

Data Stream Methods

Graham Cormode

graham@dimacs.rutgers.edu

S. Muthukrishnan

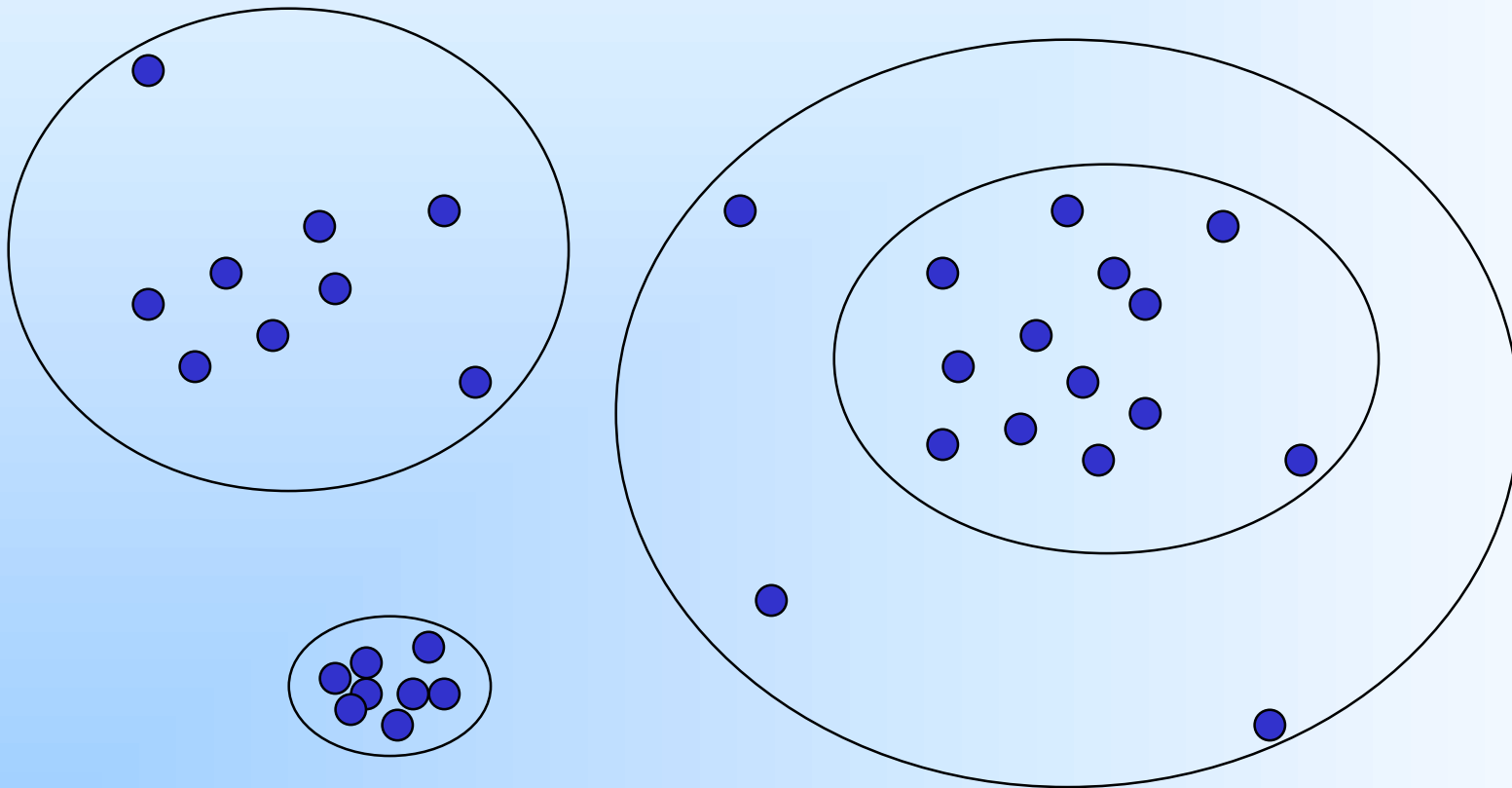
muthu@cs.rutgers.edu

Plan of attack

- Frequent Items / Heavy Hitters
- Counting Distinct Elements
- **Clustering items in Streams**
-

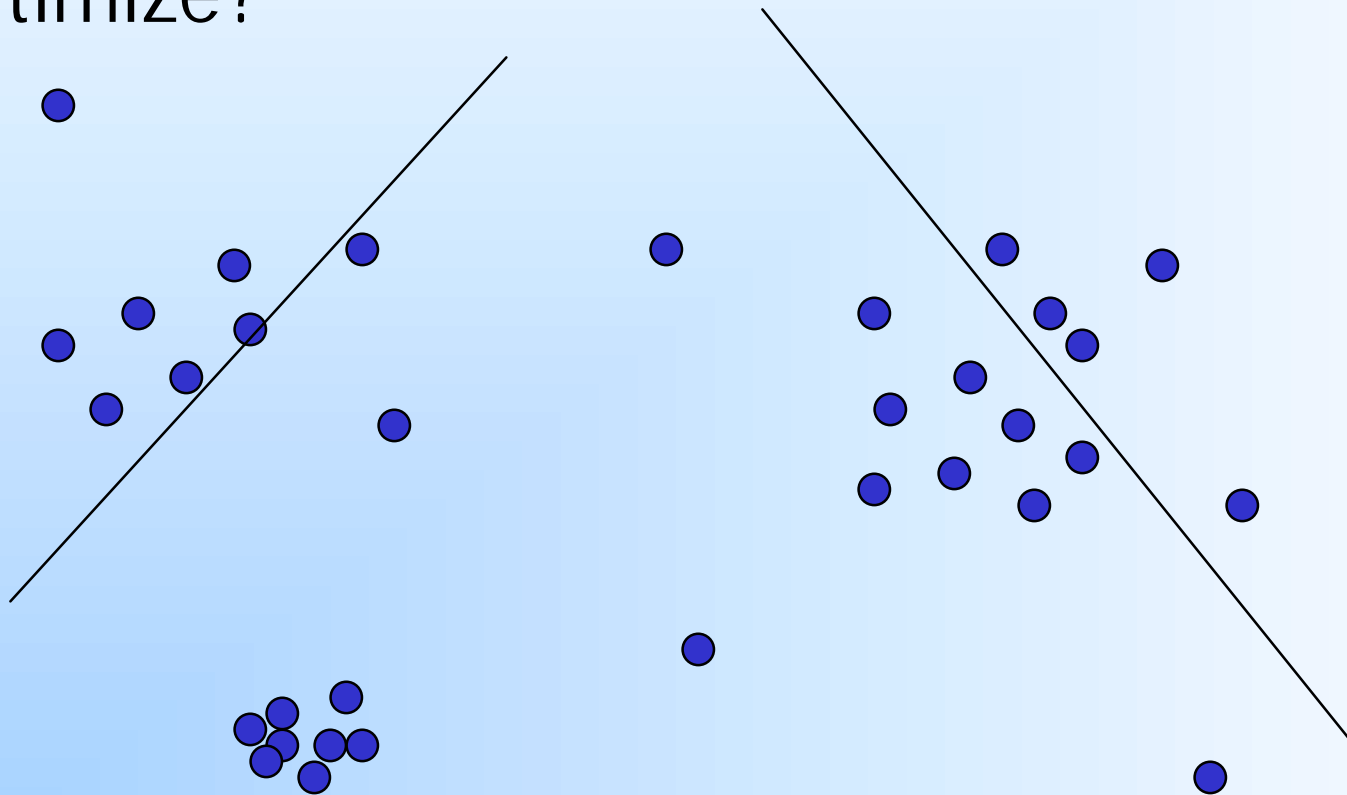
What is clustering?

- We have a bunch of items... we want to discover the clusters...



Types of clustering

- What is the quantity we are trying to optimize?



Two types of clustering

K-centers

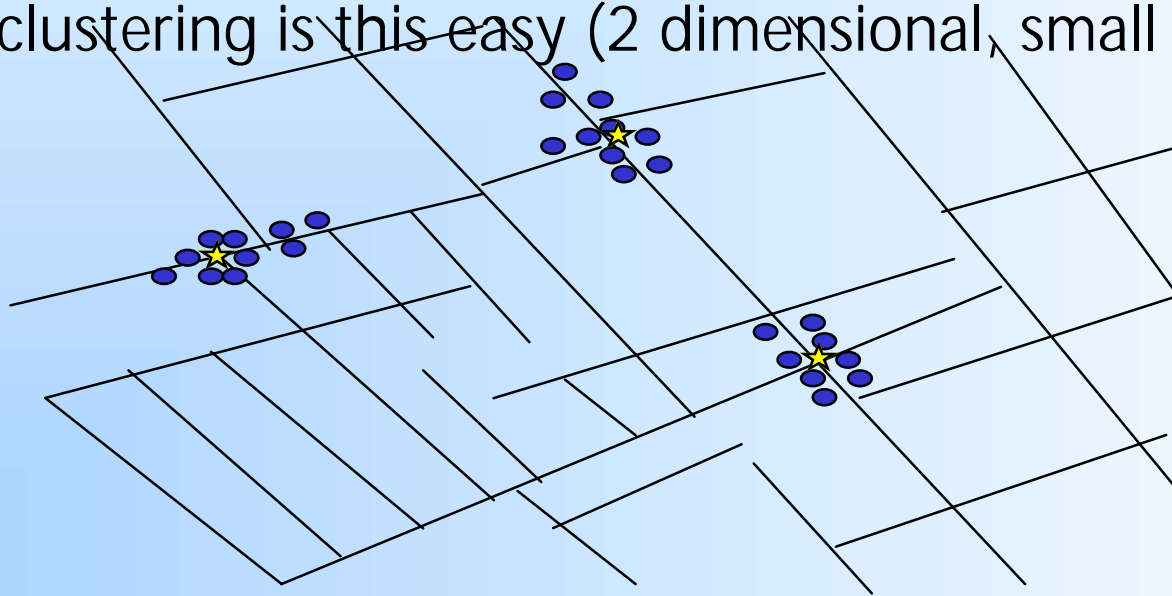
- Pick k points in the space, call these centers
- Assign each data point to its closest center
- Measure the diameter of each cluster: maximum distance between two points in the same cluster

K-medians

- Pick k points in the space, call these centers
- Assign each data point to its closest center
- Measure the average distance from each point to its closest center (or sum of distances)

The First? Application of Clustering

- A London physician plotted the location of cholera cases on a map during an outbreak in the 1850s.
- The locations indicated that cases were clustered around certain intersections where there were polluted wells -- thus exposing both the problem and the solution.
- Not all clustering is this easy (2 dimensional, small number of points.)



Slide due to Nina Mishra HP labs

Clustering is hard

- For both k-centers and k-medians, both versions are NP-complete
- We only know exponential algorithms to solve them
- So we look for approximate answers with guaranteed approximation ratios.

Metric Spaces

The algorithms will work for any metric space

Metric space: a set of points and a distance measure d on pairs of points satisfying

- Identity: $d(x,y) = 0 \Rightarrow x=y$
- Symmetry: $d(x,y) = d(y,x)$
- Triangle inequality: $d(x,z) \leq d(x,y) + d(y,z)$

Most distance measurements of interest are metric spaces: Euclidean distance, L1 distance, L_{inf} distance, edit distance...

Approximation for k-centers

From Gonzalez 1985

- Pick some point from the data as the first center
- Repeat:
 - For each data point, compute its distance d_{\min} from its closest center
 - Find the data point that maximizes d_{\min}
 - Add this point to the set of centers
- Until we have picked k centers

Gonzalez is a 2-approximation

- After we have picked k points to be centers, find the point that we would choose if we were allowed $k+1$ centers.
- Call its distance from its closest center d_{opt}
- We have $k+1$ points, every pair is separated by at least d_{opt}
- Any clustering into k sets must put some pair in the same set
- So **any** k -clustering must have diameter d_{opt}

Gonzalez is a 2-approximation

- Every point not selected as a center is at distance at most d_{opt} from its closest center
- So, for any two points allocated to the same center, they are both at distance at most d_{opt} from their closest center
- The distance between them is at most $2d_{\text{opt}}$: use triangle inequality and symmetry:
$$d(a,b) \leq d(a,c) + d(c,b)$$
- Diameter of any clustering must be at least d_{opt} , and is at most $2d_{\text{opt}}$ – so we have a 2 approximation.
- The diameter of the clustering is at most twice the best possible (could be a lot better, we don't know)

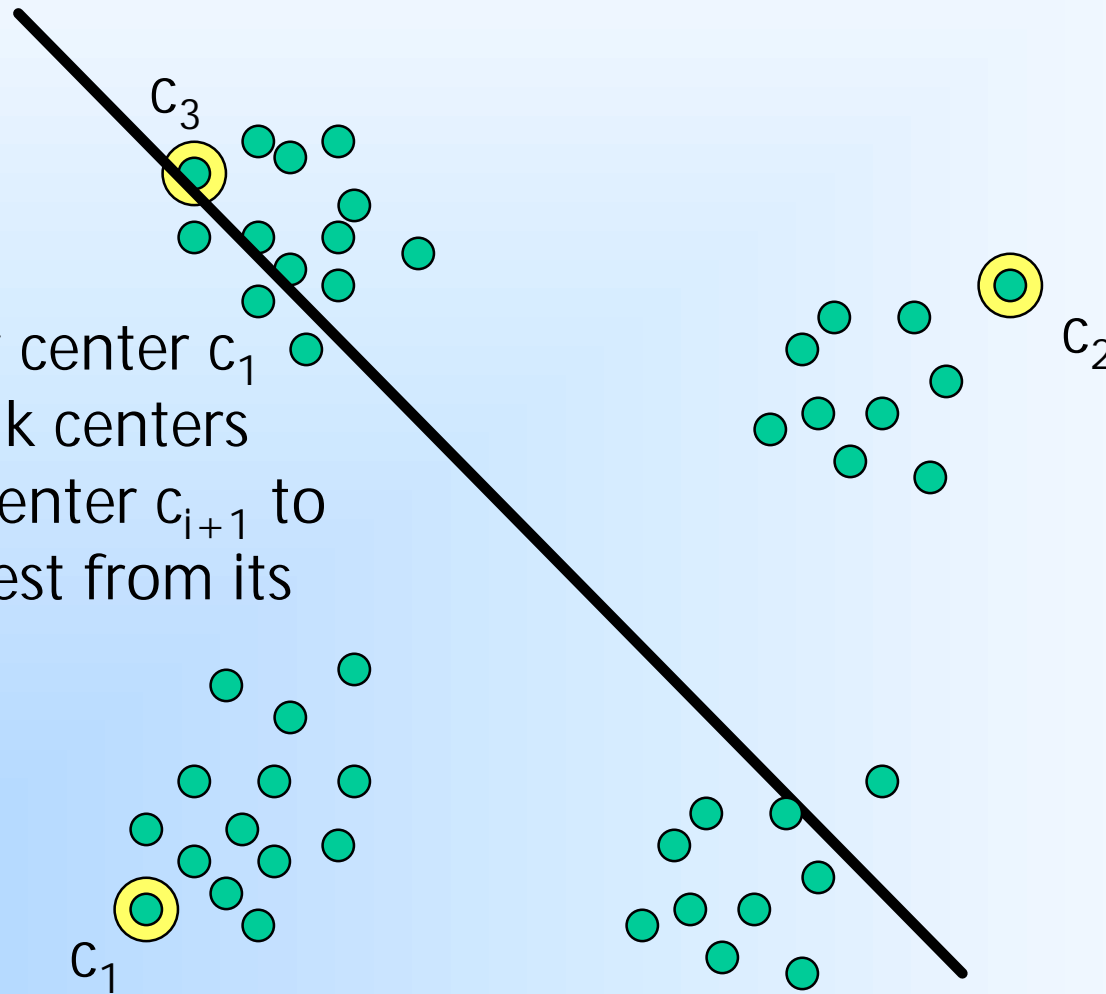
Gonzalez Clustering $k=4$

ALG:

Select an arbitrary center c_1

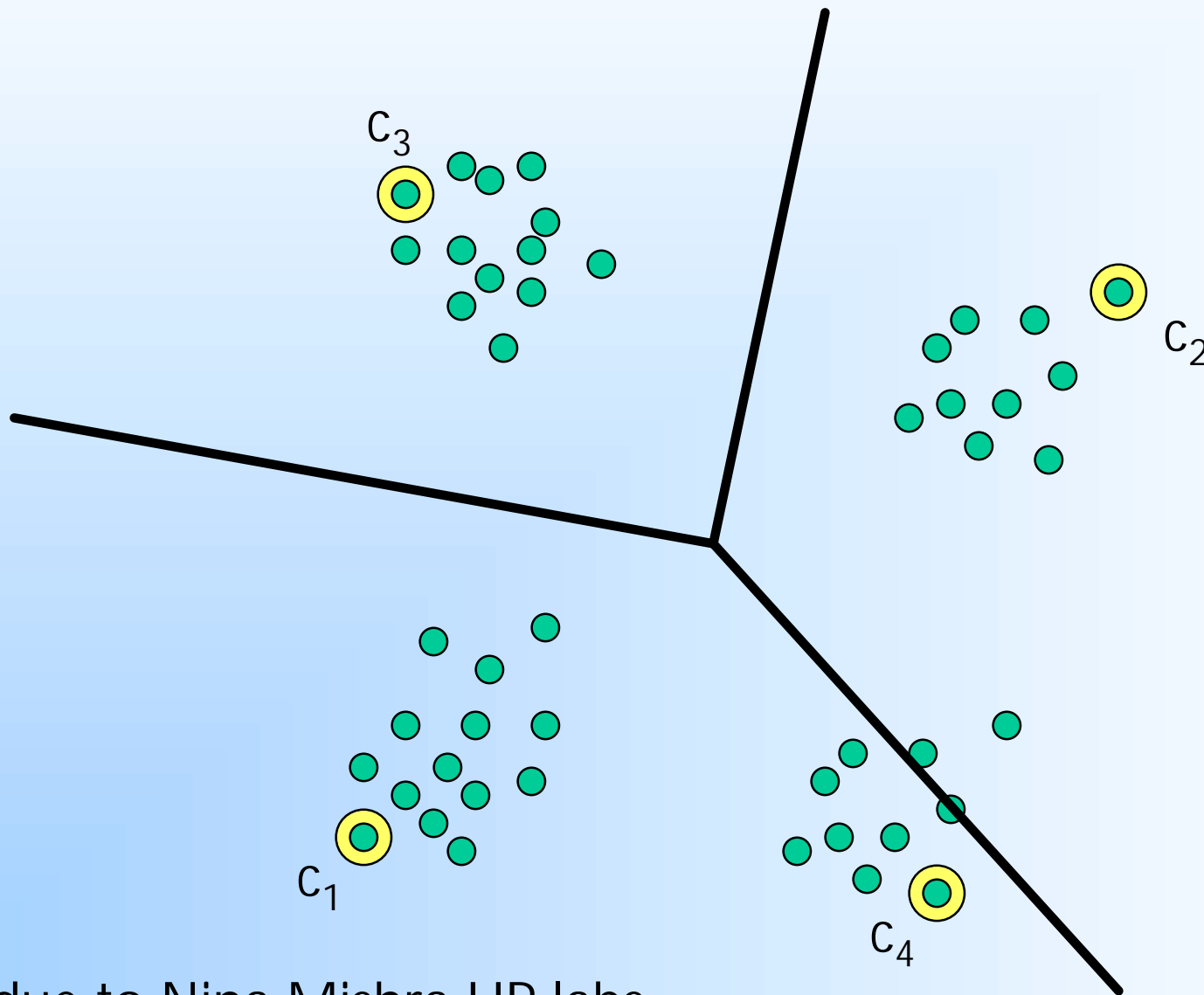
Repeat until have k centers

Select the next center c_{i+1} to be the one farthest from its closest center



Slide due to Nina Mishra HP labs

Gonzalez Clustering $k=4$



Slide due to Nina Mishra HP labs

Gonzalez Clustering $k=4$

Let $d = \max_{i \text{ and } p \text{ in } c_i} \text{dist}(c_i, p)$

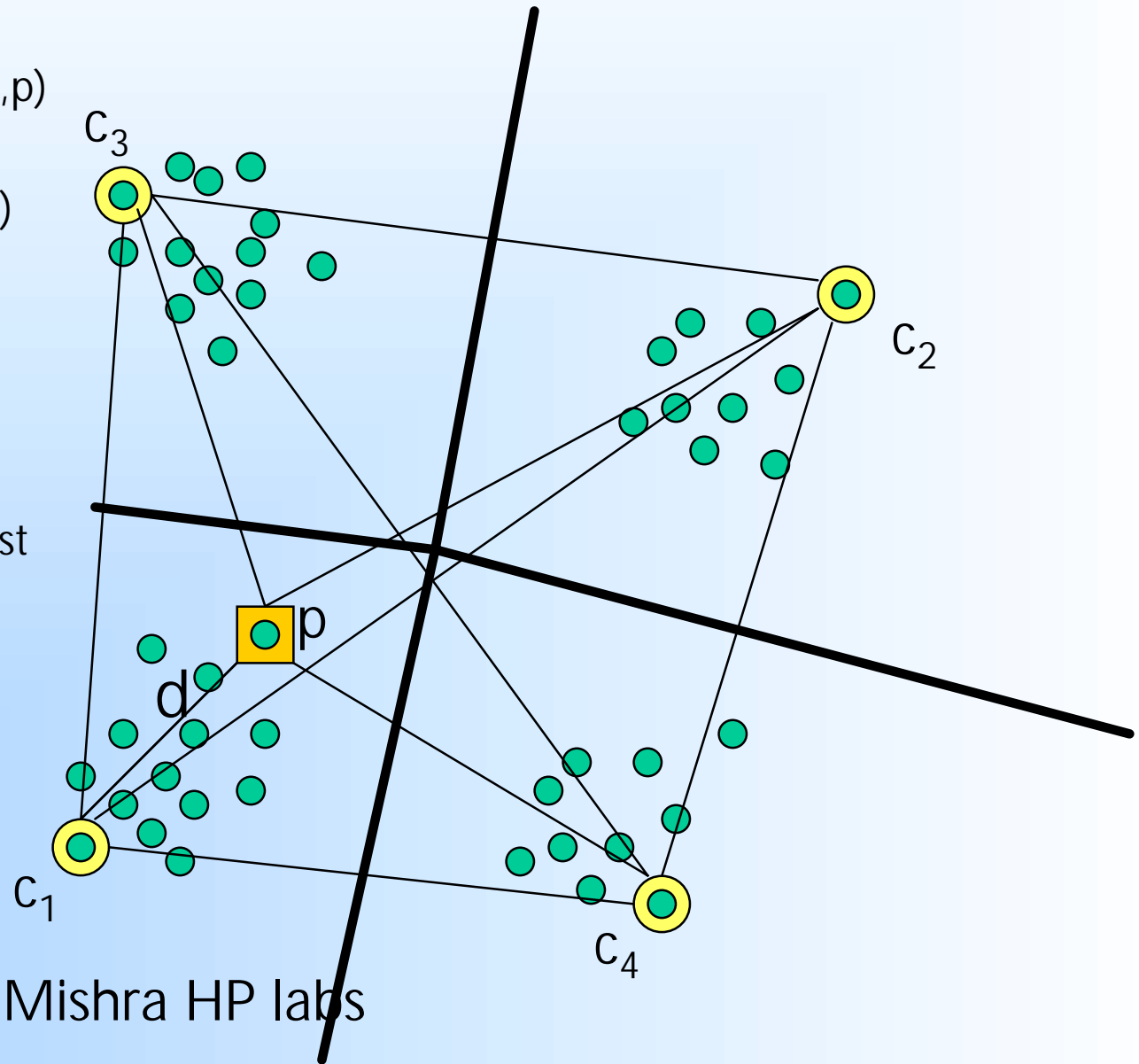
Claim: There exists a $(k+1)$ clique where each pair of points is distance $\geq d$.

- $\text{dist}(c_i, p) \geq d$ for all i
- $\text{dist}(c_i, c_j) \geq d$ for all i, j

Note: Any k -clustering must put at least two of these $k+1$ points in the same cluster.

- by pigeonhole

Thus: $d \leq 2\text{OPT}$



Slide due to Nina Mishra HP labs

Time and Space Costs

Require k passes over n data points ($k \ll n$)

Each pass computes a new center

If we store the current best center for each point, then each pass requires $O(1)$ time to update this for the new center, else $O(k)$ to compare to k centers.

So cost of Gonzalez algorithm is

- $O(kn)$ time with $O(n)$ space
- $O(k^2n)$ time with $O(k)$ space

Why Points from the data set?

- Why pick points from the data set? Why not pick arbitrary points?
- It depends on the kind of points we are dealing with... not always geometric points:
- Could be strings, web pages, database records... almost anything
- Not always reasonable to generate some arbitrary object as a cluster center, but always possible to use some observed item

Stream algorithm for clustering

What should the output of a stream algorithm for clustering be?

A classification of each input point?

Might this change as more input points arrive?

Two points which are initially put in different clusters might end up in the same one

An alternative is to output k -centers at end - any point can be classified using the centers

Will assume points arrive but do not depart – cash register model.

Gonzalez Restated

From Charikar, Chekuri, Feder, Motwani 97:

Suppose we knew d_{opt} for k -centers at the start

Do the following procedure:

- Select the first point as the first center
- For each point that arrives:
 - Compute d_{min} , the distance to the closest center
 - If $d_{\text{min}} > d_{\text{opt}}$ then set the new point to be a new center

Analysis

- d_{opt} is given to us, so we know that there are $k + 1$ points separated by $\geq d_{\text{opt}}$
- d_{opt} is as large as possible
- So there are $\leq k$ points separated by $> d_{\text{opt}}$
- The previous algorithm will output at most k centers: we only include a center when its distance is $> d_{\text{opt}}$ from every other center
- If we output more than k centers, then we have found more than k points separated by more than d_{opt} , contradicting the optimality of d_{opt} .
-

Analysis

- Every point not chosen to be a center is therefore $< d_{\text{opt}}$ from some center and so at most $2d_{\text{opt}}$ from any point allocated to the same center (triangle inequality again)
- So: if we are given d_{opt} then we will find a clustering where every point is at most twice this distance from its closest center
- Hence, still a 2-approximation
- But, we aren't given d_{opt}

Guessing the optimum solution

Suppose we guessed d_{opt} was between d and $2d$

Then we could run the algorithm

If we find more than k centers, then we guessed d_{opt} too low

So, in parallel, guess $d_{\text{opt}} = 1, 2, 4, 8 \dots$

We will reject everything less than d_{opt}

Our best guess will be $< 2d_{\text{opt}}$

So our output will be $< 2 * 2d_{\text{opt}} / d_{\text{opt}} = 4$ approx

Problems with guessing

How many copies do we need?

We need at most $\log_2 d_{\max}$ where d_{\max} is maximum distance between any pair of points, as $d_{\text{opt}} < d_{\max}$

Better: we need $\log_2 d_{\max}/d_{\text{smallest}}$, d_{smallest} is minimum distance between any pair of points, as $d_{\text{smallest}} < d_{\text{opt}}$

Problem: we are streaming, we don't necessarily know d_{\min} or d_{\max} in advance.

Keeping lots of copies in parallel is wasteful, although we can throw some away when we know our guess was too small

Doubling

- But we can take this idea, and do a doubling procedure when our guess is too small
- The Doubling Algorithm is divided into phases.
- Each phase begins with $k+1$ centers
- These are merged to get fewer centers.
- Then we process updates until the number of centers exceeds k again
- Begin by setting $k+1$ centers to the first $k+1$ points in the stream

Updating

In each phase i , we keep d_i so that every pair of centers is separated by at least d_i , and $d_i \leq d_{\text{opt}}$

- As before: compute d_{min} for each point in the stream.
- If $d_{\text{min}} > d_i$, then set the new point to be a new center
- Continue until there are $k+1$ centers, then merge

Merging

We have $k+1$ centers each at distance at least d_i

- Pick one arbitrarily, and discard all centers which are within $2d_i$ of this center.
- Repeat until all centers are separated by at least $2d_i$
- Set $d_{i+1} = 2d_i$

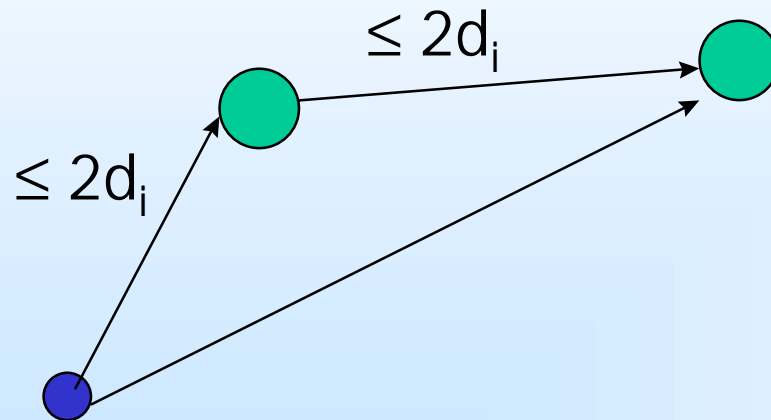
(we may not be able to remove any centers, but this doesn't matter)

We then go to phase $i+1$

Analyzing merging centers

- After merging, every pair of centers is separated by at least d_{i+1}
- Every point that has been processed is at most $2d_{i+1}$ from its closest center
- Proof by induction: before merging, every point that has been processed is at most $2d_i$ from its closest center
- We merge centers that are closer than $2d_i$
- So distance between any point and its new closest center is at most distance to old center + distance between centers = $2d_i + 2d_i = 4d_i = 2d_{i+1}$

Finishing the Induction



Base case:

The first $k+1$ (distinct) points are chosen as clusters

Set $d_0 =$ minimum distance between any pair

Every point is distance 0 from its closest center

And trivially, $0 \leq 2d_0$

Optimality Ratio

Before each merge, we know that there are $k+1$ points separated by d_i , so $d_{\text{opt}} \geq d_i$

At any point after a merge, we know that every point is at most $2d_{i+1}$ from its closest center

So we have a clustering where every pair of points in a cluster is within $4d_{i+1} = 8d_i$ of each other

$$8d_i / d_{\text{opt}} \leq 8d_{\text{opt}} / d_{\text{opt}} = 8$$

So a factor 8 approximation

Time to process each point = compare upto k centers
= $O(k)$ comparisons

Time analysis

Merging is potentially costly: may need to compare $O(k^2)$ items, may need to do several merges in succession. Will use an amortized argument.

Keep a heap of distances between centers.

Inserting a new center means adding k distances, time for this is $O(k \log k)$

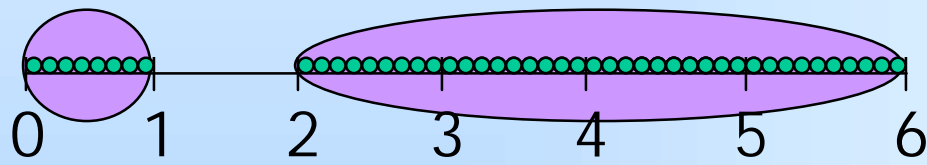
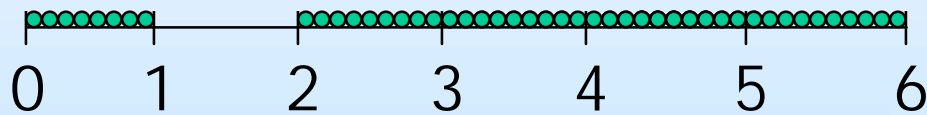
Charge $\log k$ to each distance, total credit is $O(k \log k)$

When merging, we remove some distances

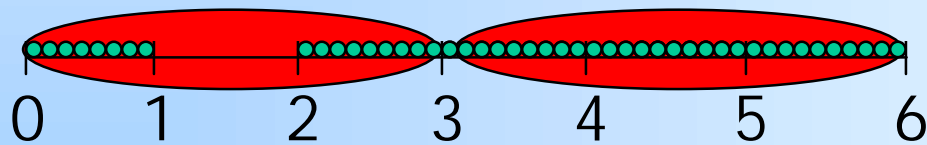
Takes $O(\log k)$ to remove each one, pay for this with the credit.

Total time to process each item is amortized $O(k \log k)$

K-Center not always the "right" clustering measure



Max radius = 2



Max radius = 1.5

Slide due to Nina Mishra HP labs

K-medians

k-medians measures the quality based on the average distance between points and their closest median.

So: $\sum_{p_1} d(p_1, \text{median}(p_1)) / n$

We can forget about the $/n$, and focus on minimizing the sum of all point-median distances

Note here, outlier points do not help us lower bound the minimum cluster size

We will assume that we have an exact method for k-medians which we will run on small instances.

Results from Guha, Misra, Motwani & O'Callaghan '00

Divide and conquer

Suppose we are given n points to cluster.

Split them into $n^{1/2}$ groups of $n^{1/2}$ points.

Cluster each group in turn to get k -medians.

Then cluster the group of k -medians to get a final set.

The space required is $n^{1/2}$ for each group of points, and $kn^{1/2}$ for all the intermediate medians.

Need to analyze the quality of the resultant clustering in terms of the optimal clustering for the whole set of points.

Analysis

Firstly, analyze the effect of picking points from the input as the medians, instead of arbitrary points

Consider optimal solution. Point p is allocated to median m .

Let q be the point closest to m from the input

$$d(p,q) \leq d(p,m) + d(q,m) \leq 2d(p,m)$$

(since q is closest, $d(q,m) \leq d(p,m)$)

So using points from the input at most doubles the distance.

Analysis

Next, what is cost of dividing points into separate groups, and clustering each?

Consider the total cost (=sum of distances) of the optimum for the groups C , & the overall optimum C^*

Suppose we choose the medians from the points in each group.

The “optimum” medians are not present in each group, but we can use the closest point in each group to the optimum median.

Then $C \leq 2C^*$ using the previous result.

How to recluster

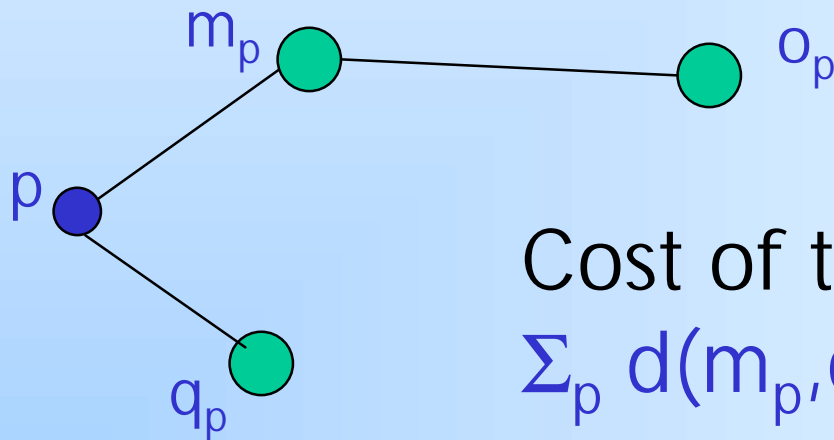
- After processing all groups, we have $n^{1/2}$ sets of k -medians.
- For each median, we should record how many points were allocated to it by the initial clustering.
- This gives a “weight” of that point
- We cluster again, using the weights to represent that many points at that location.

Cost of reclustering

What is the cost of this reclustering?

Each point p is allocated to some median m_p , which is then reclustered to some new median o_p .

Let the optimal k-median for point p be q_p



Cost of the reclustering is
 $\sum_p d(m_p, o_p)$

Cost of reclustering

$$\sum_p d(m_p, o_p) \leq \sum_p d(m_p, q_p)$$

Because o_p is the optimal median for m_p , then the sum of distances to the q_p s must be more

$$\begin{aligned} \sum_p d(m_p, q_p) &\leq \sum_p d(m_p, p) + d(p, q_p) \\ &= \text{Cost of first clustering} \\ &\quad + \text{cost of optimal clustering} \\ &= C + C^* \end{aligned}$$

If we restrict to using points from the original dataset, then we at most double this to $2(C + C^*)$.

Total cost = $2(C + C^*) + C \leq 8C^*$ using previous result

Approximate version

The previous analysis assumes a method capable of producing the optimum k-medians clustering

This is expensive; in practice, we find solutions with cost guaranteed to be within c times the optimal.

So $C \leq 2cC^*$ and $\sum_p d(m_p, o_p) \leq c \sum_p d(m_p, q_p)$

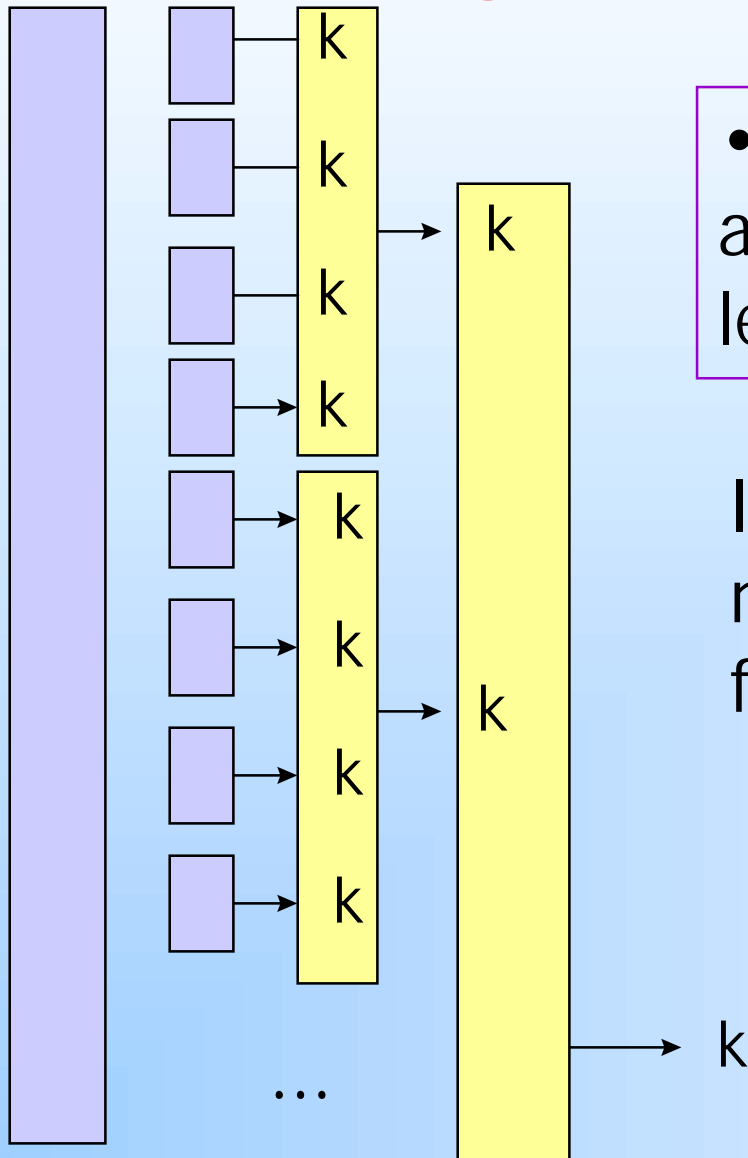
Putting this together gives a bound of

$$[2c(2C + C^*) + C]/C^* = 2c(2c + 1) + 2c = 4c(c + 1)$$

Reducing space further

- This uses $O(kn^{1/2})$ space, which is still a lot
- We can use this procedure as a black box, and repeatedly merge clusterings.
- The approximation factor will get worse the more levels we include (with one level it is $O(c)$, with two it is $O(c^2)$, with i it is $O(c^i)$)
- Need to use a tree structure approach:

Clustering with small Memory



- A factor is lost in the approximation with each level of divide and conquer

In general, if $|\text{Memory}| = n^\epsilon$, need $1/\epsilon$ levels, approximation factor $2^{O(1/\epsilon)}$

- If $n = 10^{12}$ and $M = 10^6$, then regular 2-level algorithm
- If $n = 10^{12}$ and $M = 10^3$ then need 4 levels, approximation factor 2^4

Conclusions

- We have seen methods for clustering points using small space
- Some are easier or more efficient than others, depending on the kind of clustering desired.
- Two basic techniques:
 - Guessing the parameter and doubling when it is shown to be too small
 - A hierarchic approach: build a tree structure on the data stream, and only keep the path to the current leaf in memory.

Final Conclusions

- A lot of important database and data mining questions can be solved on the data stream
- Exact answers are unlikely: instead we apply approximation and randomization to keep memory requirements low
- Need tools from algorithms, statistics & database to design and analyze these methods.
- Proofs & proof of concepts are important.

References

- "Clustering to minimize the maximum intercluster distance" T. Gonzalez, Theoretical Computer Science, 1985
- "Incremental clustering and dynamic information retrieval", Charikar, Chekuri, Feder, Motwani, STOC 1997
- "Clustering Data Streams" Guha, Mishra, Motwani, O'Callaghan, FOCS 2000