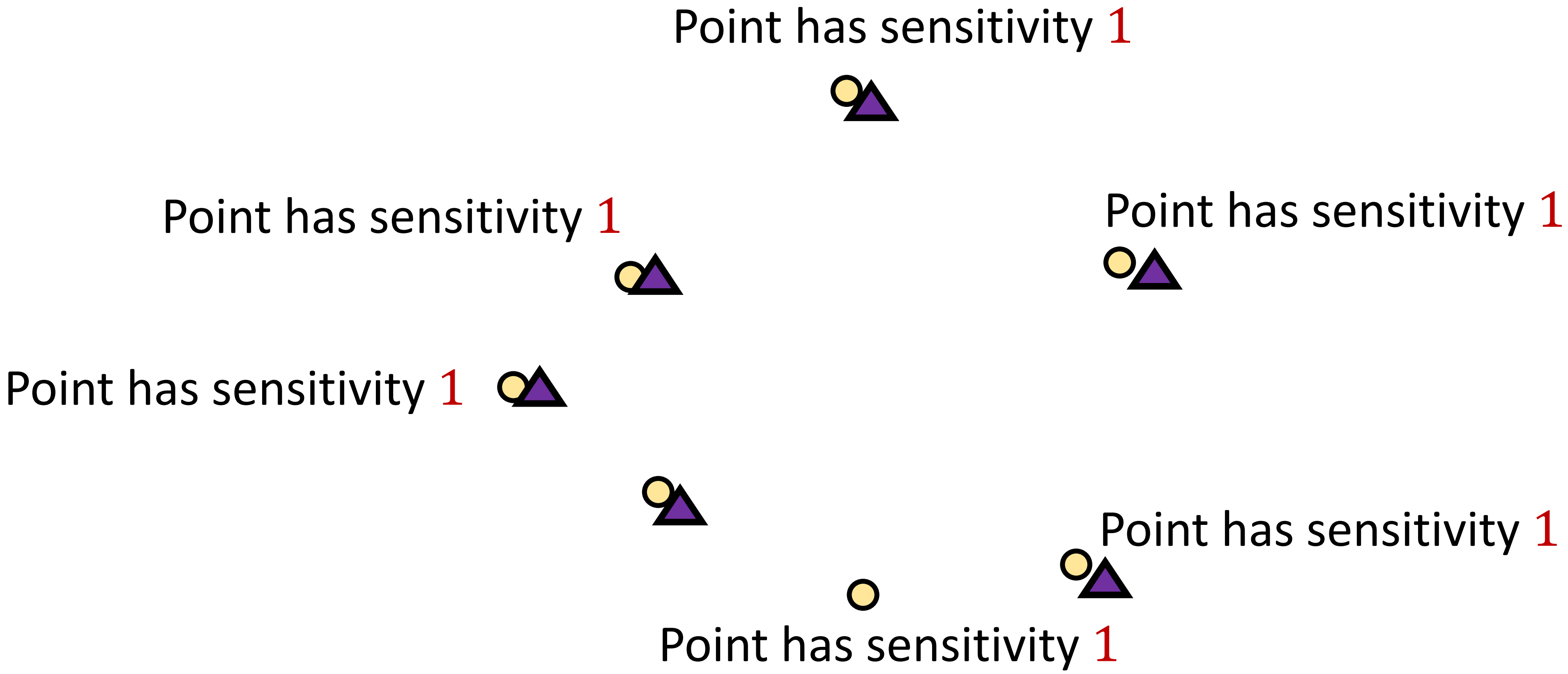
$$\varphi(x_t) = \max_{C:|C|\le k} \frac{\mathrm{Cost}(x_t, C)}{\mathrm{Cost}(X_t, C)} = \max_{C:|C|\le k} \frac{\mathrm{Cost}(x_t, C)}{\sum_{i=1}^{t} \mathrm{Cost}(x_i, C)}$$



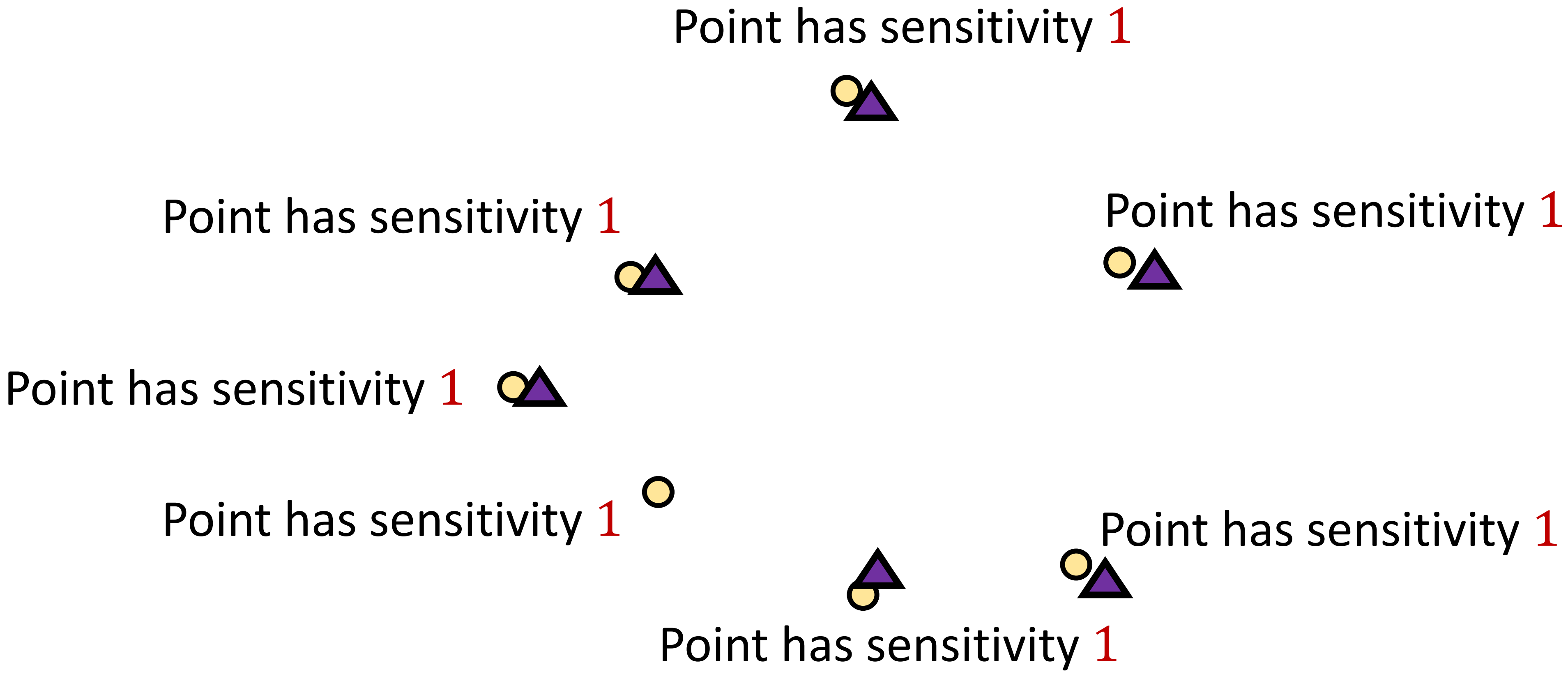
$$\varphi(x_t) = \max_{C:|C|\leq k} \frac{\text{Cost}(x_t, C)}{\text{Cost}(X_t, C)} = \max_{C:|C|\leq k} \frac{\text{Cost}(x_t, C)}{\sum_{i=1}^{t} \text{Cost}(x_i, C)}$$



$$\varphi(x_t) = \max_{C:|C|\le k} \frac{\mathrm{Cost}(x_t, C)}{\mathrm{Cost}(X_t, C)} = \max_{C:|C|\le k} \frac{\mathrm{Cost}(x_t, C)}{\sum_{i=1}^{t} \mathrm{Cost}(x_i, C)}$$



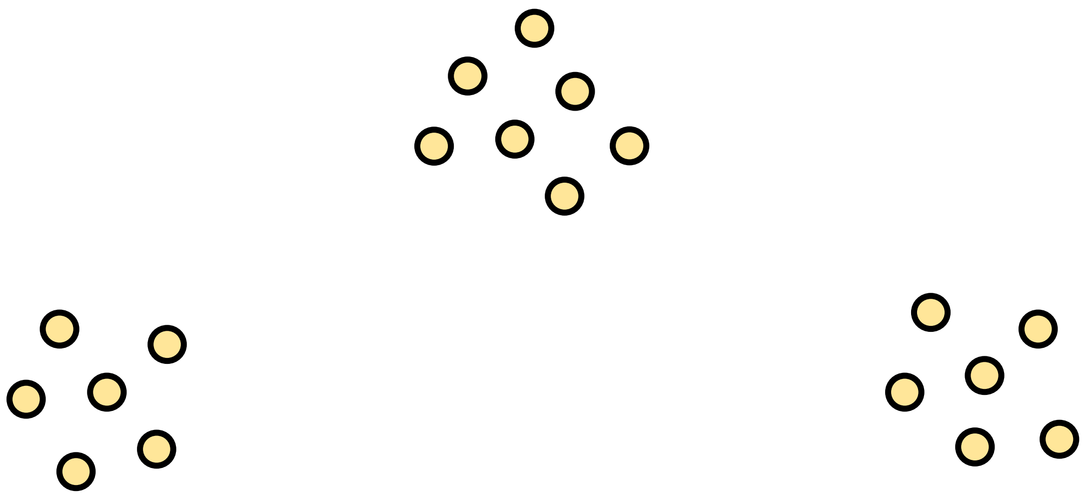
$$\varphi(x_t) = \max_{C:|C|\leq k} \frac{\mathrm{Cost}(x_t, C)}{\mathrm{Cost}(X_t, C)} = \max_{C:|C|\leq k} \frac{\mathrm{Cost}(x_t, C)}{\sum_{i=1}^{t} \mathrm{Cost}(x_i, C)}$$



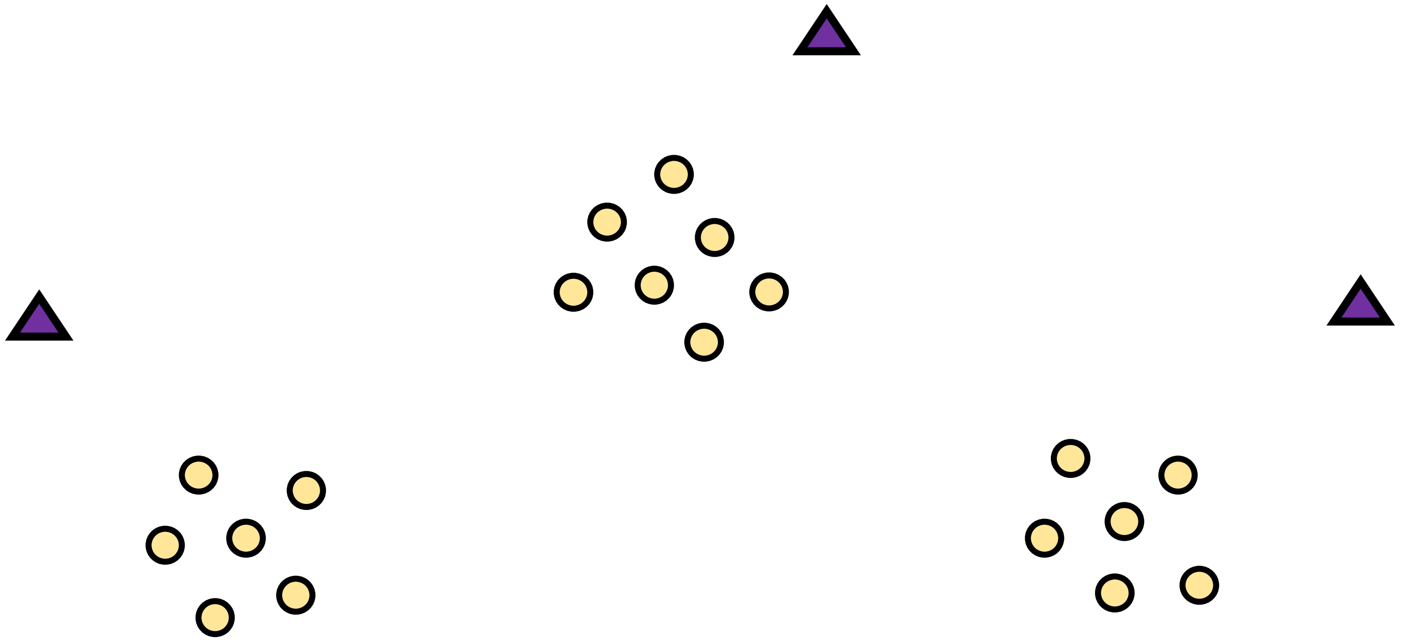
Sum of Online Sensitivity

- Sum of online sensitivities can be at least k
- How large can it be?

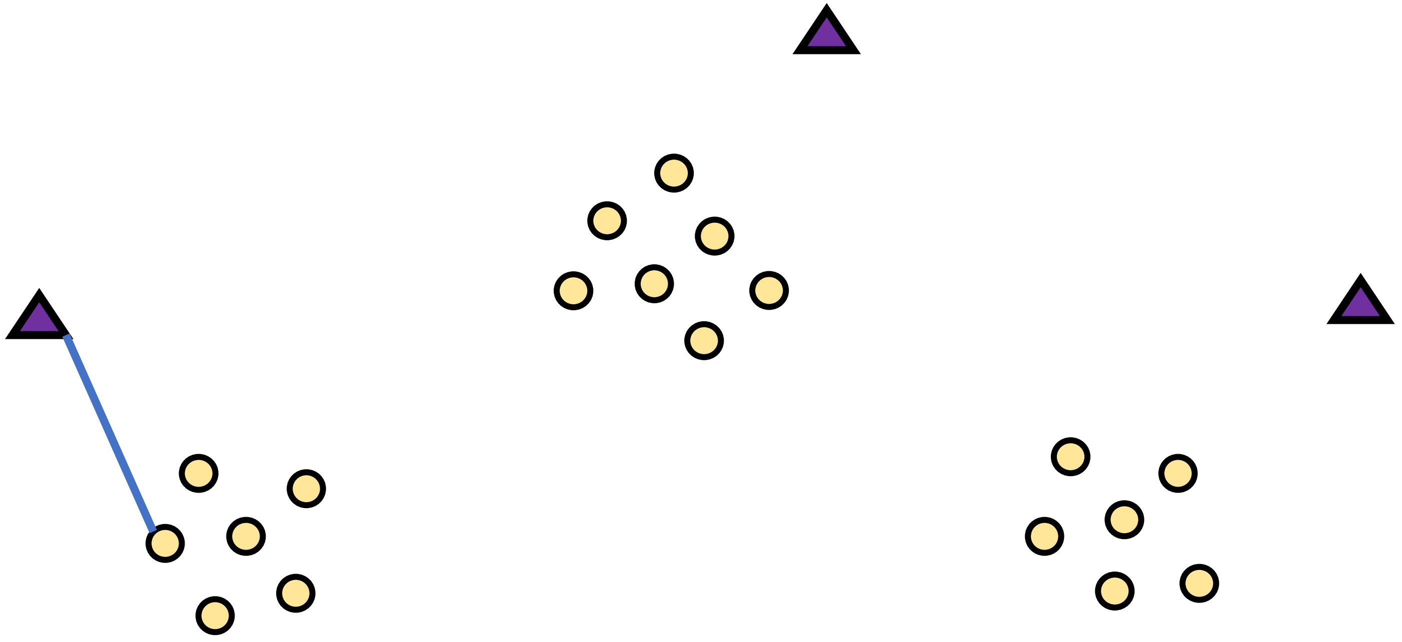
$$\varphi(x_t) = \max_{C:|C|\leq k} \frac{\mathrm{Cost}(x_t, C)}{\mathrm{Cost}(X_t, C)} = \max_{C:|C|\leq k} \frac{\mathrm{Cost}(x_t, C)}{\sum_{i=1}^{t} \mathrm{Cost}(x_i, C)}$$



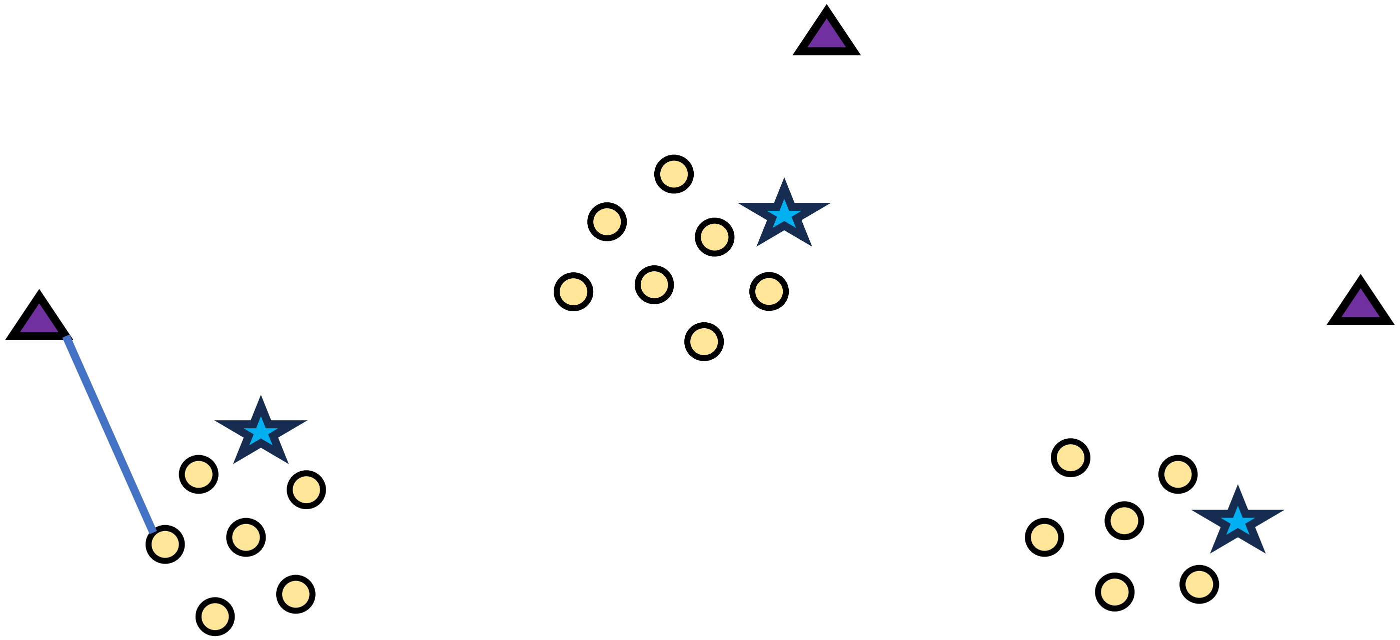
$$\varphi(x_t) = \max_{C:|C|\le k} \frac{\mathrm{Cost}(x_t, C)}{\mathrm{Cost}(X_t, C)} = \max_{C:|C|\le k} \frac{\mathrm{Cost}(x_t, C)}{\sum_{i=1}^{t} \mathrm{Cost}(x_i, C)}$$



$$\varphi(x_t) = \max_{C:|C|\leq k} \frac{\text{Cost}(x_t, C)}{\text{Cost}(X_t, C)} = \max_{C:|C|\leq k} \frac{\text{Cost}(x_t, C)}{\sum_{i=1}^{t} \text{Cost}(x_i, C)}$$

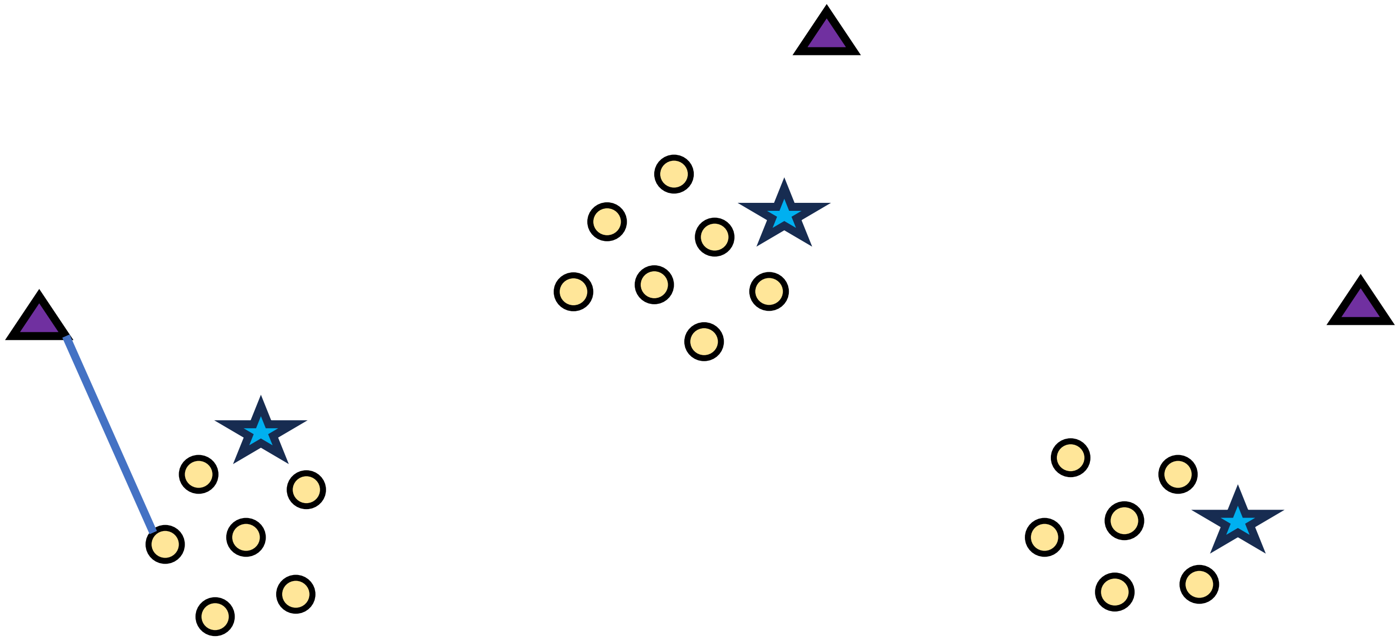


$$\varphi(x_t) = \max_{C:|C|\leq k} \frac{\text{Cost}(x_t, C)}{\text{Cost}(X_t, C)} = \max_{C:|C|\leq k} \frac{\text{Cost}(x_t, C)}{\sum_{i=1}^{t} \text{Cost}(x_i, C)}$$



$$\varphi(x_t) = \max_{C:|C|\le k} \frac{\mathrm{Cost}(x_t, C)}{\mathrm{Cost}(X_t, C)} = \max_{C:|C|\le k} \frac{\mathrm{Cost}(x_t, C)}{\sum_{i=1}^{t} \mathrm{Cost}(x_i, C)}$$

Partition the sum of the sensitivities by each cluster



Sum of Online Sensitivity

- Intuition: The sum of the sensitivities in each cluster induced by OPT is at most 1

- Since there are k clusters, the sum of the sensitivities is $O_z(k)$

- The sum of the online sensitivities is $O_z(k \log^2 nd\Delta)$

Insertion-Only Algorithm

1. Perform online sensitivity sampling to implicitly create new stream X'
2. In parallel, run merge-and-reduce on X'

Insertion-Only Summary

- New stream X' has length $\frac{k^2 d}{\varepsilon^2} \cdot \text{polylog}(n\Delta)$
- Can run merge-and-reduce framework on X'
- Recall total space used by merge-and-reduce was $f\left(k, \frac{\log n}{\varepsilon}\right) \cdot O(\log n)$ points, but n was the length of the stream
- Total space is $f\left(k, \frac{\log |S'|}{\varepsilon}\right) \cdot O(\log |X'|)$ points with $f\left(k, \frac{1}{\varepsilon}\right) = \tilde{O}\left(\frac{k}{\varepsilon^2}\right) \cdot \min\left(k, \frac{1}{\varepsilon^z}\right)$, i.e., $o(\log n)$

Format

- Part 1: Background
- Part 2: Insertion-Only Streams
- Part 3: k-Median on Dynamic Streams
- Part 4: (k, z)-Clustering on Dynamic Streams

Questions?

Insertion-Deletion Streams

- Use first pass to estimate sensitivity of each point n in the stream

- Use second pass to perform sensitivity sampling

Sensitivity Estimation

- Sensitivity of a point x is $s(x) := \max_{C:|C|\leq k} \frac{\mathrm{Cost}(x, C)}{\mathrm{Cost}(X, C)}$

- Suppose S is the optimal (capacitated) set of k centers, so that $\mathrm{Cost}(X, S) \leq \mathrm{Cost}(X, C)$ for all sets C of k centers

- Claim: $\frac{4 \cdot 2^z \cdot \mathrm{Cost}(x, C)}{\mathrm{Cost}(C, S) + \mathrm{Cost}(X, S)}$ is a good approximation of $s(x)$

Sensitivity Estimation

$$\frac{\text{Cost}(x, C)}{\text{Cost}(X, C)} = \frac{4 \cdot \text{Cost}(x, C)}{4 \cdot \text{Cost}(X, C)}$$

(Optimality of S) $$\leq \frac{4 \cdot \text{Cost}(x, C)}{2 \cdot \text{Cost}(X, C) + 2 \cdot \text{Cost}(X, S)}$$

$$\leq \frac{4 \cdot \text{Cost}(x, C)}{\text{Cost}(X, C) + 2 \cdot \text{Cost}(X, S)}$$

(Triangle Inequality) $$\leq \frac{4 \cdot 2^z \cdot \text{Cost}(x, C)}{\text{Cost}(C, S) + \text{Cost}(X, S)}$$

Sensitivity Estimation

$$\frac{4 \cdot 2^{z} \cdot \mathrm{Cost}(x, C)}{\mathrm{Cost}(C, S) + \mathrm{Cost}(X, S)} \leq \frac{2^{O(z)} \cdot \mathrm{Cost}(x, C)}{\mathrm{Cost}(X, S) + \mathrm{Cost}(X, C)}$$

(Triangle Inequality)

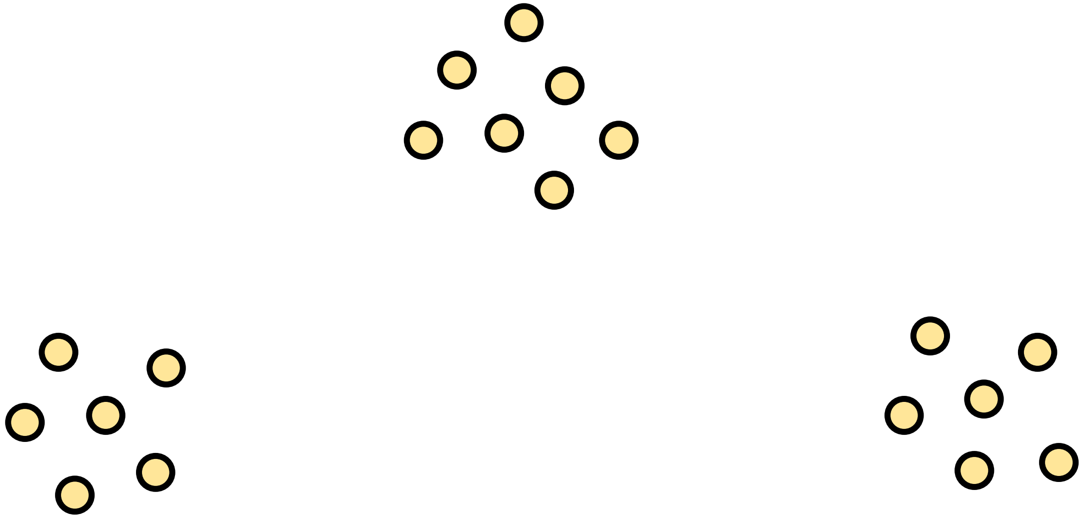
$$\leq \frac{2^{O(z)} \cdot \mathrm{Cost}(x, C)}{\mathrm{Cost}(X, C)}$$

Sensitivity Estimation

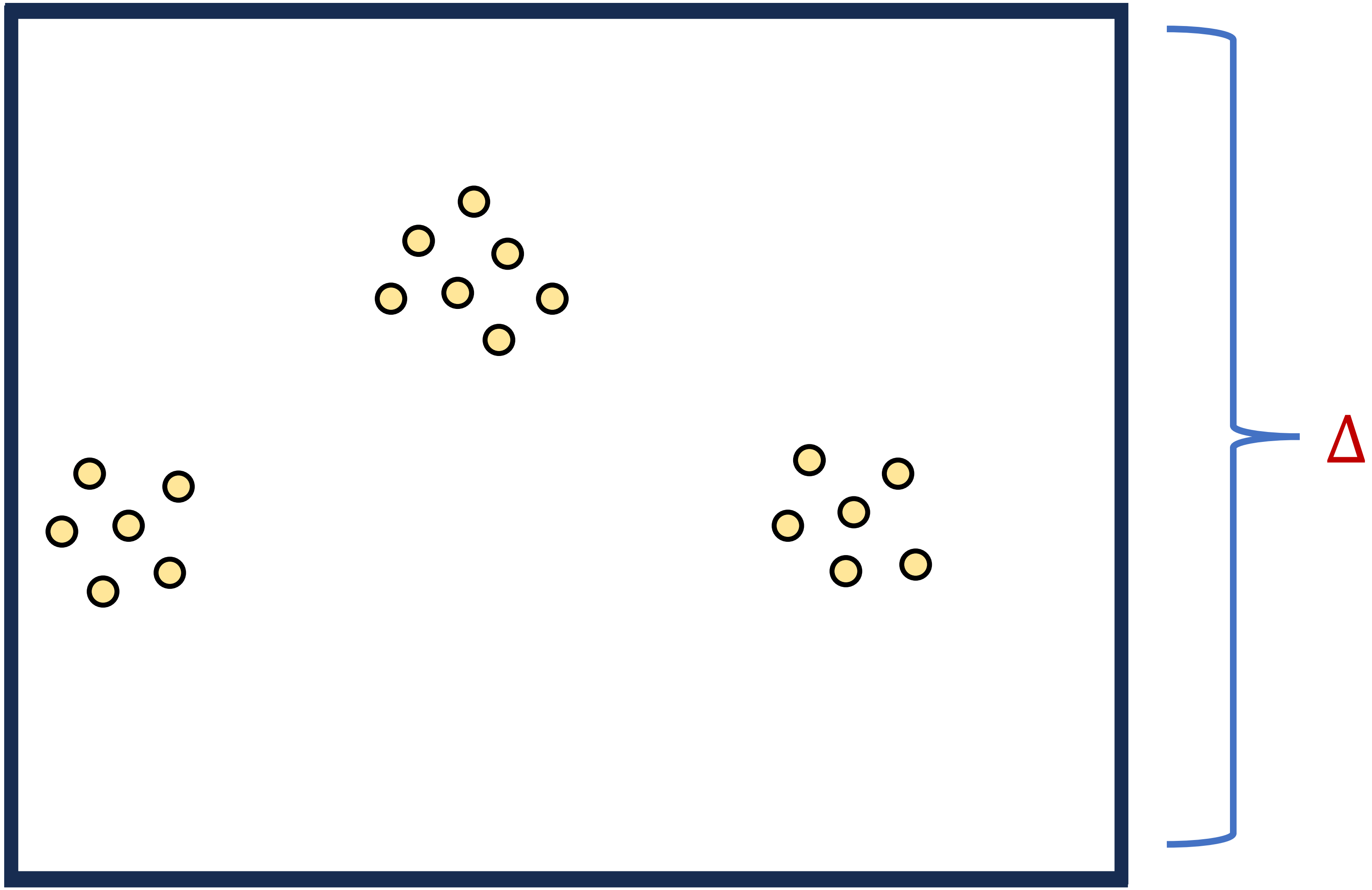
- Takeaway: Can use a “good” (capacitated) set S of k centers along with an approximation of its cost to estimate sensitivities $s(x)$ of all points

- How to find such an estimate?
- Cannot use online sensitivity sampling or merge-and-reduce anymore

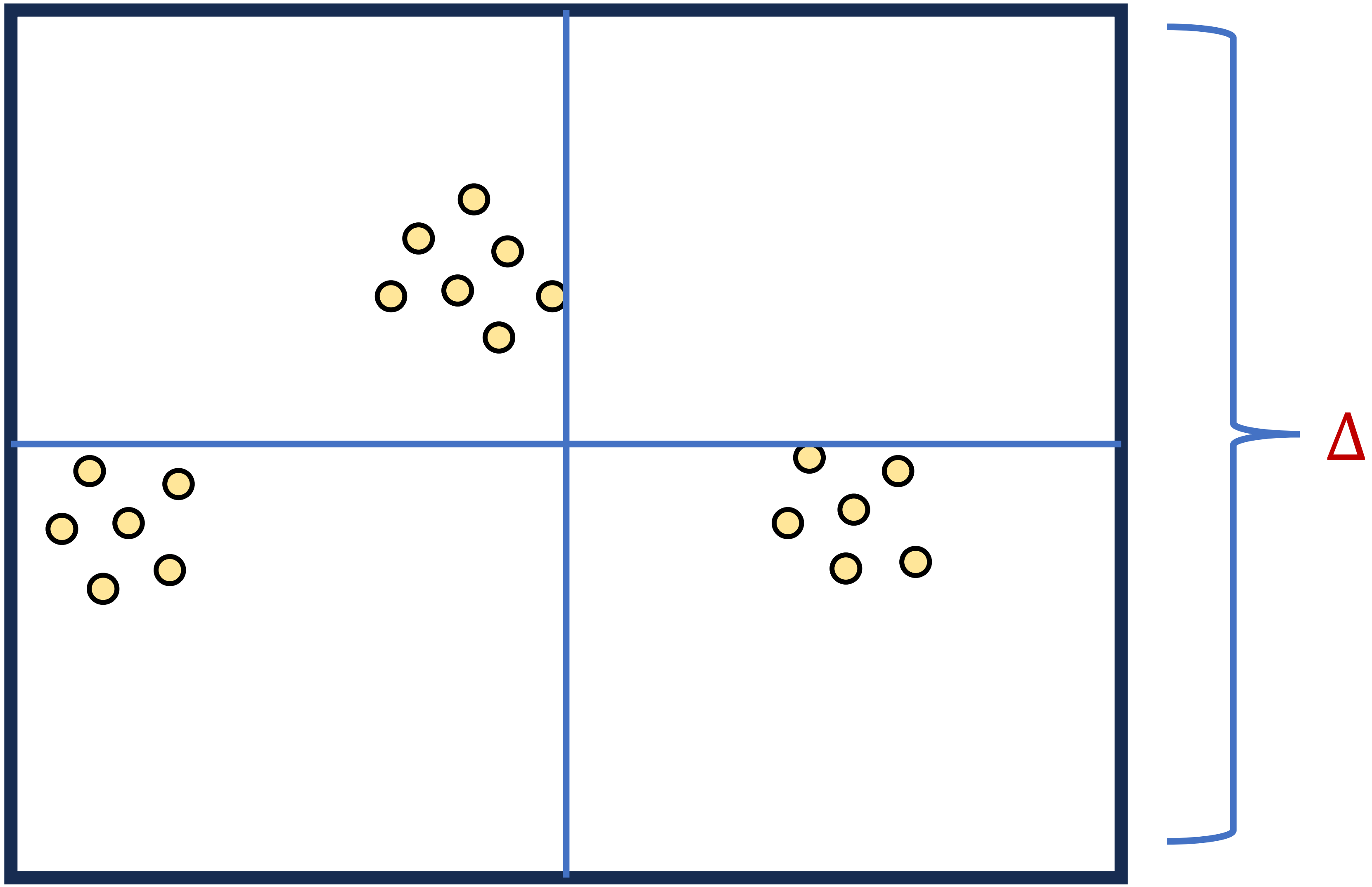
Quadtree Embedding



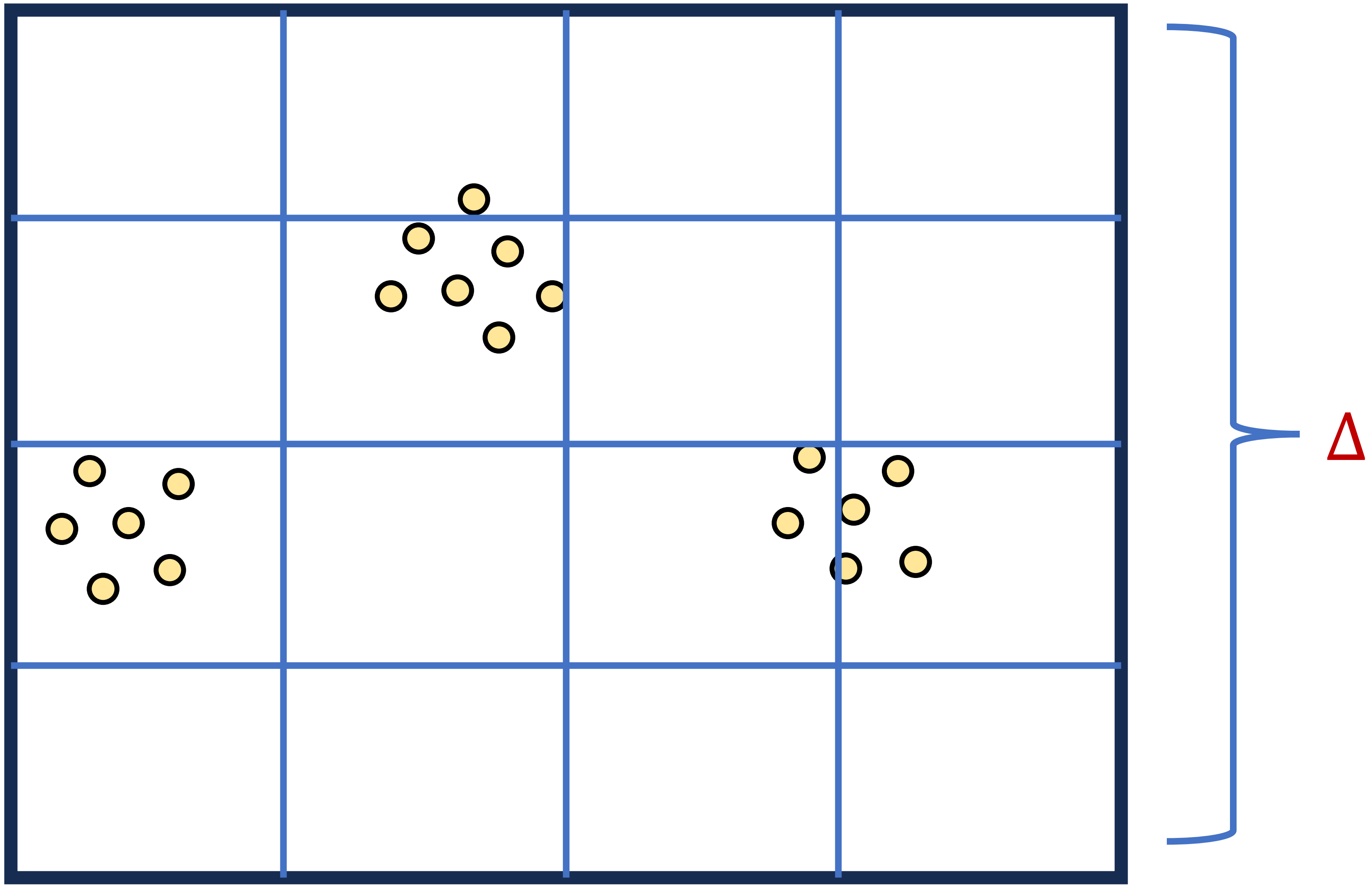
Quadtree Embedding



Quadtree Embedding

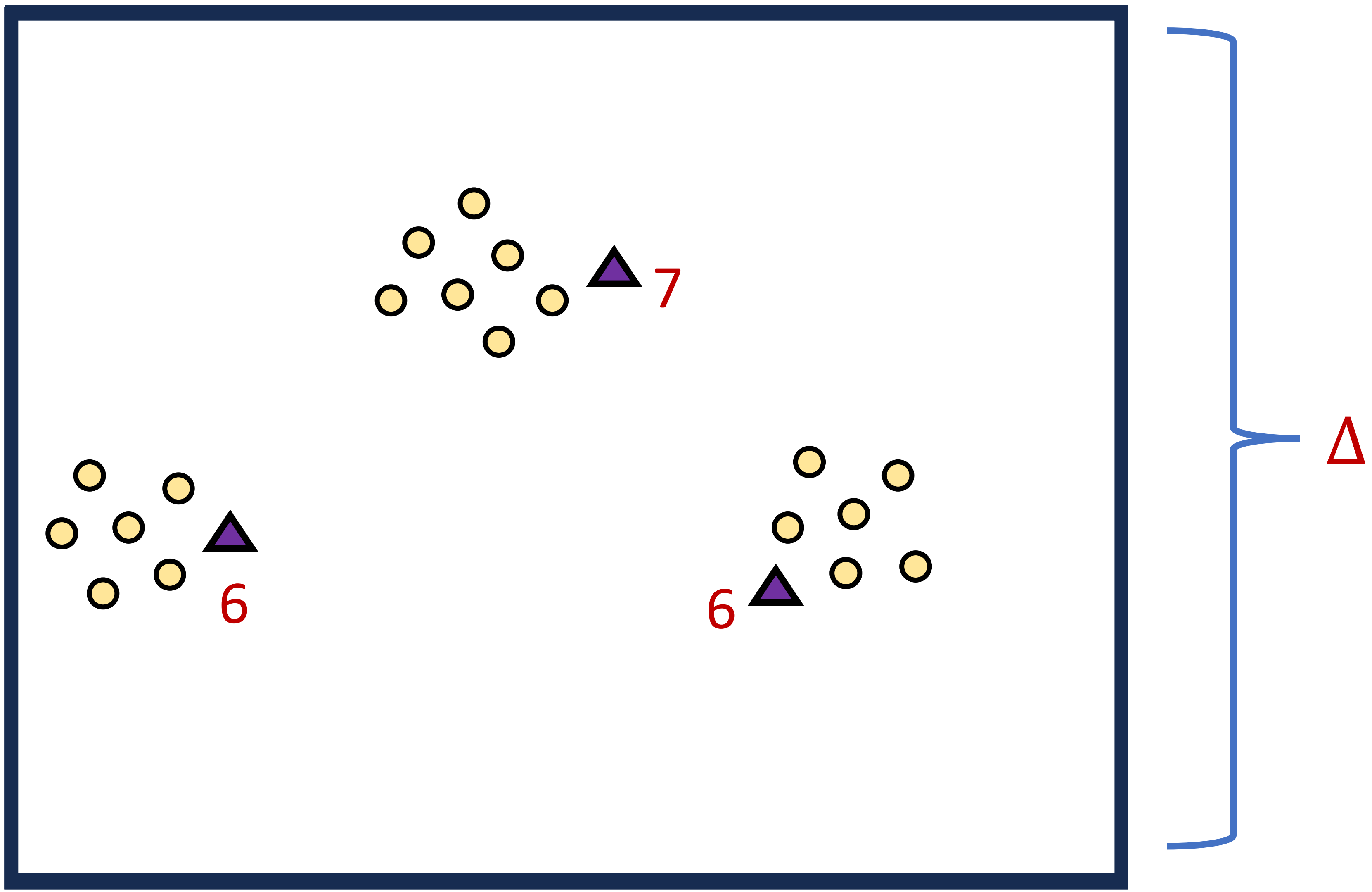


Quadtree Embedding



Quadtree Embedding

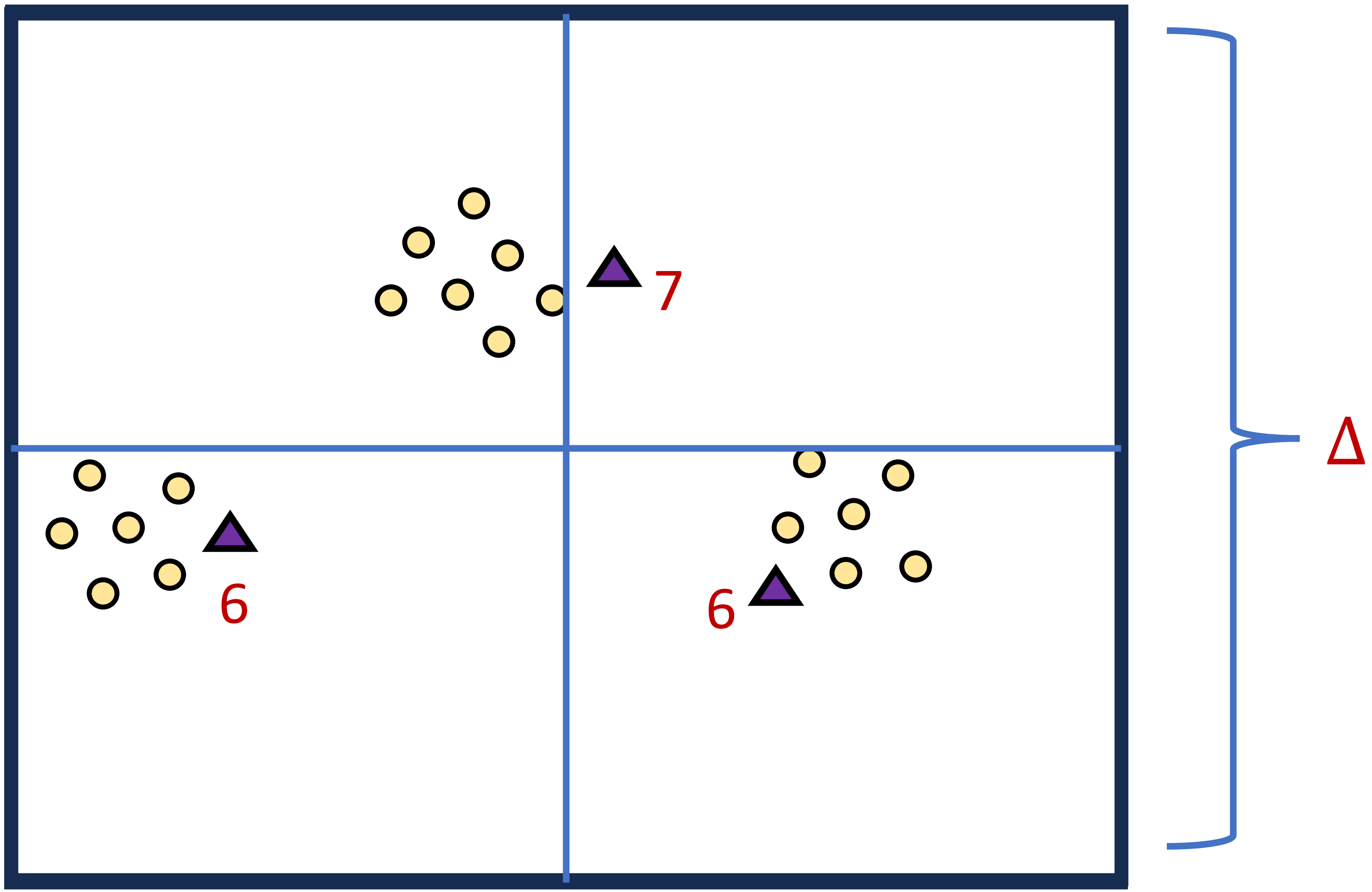
Total cost: 0
Level cost: 0



Quadtree Embedding

Total cost: $\frac{\Delta}{2} \cdot 7$

Level cost: $\frac{\Delta}{2} \cdot 7$

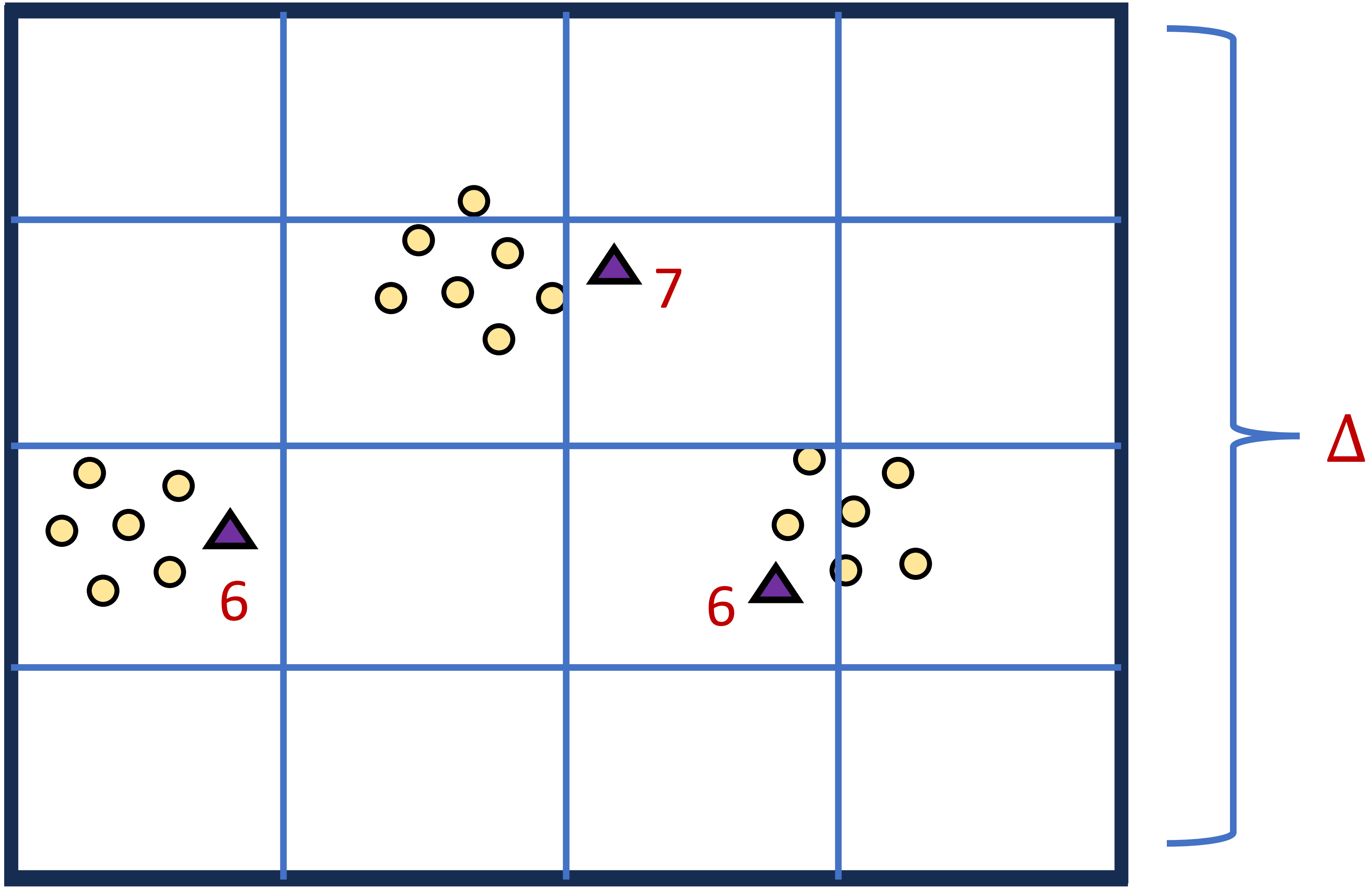


Quadtree Embedding

Total cost:

$\left(\frac{7}{2}+\frac{11}{4}\right)\Delta$

Level cost: $\frac{\Delta}{4}\cdot 11$



Quadtree Embedding

- Earth mover distance: $\mathrm{EMD}(C, X)$ denotes the k-median clustering cost $\mathrm{Cost}(C, X)$ for X using a (capacitated) set C of centers

- Quadtree embedding: For a (weighted) set C of centers, the quadtree embedding outputs Z such that

$$\mathrm{EMD}(C, X) \leq O\left(\sqrt{d}\right) \cdot Z \leq \cdot\, O(d^{1.5})(\log k + \log\log \Delta)\, \mathrm{EMD}(C, X)$$

Quadtree Embedding

- Quadtree embedding produces a vector of dimension $\Delta^{O(d)}$
- The computation of Z is the sum of the level costs, which is the L_1 norm of the frequency vector
- There exists a one-pass streaming algorithm that outputs a constant-factor approximation to the L_1 norm of a frequency vector in $\mathbb{R}^n$ and uses $O(\log n)$ bits of space [Indyk06]

L_1 Norm Approximation

- There exists a one-pass streaming algorithm that outputs a constant-factor approximation to the L_1 norm of an underlying vector x in $\mathbb{R}^n$ and uses $O(\log n)$ bits of space [Indyk06]

- Generate vector $v_1, \ldots v_\alpha \in \mathbb{R}^n$ of Cauchy random variables (ratio of two normal random variables) for $\alpha = O(1)$
- Output $\mathrm{median}_{i\in[\alpha]}\{|\langle v_1, x\rangle|, \ldots, |\langle v_\alpha, x\rangle|\}$

EMD Sketch

- EMD sketch: There exists a one-pass streaming algorithm that uses $O(d \log \Delta)$ bits of space and outputs Z such that

$$\mathrm{EMD}(C, X) \leq O\left(\sqrt{d}\right) \cdot Z \leq \cdot\, O(d^{1.5})(\log k + \log\log \Delta)\, \mathrm{EMD}(C, X)$$

EMD Sketch

- [BackursIndykRazenshteynWoodruff16] To estimate $\min_{C,|C|\leq k} \text{Cost}(C, X)$, it suffices to union bound over a net of size $\exp\left(kd(\log\log\Delta)\right)$

- EMD sketch: There exists a one-pass streaming algorithm that uses $O\left(kd^2 \log\Delta\, (\log\log\Delta)\right)$ bits of space and outputs Z (as well as the capacitated set of centers) such that

$$\text{OPT} \leq O\left(\sqrt{d}\right) \cdot Z \leq \cdot\, O(d^{1.5})(\log k + \log\log\Delta)\,\text{OPT}$$

EMD Sketch Summary

- EMD sketch: There exists a one-pass streaming algorithm that uses $O\left(kd^2 \log \Delta \left(\log \log \Delta\right)\right)$ bits of space and outputs Z (as well as the capacitated set of centers) such that

$$\mathrm{OPT} \leq O\left(\sqrt{d}\right) \cdot Z \leq \cdot\, O(d^{1.5})(\log k + \log \log \Delta)\ \mathrm{OPT}$$

- Recall: Can use a "good" (capacitated) set S of k centers along with an approximation of its cost to estimate sensitivities $s(x)$ of all points

First Pass to Second Pass

- We can set up the EMD sketch in the first pass of the stream

- At the end of the first pass of the stream, we have a data structure that can estimate the sensitivity $s(x)$ for any query $x \in [\Delta]^d$

- In the second pass of the stream, we would like to perform sensitivity sampling

Sensitivity Sampling

- DO NOT: Sample each point x in the stream with probability proportional to $s(x)$
  - Does not work for insertion-deletion streams
- DO: Sample each point x in the universe $[\Delta]^d$ into a substream U' with probability proportional to $s(x)$
  - U' can have a large number of points
  - U' can have a small number of points at the end of the stream

Sensitivity Sampling

- Sample each point x in the universe $[\Delta]^d$ into a substream U' with probability proportional to $s(x)$

- U' will have $\text{poly}\left(k, d, \frac{1}{\varepsilon^2}\right)$ points at the end of the stream

- Use sparse recovery on U'

Sparse Recovery

- Given a stream U' that induces a frequency vector of length n with s nonzero entries, there exists an algorithm that uses $O(s \log n)$ bits of space and recovers the nonzero coordinates and their frequencies

- Since elements are sampled into U' by their sensitivities, recovering U' by sparse recovery corresponds to sensitivity sampling!

k-Median Framework

- First pass: set up the EMD sketch
- Second pass:
  - Sample elements into a substream U' with probability proportional to their sensitivities
  - Run sparse recovery on U'

Format

- Part 1: Background
- Part 2: Insertion-Only Streams
- Part 3: k-Median on Dynamic Streams
- Part 4: (k, z)-Clustering on Dynamic Streams

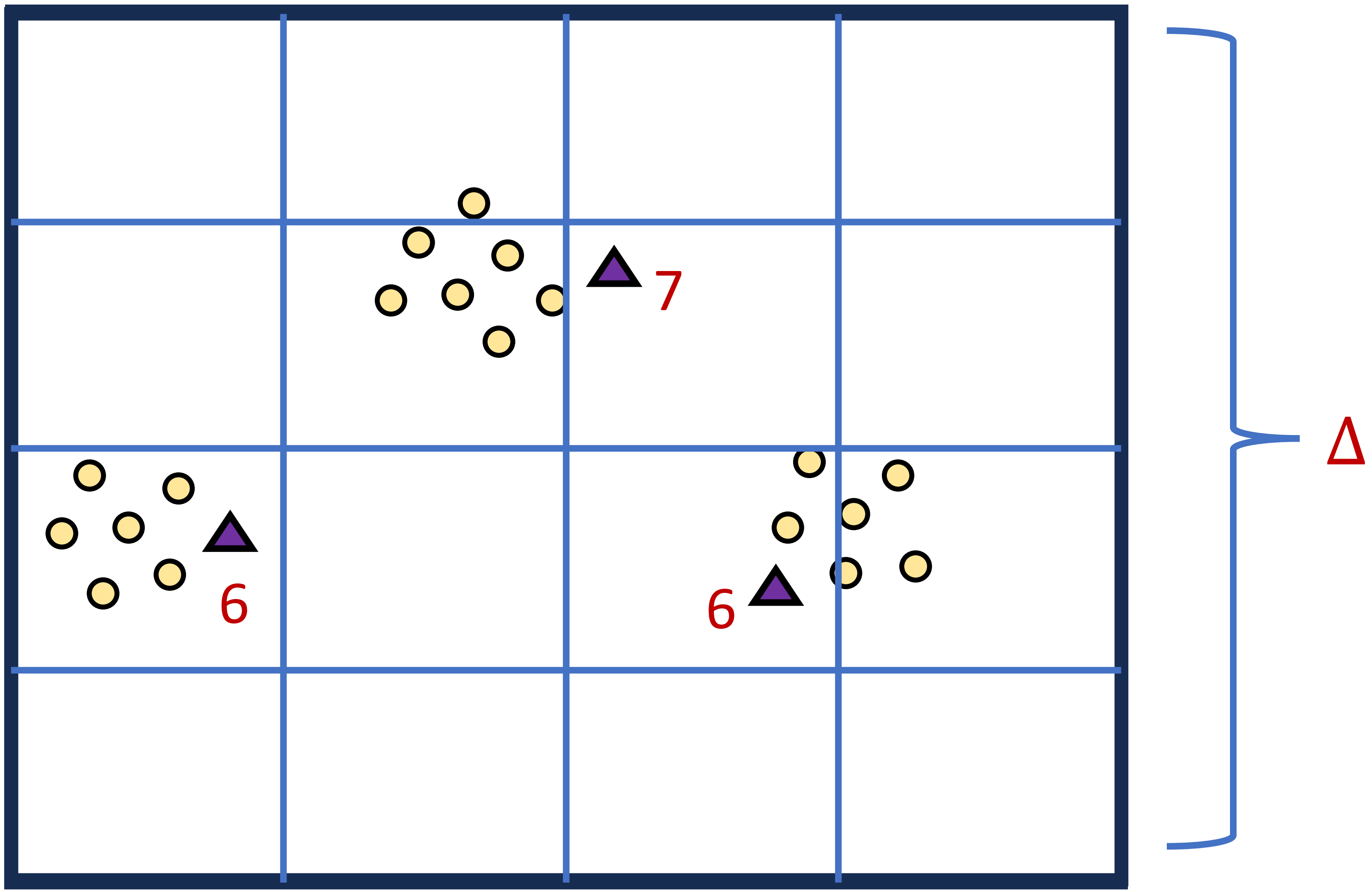
Questions?

k-Median Framework

- First pass: set up the EMD sketch
- Second pass:
  - Sample elements into a substream U' with probability proportional to their sensitivities
  - Run sparse recovery on U'

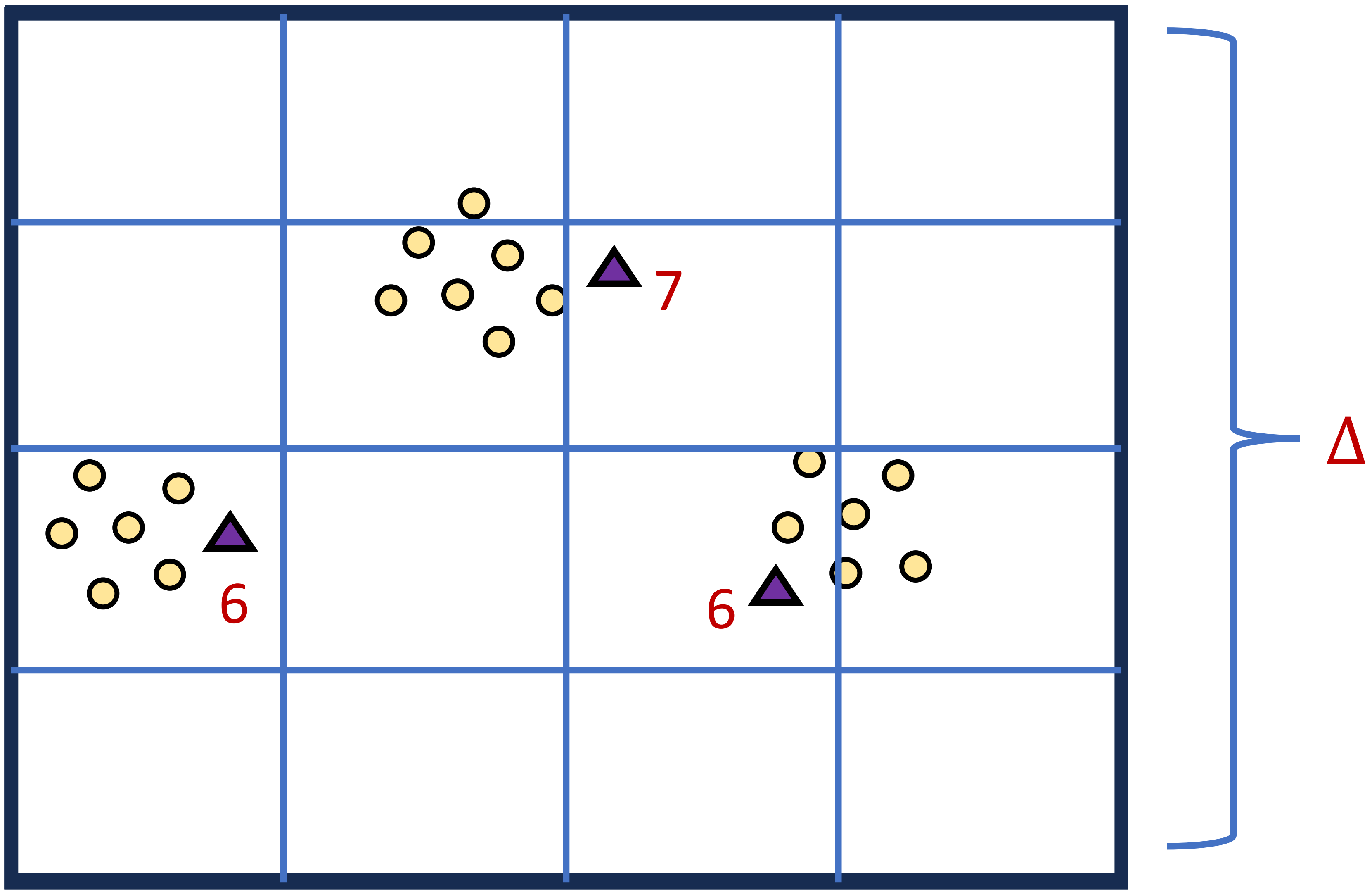
Quadtree Embedding

Level cost: $\frac{\Delta}{4} \cdot 11$



Quadtree Embedding

Level cost: $\frac{\Delta^2}{16} \cdot 11$



Quadtree Embedding

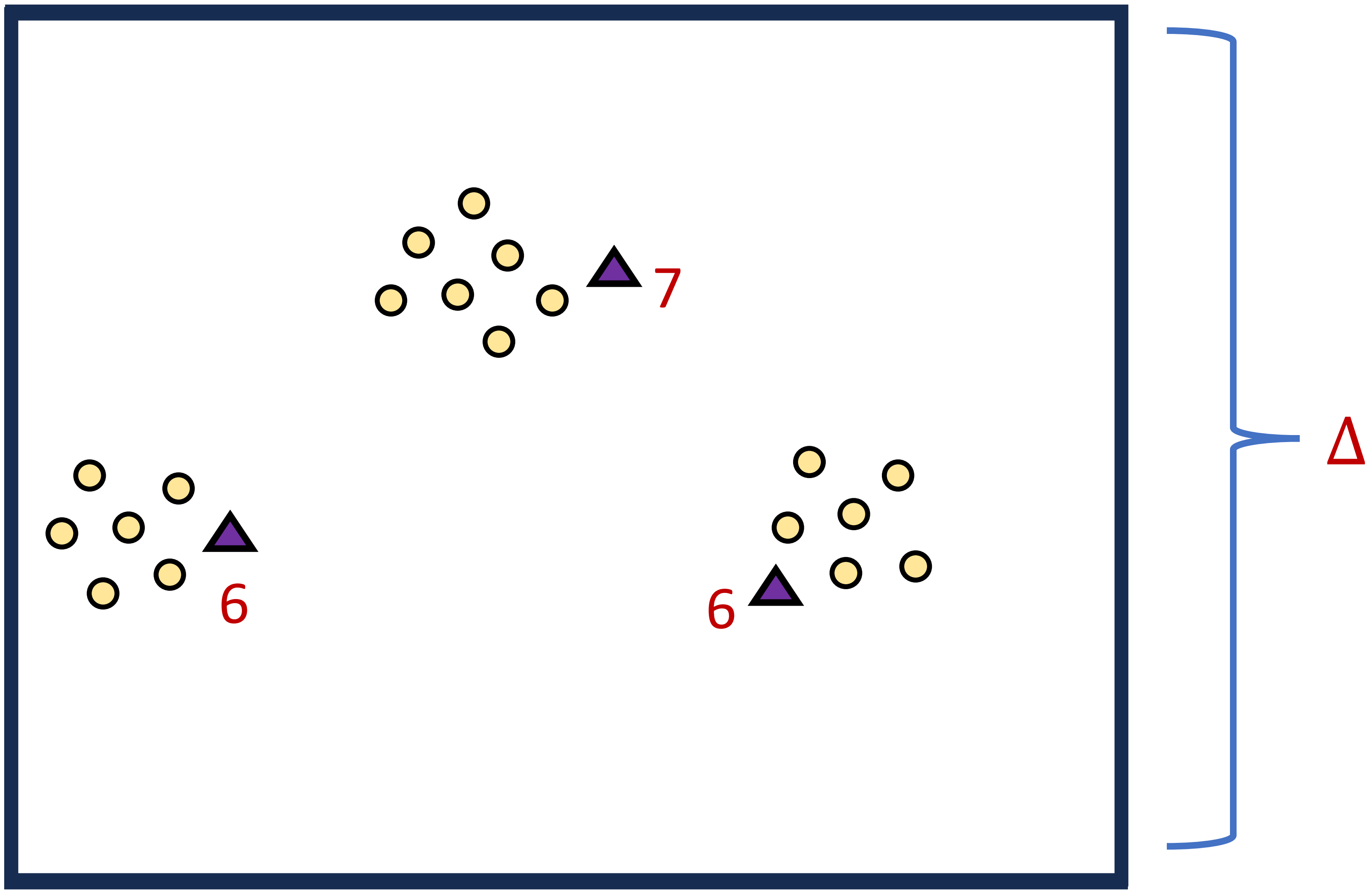
- If x and c have distance $\alpha\Delta$, the probability it will be split by a grid of length $\frac{\Delta}{2^i}$ is roughly $\frac{2^i}{\alpha}$
- Expected cost for k-median is $\alpha\Delta$
- Expected cost of k-means is $\frac{\Delta^2}{2^i\alpha}$, i.e., distortion $2^i\alpha^3$

- Recall: worse EMD sketch guarantee corresponds to larger oversampling necessary for sensitivity sampling

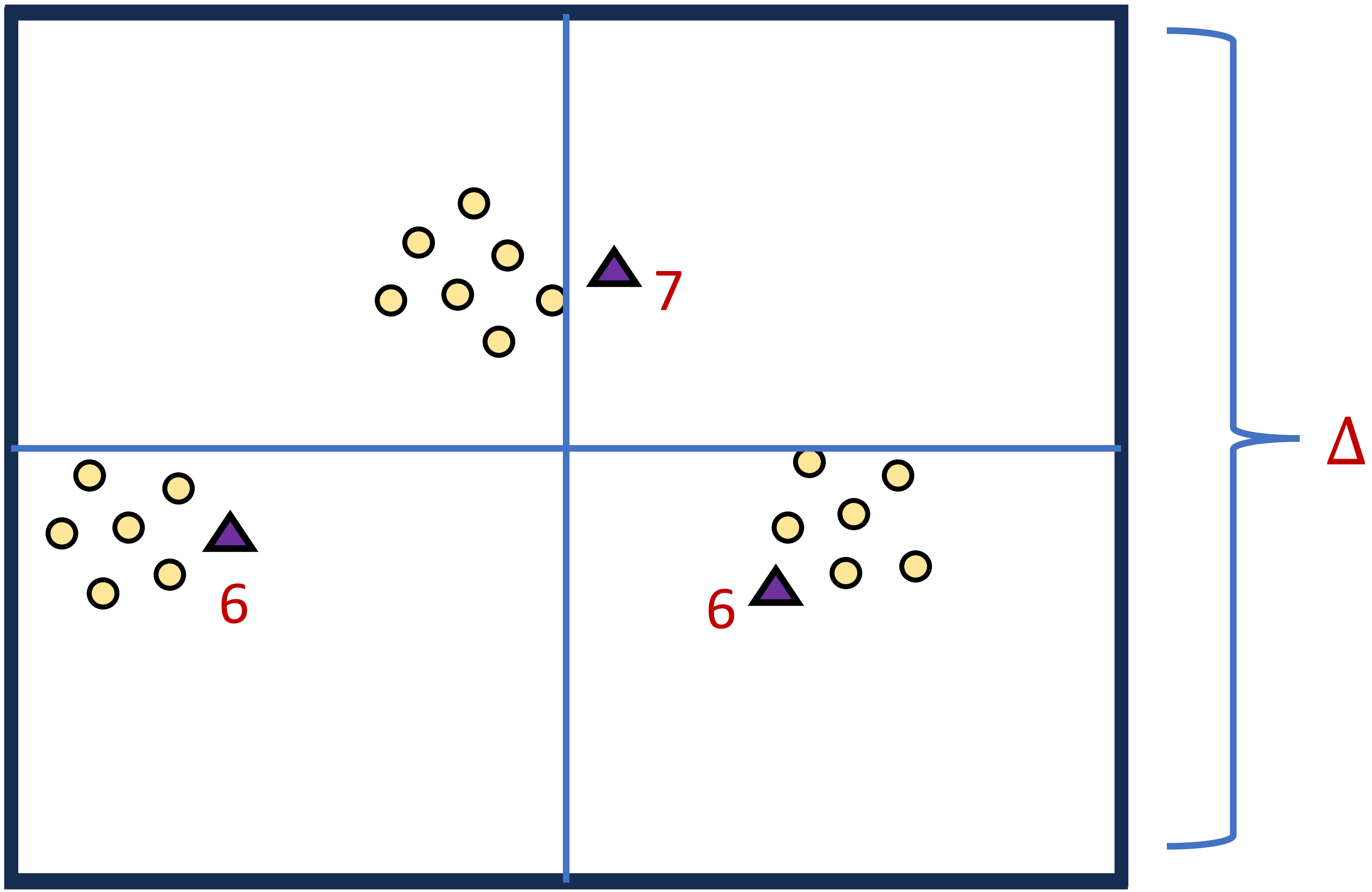
Quadtree Embedding

- Intuition: Bad distortion results when pairs of points are "too close" to the boundary of the hypergrid
- Goal: Prevent this case from happening
- Fix: When a query center is too close to the boundary of the hypergrid, create another center on the opposite cell!

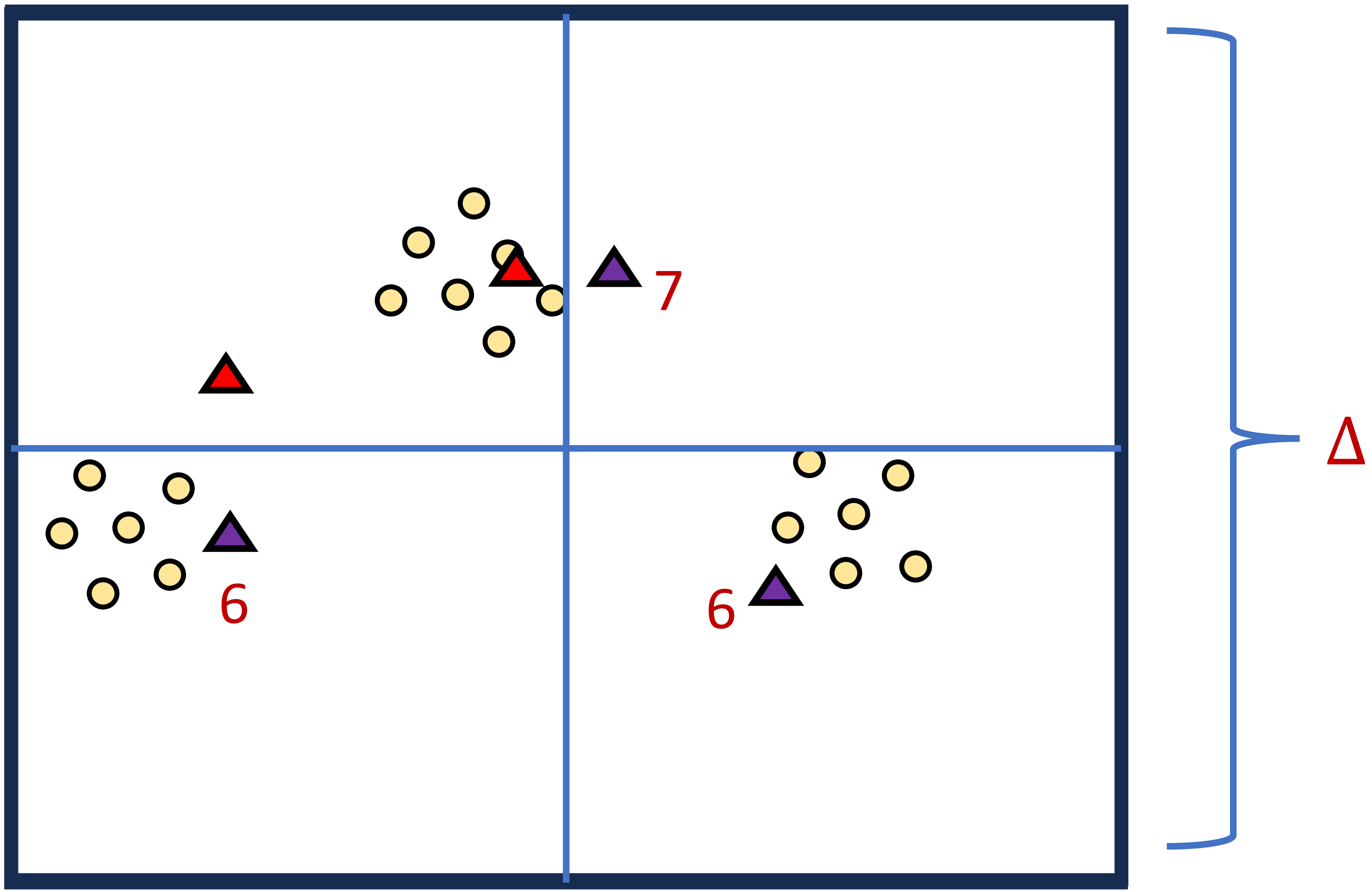
Quadtree Embedding



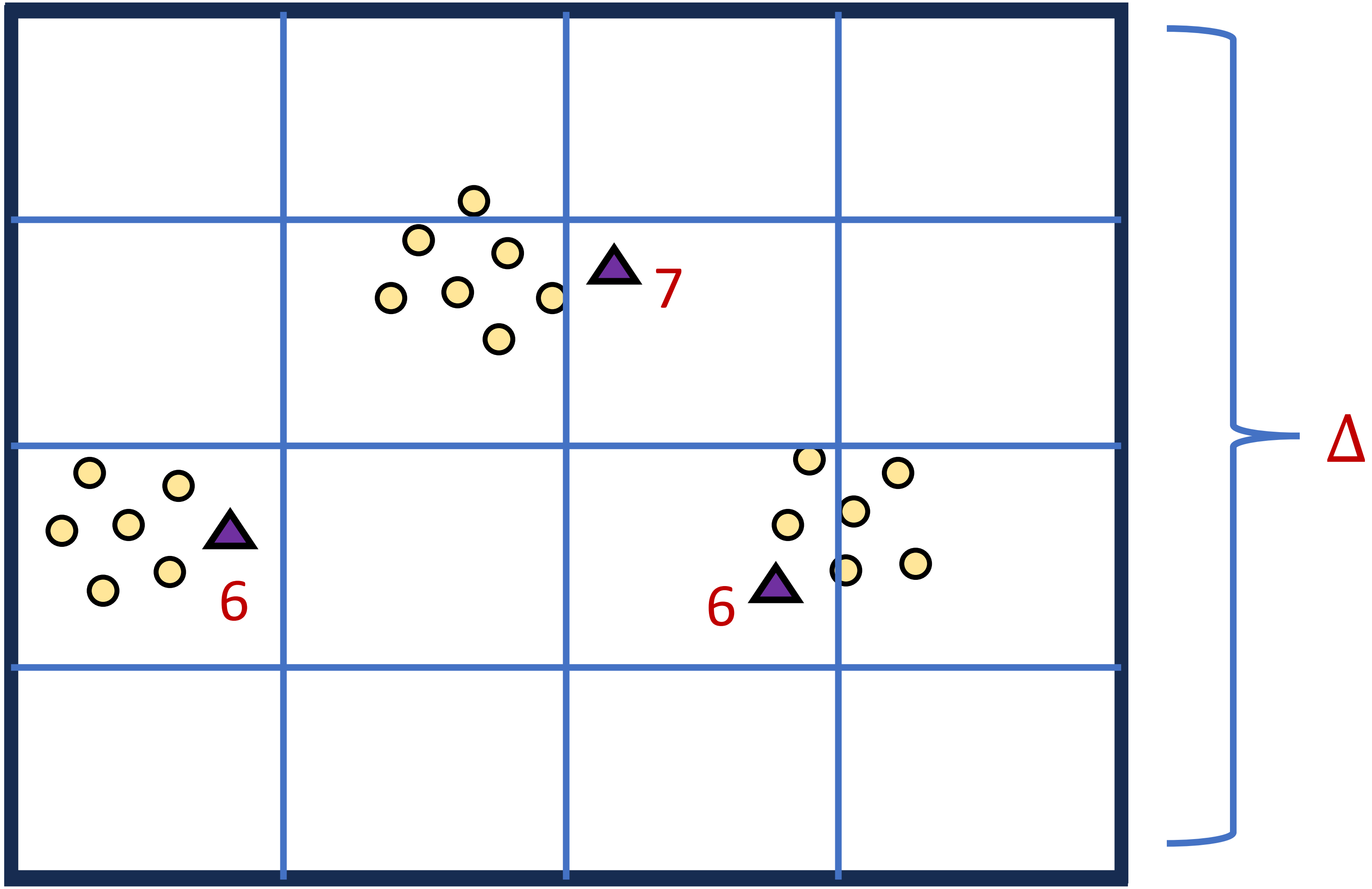
Quadtree Embedding



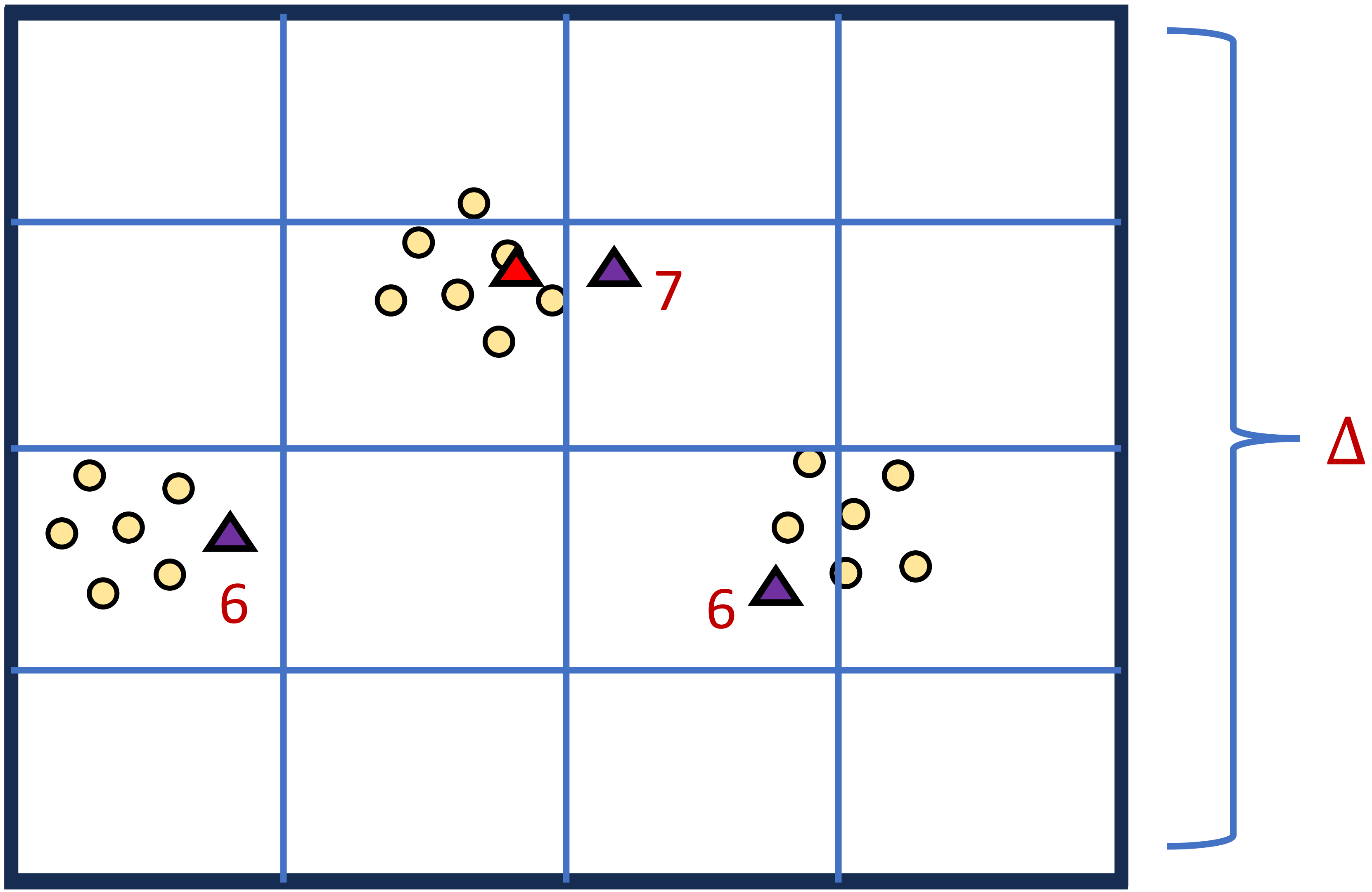
Quadtree Embedding



Quadtree Embedding



Quadtree Embedding



Quadtree Embedding

- Make a new center when distance from query center and hypergrid with length 2^i is at most $\frac{2^i}{d \log \Delta}$

- In expectation (over d dimensions, $\log \Delta$ levels of the hypergrid, and k query centers), $O(k)$ new centers are created

Wasserstein Sketch

- Wasserstein-z distance: $\mathrm{WASSD}(C, X)$ denotes the (k, z)-clustering cost $\mathrm{Cost}(C, X)$ for X a (capacitated) set C of centers

- Wasserstein sketch: There exists a one-pass streaming algorithm that uses $O(d \log \Delta)$ bits of space and outputs Z such that

$$Z \leq \cdot\, O(d^{1+0.5z} \log^{z-1} \Delta) \cdot \mathrm{WASSD}(C, X)$$

Applying k-Median Framework to k-Means

- First pass: set up the Wasserstein sketch

- Second pass:
  - Sample elements into a substream U' with probability proportional to their sensitivities
  - Run sparse recovery on U'

Applying k-Median Framework to k-Means

- Problem: Because the distortion of the Wasserstein embedding is $O(d^{1+0.5z} \log^{z-1} \Delta)$, we need to sample $O(d^2 \log \Delta)$ points for k-means

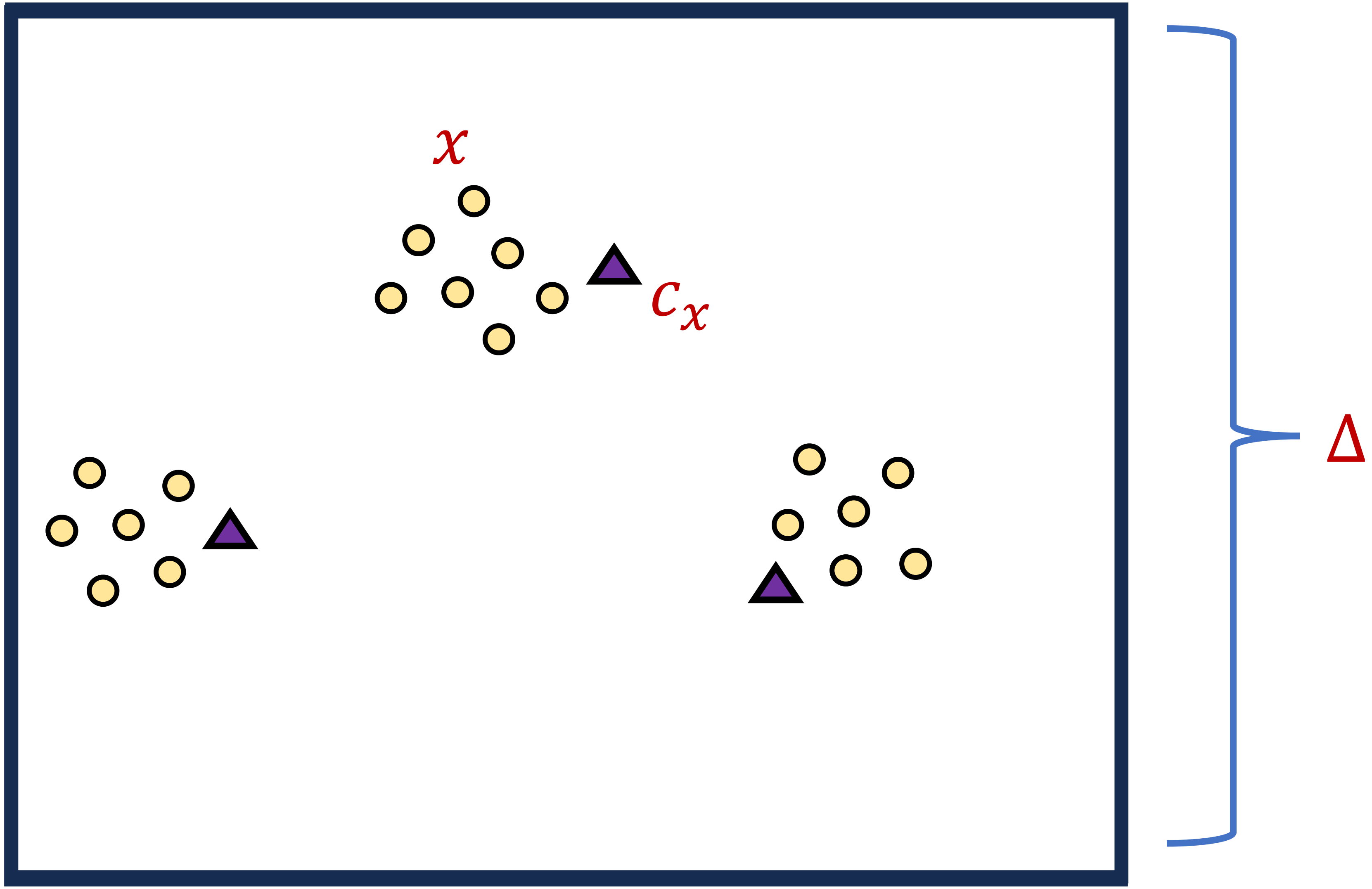
- For k-median, we stored all the points, using $O(d \log \Delta)$ bits of space per point

- Cannot afford to store all points explicitly here

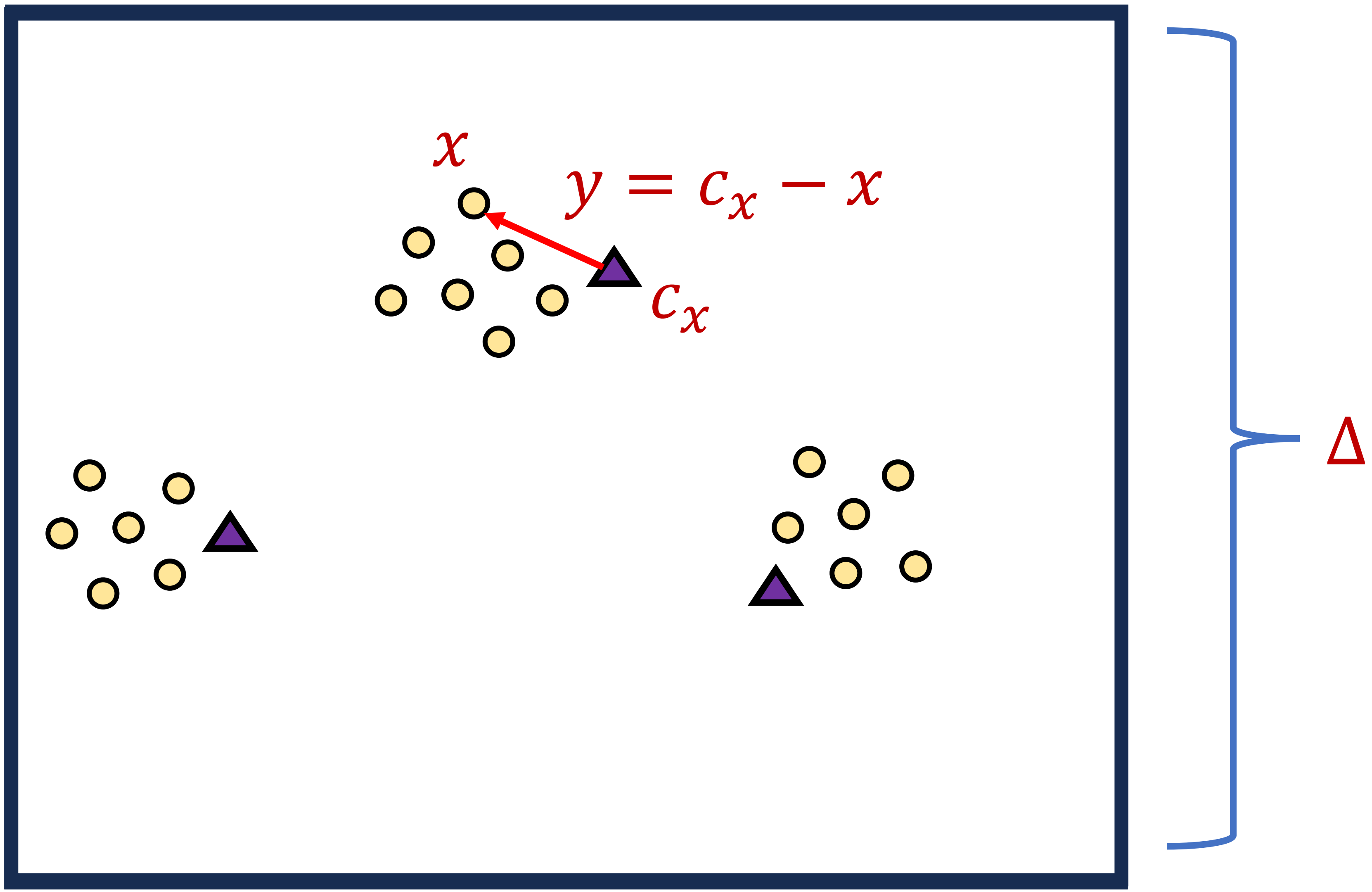
Applying k-Median Framework to k-Means

- Cannot afford to store all points explicitly here
- Instead, store *offset* of each point from one of the centers of near-optimal solution S
- For each point x, let c_x be the closest center of S and $y = c_x - x$
- Round y coordinate-wise to nearest power of $1 + \text{poly}\left(\frac{\varepsilon}{\log nd\Delta}\right)$ and store the vector of exponents $\tilde{y}$

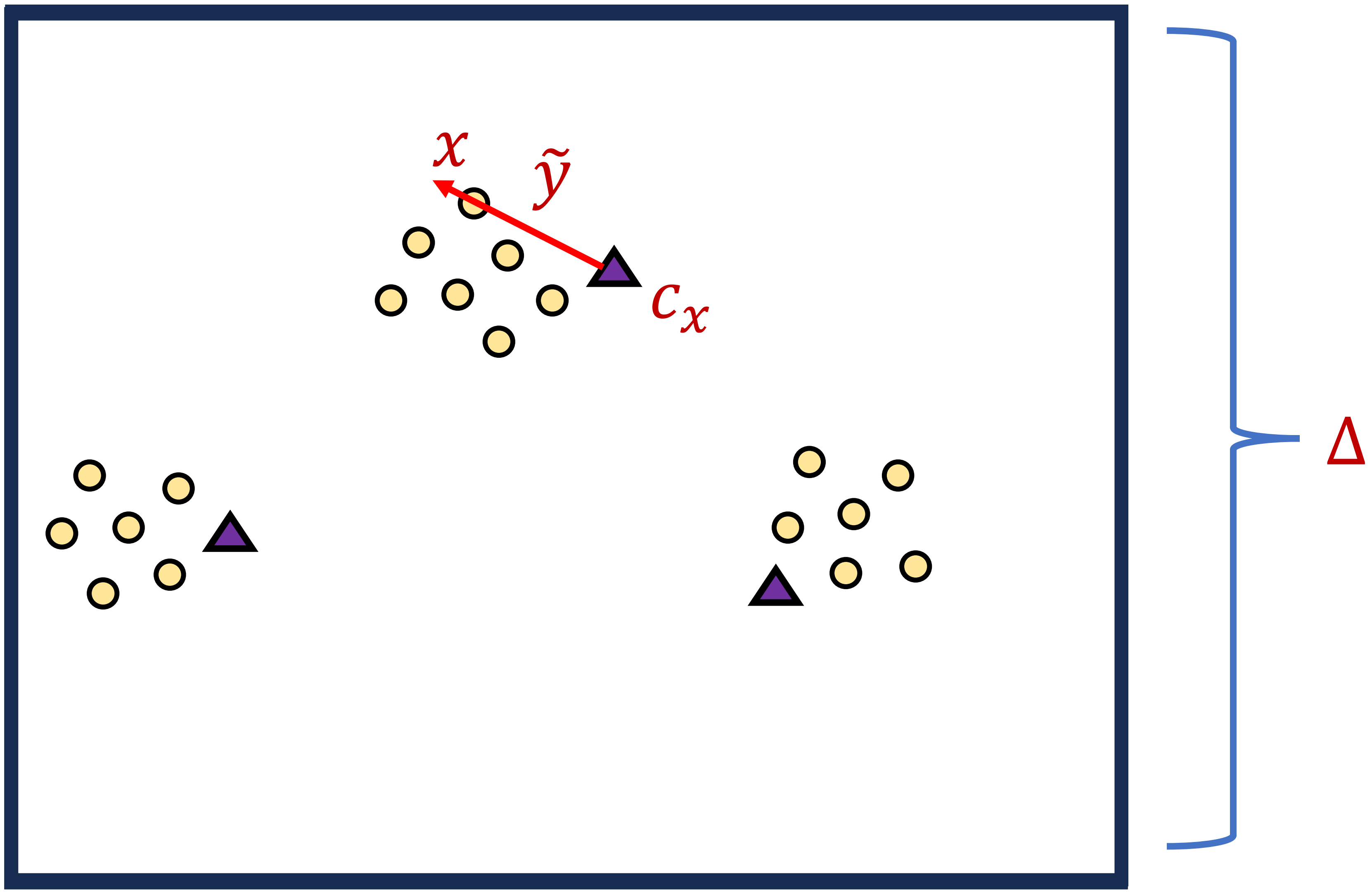
Quadtree Embedding



Quadtree Embedding



Quadtree Embedding



k-Means Framework

- First pass: set up the Wasserstein-z sketch
- Second pass:
  - Sample offsets of elements into a substream U' with probability proportional to their sensitivities
  - Run sparse recovery on U'

k-Means Framework

- We show the resulting samples forms a semi-coreset

- Sample $O(d^2 \log \Delta)$ points, each point using $d \cdot O\left(\log \frac{1}{\varepsilon} + \log\log nd\Delta\right)$

- Total space: $\tilde{O}\left(\frac{1}{\varepsilon^2}\right) \cdot \mathrm{poly}(d, k, \log\log n\Delta)$ words

Summary

- Insertion-only for (k, z)-clustering: One-pass streaming algorithm that uses $\tilde{O}\left(\frac{dk}{\varepsilon^2}\right) \cdot \min\left(k, \frac{1}{\varepsilon^z}\right) \cdot \text{poly}(\log \log n\Delta)$ words of space
- Insertion-deletion for k-median and k-means: Two-pass streaming algorithms that use $\tilde{O}\left(\frac{1}{\varepsilon^2}\right) \cdot \text{poly}(d, k, \log \log n\Delta)$ words of space
- Lower bounds: Even 2-approximation to the (k, z)-clustering cost *from a weighted subset of the input* uses $\Omega(\log^2 n)$ bits of space on insertion-deletion streams in one pass

Bounding Sum of Online Sensitivity

- Let $X = \{x_1, \ldots, x_n\} \subset [\Delta]^d$ and let t_{i-1} and t_i be times between which the optimal cost of the stream doubles
- Let K_i be the optimal clustering at time t_i and $\pi: X_{t_i} \to K_i$ be the mapping
- By triangle inequality,

$$\frac{\text{Cost}(x_t, C)}{\text{Cost}(X_t, C)} \le \frac{2^{z-1} \cdot \text{Cost}\big(x_t, \pi(x_t)\big)}{\text{Cost}(X_t, C)} + \frac{2^{z-1} \cdot \text{Cost}(\pi(x_t), C)}{\text{Cost}(X_t, C)}$$

Bounding Sum of Online Sensitivity

$$\varphi(x_t) = \frac{\text{Cost}(x_t, C)}{\text{Cost}(X_t, C)} \leq \frac{2^{z-1} \cdot \text{Cost}\big(x_t, \pi(x_t)\big)}{\text{Cost}(X_t, C)} + \frac{2^{z-1} \cdot \text{Cost}(\pi(x_t), C)}{\text{Cost}(X_t, C)}$$

- For $t \in (t_{i-1}, t_i]$, we have $\text{Cost}(X_t, C) > \frac{1}{2} \cdot \text{OPT}_i$
- By triangle inequality, $\frac{\text{Cost}(\pi(x_t), C)}{\text{Cost}(X_t, C)} \leq 3 \cdot \frac{2^{z-1}}{|S_t|}$, where S_t is the subset of X_t that maps to $\pi(x_t)$

$$\sum_{t \in (t_{i-1}, t_i]} \varphi(x_t) \leq \sum_{t \in (t_{i-1}, t_i]} \left(2^{z-1} + 3 \cdot \frac{2^{2z-2}}{|S_t|}\right)$$

Bounding Sum of Online Sensitivity

$$\sum_{t\in(t_{i-1},t_i]} \varphi(x_t) \le \sum_{t\in(t_{i-1},t_i]} \left(2^{z-1} + 3 \cdot \frac{2^{2z-2}}{|S_t|}\right)$$

- Since S_t is the subset of X_t that maps to $\pi(x_t)$ and can be one of k subsets, then $\sum_t S_t \le k\left(1 + \cdots + \frac{1}{n}\right) \le k \log n$
- Taking the sum over $O(\log nd\Delta)$ possible indices i, the sum of the online sensitivities is $O(2^{2z} k \log^2 nd\Delta)$

Lower Bound

- Any one-pass algorithm on insertion-deletion streams that outputs a 2-approximation to the (k, z)-clustering cost *at all times* in the stream with $d = \Omega(\log n)$ must use $\Omega(\log^2 n)$ bits of space
- Augmented Equality with Large Domain: Alice and Bob get $A, B \in [M]^n$ and Bob gets $j \in [n]$, $A_1, \ldots, A_{j-1}$ and must whether $A_j = B_j$
- Any protocol that succeeds w.h.p. requires $\Omega(n \log M)$ information cost

Lower Bound

- Augmented Equality with Large Domain: Alice and Bob get $A, B \in [M]^n$ and Bob gets $j \in [n], A_1, \ldots, A_{j-1}$ and must whether $A_j = B_j$
- Any protocol that succeeds w.h.p. requires $\Omega(n \log M)$ information cost
- Set $k = 1$ and write $X_i \in \{0,1\}^{\log M}$ in binary and insert $(100^z \log^2 n)^i$ copies of X_i
- Information cost of solving $O(\sqrt{n})$ copies of the problem

Lower Bound

- Any one-pass algorithm on insertion-deletion streams that outputs a 2-approximation to the (k, z)-clustering cost *from a weighted subset of the input* must use $\Omega(\log^2 n)$ bits of space
- Augmented Index with Large Domain: Alice gets $X \in [2^t]^m$ and Bob gets $j \in [m], X_1, \ldots, X_{j-1}$ and must output X_j
- Any constant probability protocol requires $\Omega(mt)$ bits of communication

Lower Bound

- Augmented Index with Large Domain: Alice gets $X \in [2^t]^m$ and Bob gets $j \in [m], X_1, \ldots, X_{j-1}$ and must output X_j
- Any constant probability protocol requires $\Omega(mt)$ bits of communication

- For $t = m = \log n$, map each point X_i to a lattice point between 7^{id} and 9^{id}, add $k - 1$ points at ∞
- Any 2-approximation using a weighted subset of the points must contain the exact point