Cluster and Data Stream Analysis

Graham Cormode
cormode@bell-labs.com

Outline

- Cluster Analysis
  - Clustering Issues
  - Clustering algorithms:
    Hierarchical Agglomerative Clustering, K-means, Expectation Maximization, Gonzalez approximation for K-center
- Data Stream Analysis
  - Massive Data Scenarios
  - Distance Estimates for High Dimensional Data:
    Count-Min Sketch for L_∞, AMS sketch for L_2, Stable sketches for L_p, Experiments on tabular data
  - Too many data points to store:
    Doubling Algorithm for k-center clustering, Hierarchical Algorithm for k-median, Grid algorithms for k-median
- Conclusion and Summary

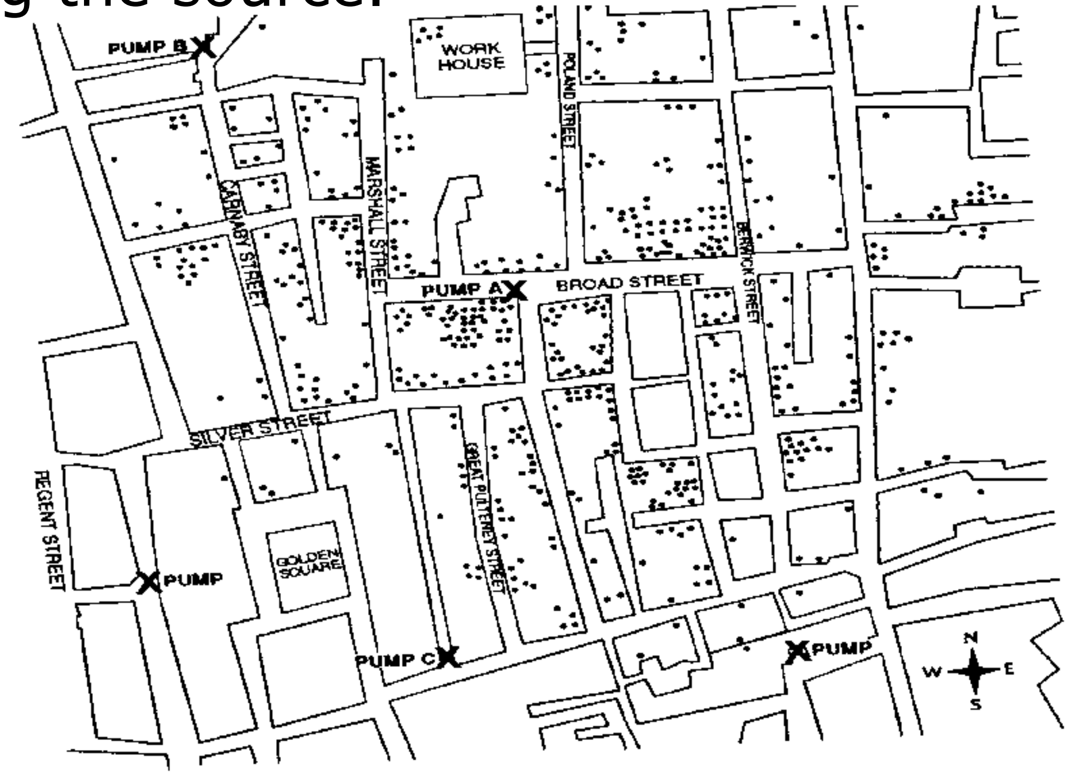
1. Cluster Analysis

An Early Application of Clustering

John Snow plotted the location of cholera cases on a map during an outbreak in the summer of 1854.

His hypothesis was that the disease was carried in water, so he plotted location of cases and water pumps, identifying the source.

Clusters easy to identify visually in 2 dimensions… more points and higher dimension?



Clustering Overview

Clustering has an intuitive appeal

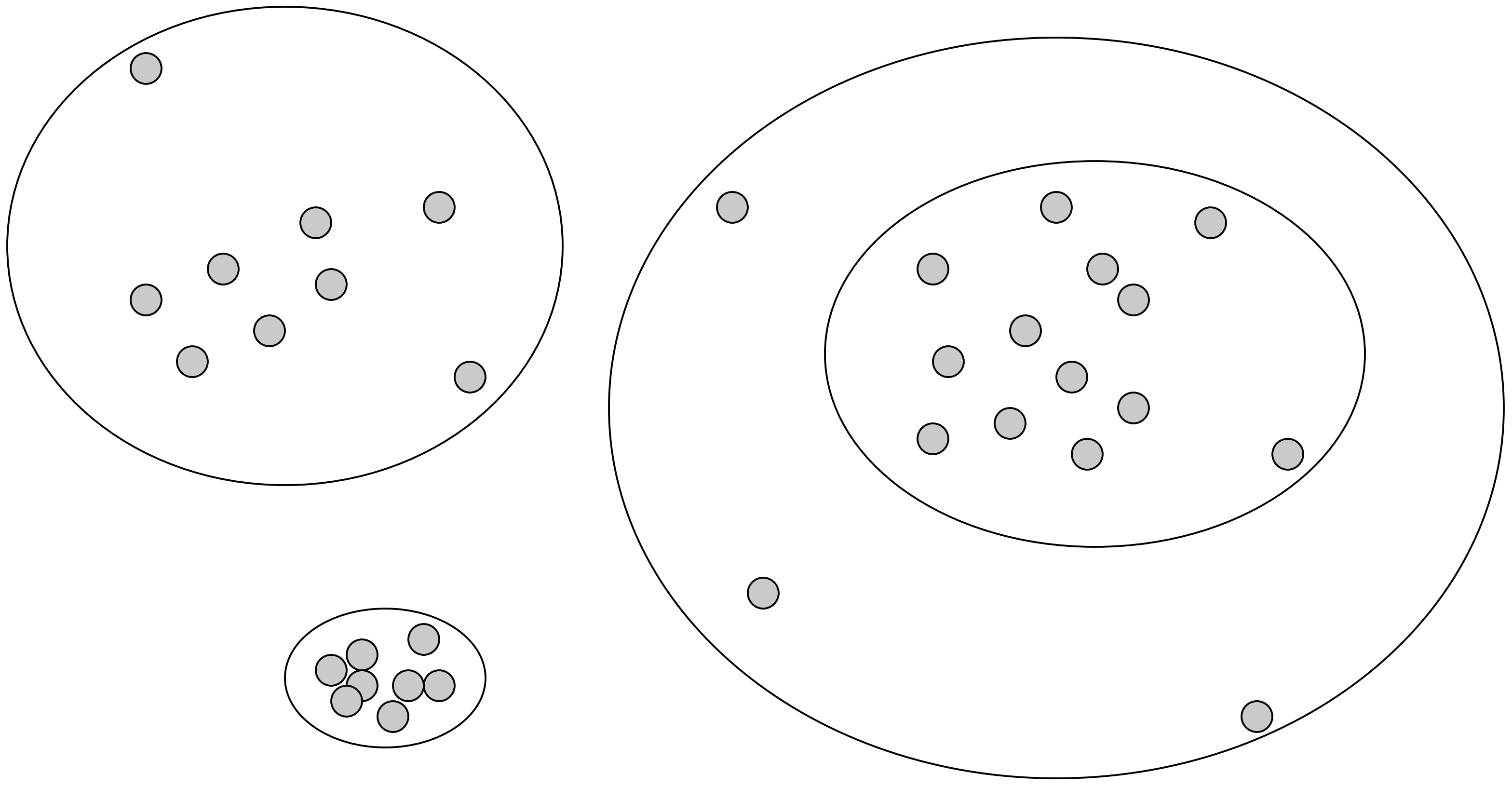
We often talk informally about “clusters”: ‘cancer clusters’, ‘disease clusters’ or ‘crime clusters’

Will try to define what is meant by clustering, formalize the goals of clustering, and give algorithms for clustering data

My background: algorithms and theory, so will have algorithmic bias, less statistical

What is clustering?

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



Unsupervised Learning

Supervised Learning: training data has labels (positive/negative, severity score), and we try to learn the function mapping data to labels

Clustering is a case of unsupervised learning: there are no labeled examples

We try to learn the "classes" of similar data, grouping together items we believe should have the same label

Harder to evaluate "correctness" of clustering, since no explicit function is being learned to check against.

Will introduce objective functions so that we can compare two different clusterings of same data

Why Cluster?

What are some reasons to use clustering?

- It has intuitive appeal to identify patterns
- To identify common groups of individuals (identifying customer habits; finding disease patterns)
- For data reduction, visualization, understanding: pick a representative point from each cluster
- To help generate and test hypotheses: what are the common factors shared by points in a cluster?
- A first step in understanding large data with no expert labeling.

Before we start...

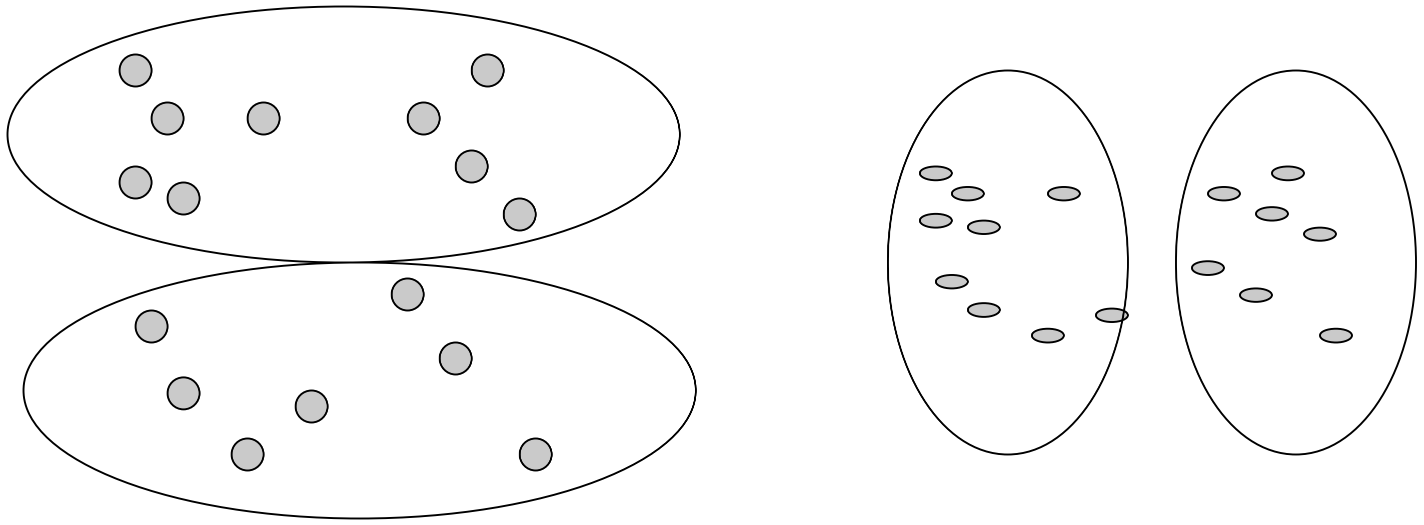
Before we jump into clustering, pause to consider:

- Data Collection – need to collect data to start with
- Data Cleaning – need to deal with imperfections, missing data, impossible values (age > 120?)
- How many clusters - Often need to specify k, desired number of clusters to be output by algorithm
- Data Interpretation – what to do with clusters when found? Cholera example required hypothesis on water for conclusion to be drawn
- Hypothesis testing – are the results significant? Can there be other explanations?

Distance Measurement

How do we measure distance between points?

In 2D plots it is obvious – or is it?



What happens when data is not numeric, but contains mix of time, text, boolean values etc.?

How to weight different attributes?

Application dependent, somewhat independent of algorithm used (but some require Euclidean distance)

Metric Spaces

We assume that the distances form a metric space

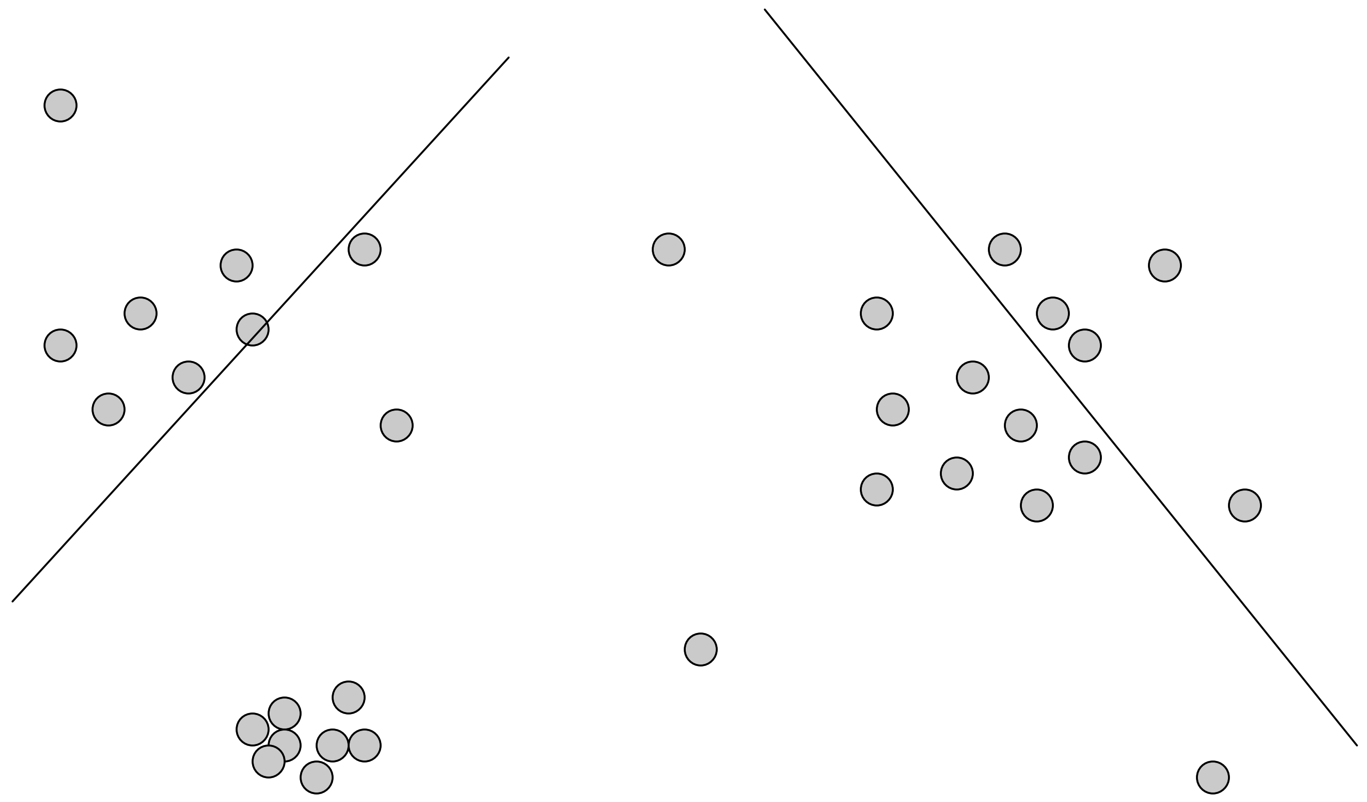
Metric space: a set of points and a distance measure d on pairs of points satisfying

- 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, L_1 distance, L_∞ distance, edit distance, weighted combinations...

Types of clustering

What is the quantity we are trying to optimize?



Two objective functions

K-centers

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

K-medians

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

Clustering is hard

For both k-centers and k-medians on distances like 2D Euclidean, it is NP-Complete to find best clustering.

(We only know exponential algorithms to find them exactly)

Two approaches:

- Look for approximate answers with guaranteed approximation ratios.
- Look for heuristic methods that give good results in practice but limited or no guarantees

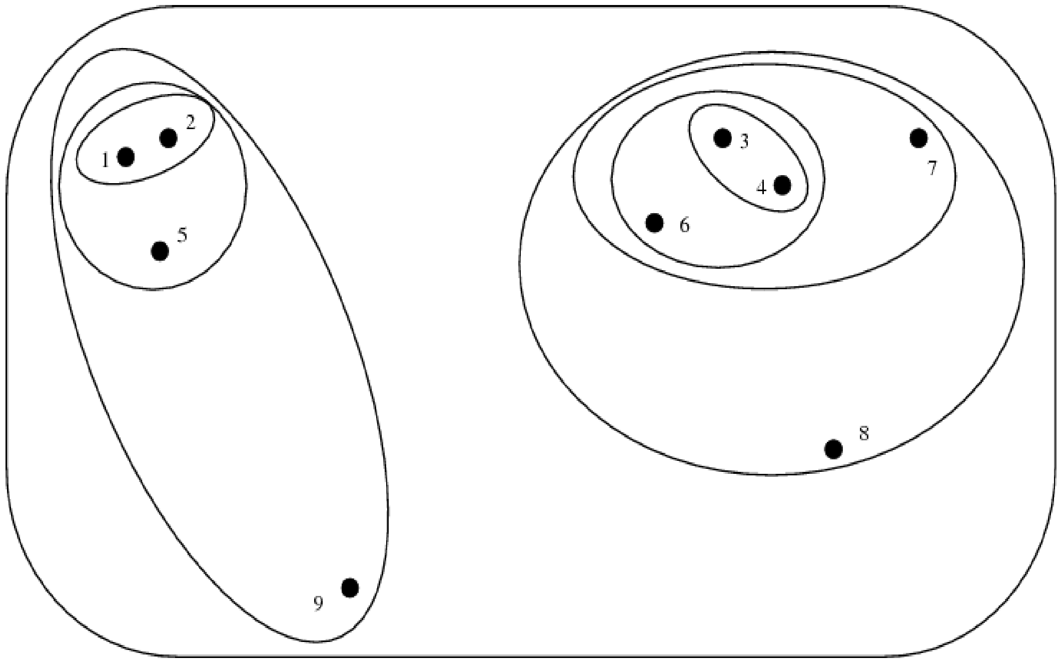
Hierarchical Clustering

Hierarchical Agglomerative Clustering (HAC) has been reinvented many times. Intuitive:

```
Make each input point into an input cluster.
  Repeat: merge closest pair of clusters, until
  a single cluster remains.
```

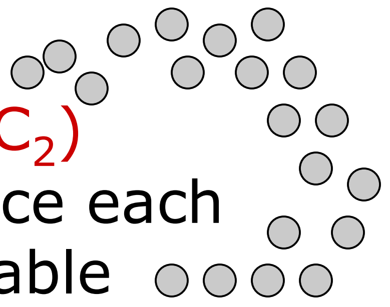
To find k clusters: output last k clusters.

View result as binary tree structure: leaves are input points, internal nodes correspond to clusters, merging up to root.

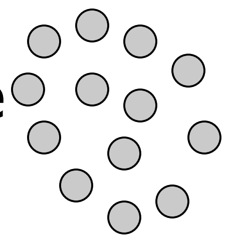


Types of HAC

Big question: how to measure distance between clusters to find the closest pair?

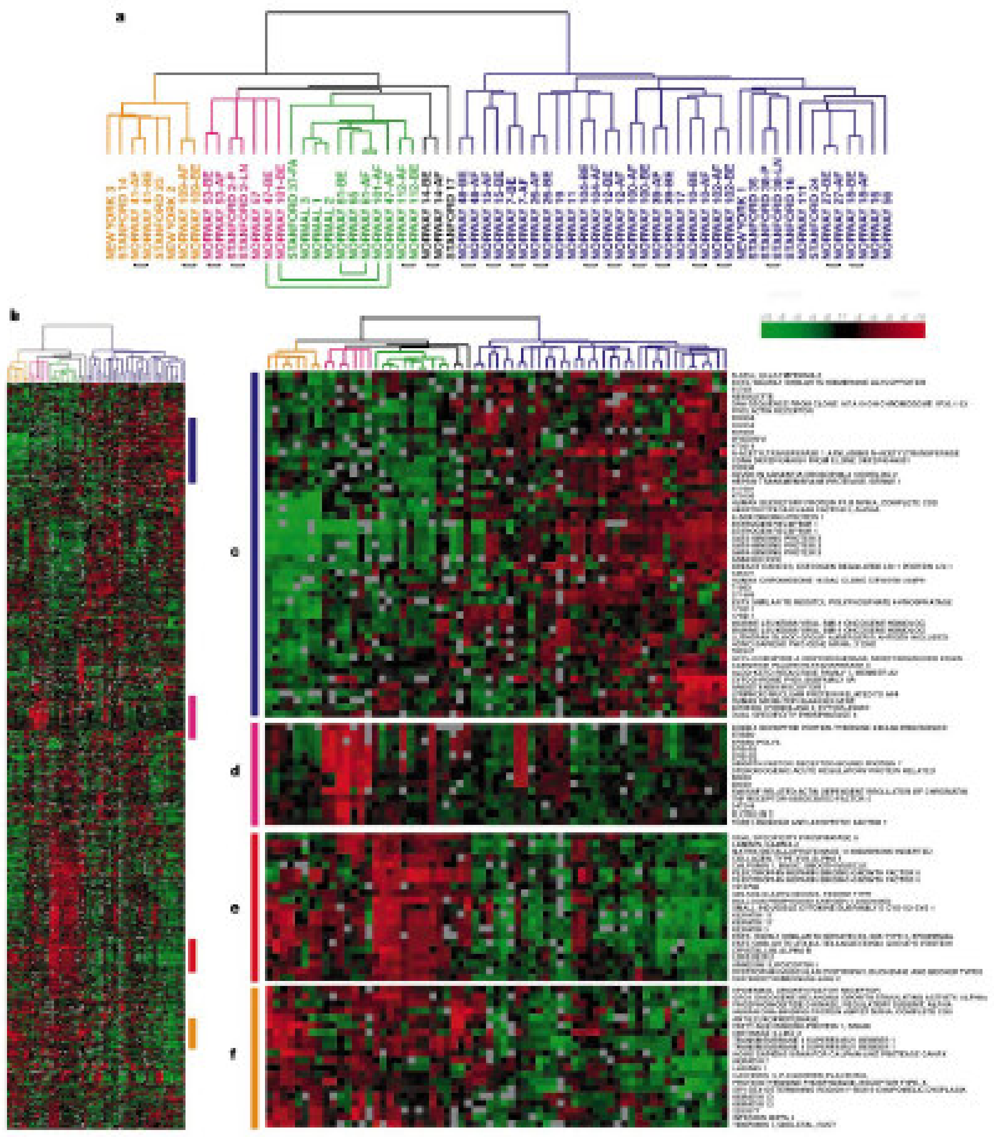


- Single-link: $d(C_1, C_2) = \min d(c_1 \in C_1, c_2 \in C_2)$
  Can lead to “snakes”: long thin clusters, since each point is close to the next. May not be desirable



- Complete-link: $d(C_1, C_2) = \max d(c_1 \in C_1, c_2 \in C_2)$
  Favors circular clusters… also may not be desirable
- Average-link: $d(C_1, C_2) = \text{avg } d(c_1 \in C_1, c_2 \in C_2)$
  Often thought to be better, but more expensive to compute…

HAC Example



Popular way to study gene expression data from microarrays.

Use the cluster tree to create a linear order of (high dimensional) gene data.

Cost of HAC

Hierarchical Clustering can be costly to implement:

Initially, there are $\Theta(n^2)$ inter-cluster distances to compute.

Each merge requires a new computation of distances involving the merged clusters.

Gives a cost of $O(n^3)$ for single-link and complete-link

Average link can cost as much as $O(n^4)$ time

This limits scalability: with only few hundred thousand points, the clustering could take days or months.

Need clustering methods that take time closer to $O(n)$ to allow processing of large data sets. ❑

K-means

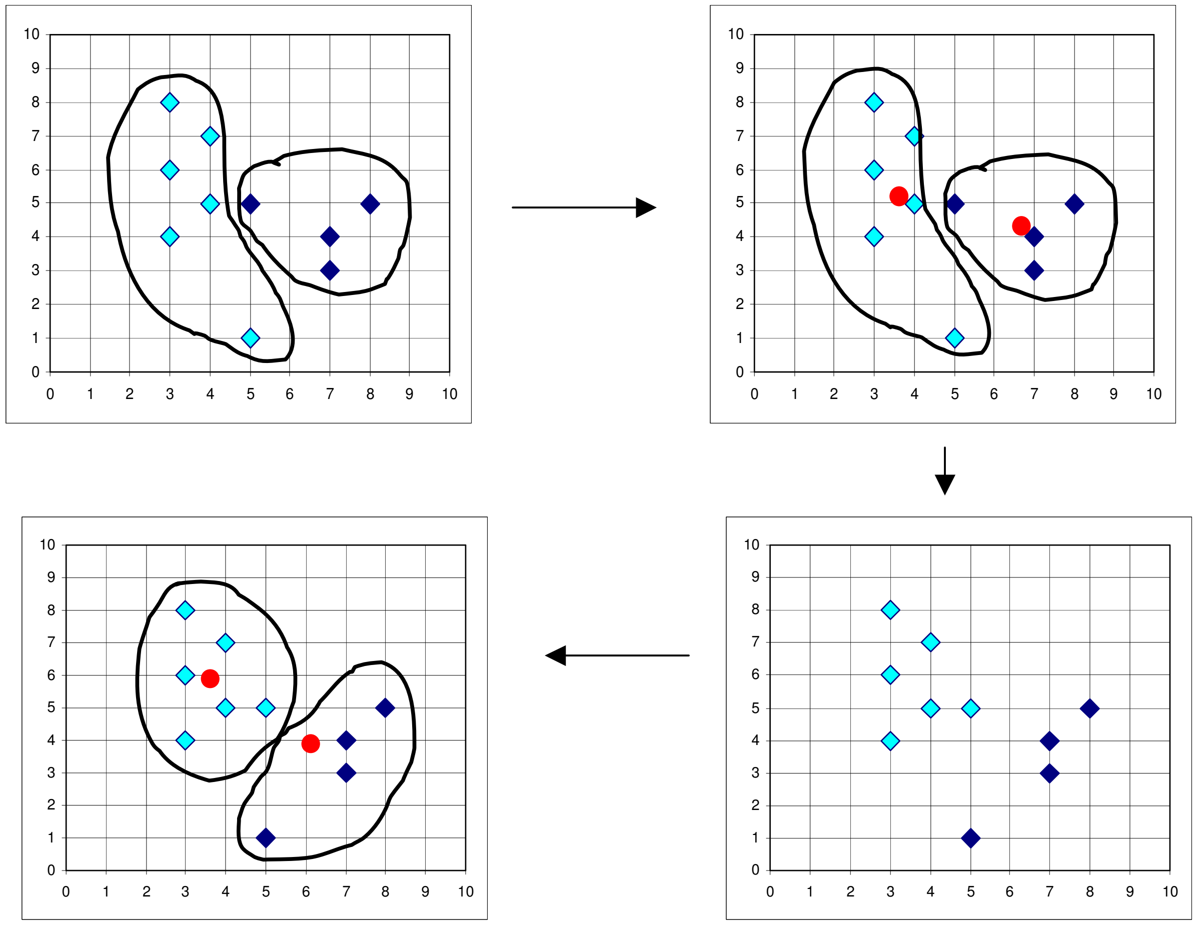
K-means is a simple and popular method for clustering data in Euclidean space.

It finds a local minimum of the objective function that is average sum of squared distance of points from the cluster center.

```
Begin by picking k points randomly from the data
  Repeatedly alternate two phases:
    Assign each input point to its closest center
    Compute centroid of each cluster (average point)
    Replace cluster centers with centroids
  Until converges / constant number of iterations
```

K-means example

Example due to Han and Kanber:



K-means issues

- Results not always ideal:
  - if two centroids are close to each other, one can “swallow” the other, wasting a cluster
  - Outliers can also use up clusters
  - Depends on initial choice of centers: repetition can improve the results
- (Like many other algorithms) Requires k to be known or specified up front, hard to tell what is best value of k to use
- But, is fast – each iteration takes time at most O(kn), typically requires only a few iterations to converge. ❑

Expectation Maximization

Think of a more general and formal version of k-means

Assume that the data is generated by some particular distribution, eg, by k Gaussian dbns with unknown mean and variance.

Expectation Maximization (EM) looks for parameters of the distribution that agree best with the data.

Also proceeds by repeating an alternating procedure:

- Given current estimated dbn, compute likelihood for each data point being in each cluster.
- From likelihoods, data and clusters, recompute parameters of dbn

Until result stabilizes or after sufficient iterations

Expectation Maximization

- Cost and details depend a lot on what model of the probability distribution is being used:
  mixture of Gaussians, log-normal, Poisson, discrete, combination of all of these…
- Gaussians often easiest to work with, but is this a good fit for the data?
- Can more easily include categorical data, by fitting a discrete probability distribution to categorical attributes
- Result is a probability distribution assigning probability of membership to different clusters
  From this, can fix clustering based on maximum likelihood. ❑

Approximation for k-centers

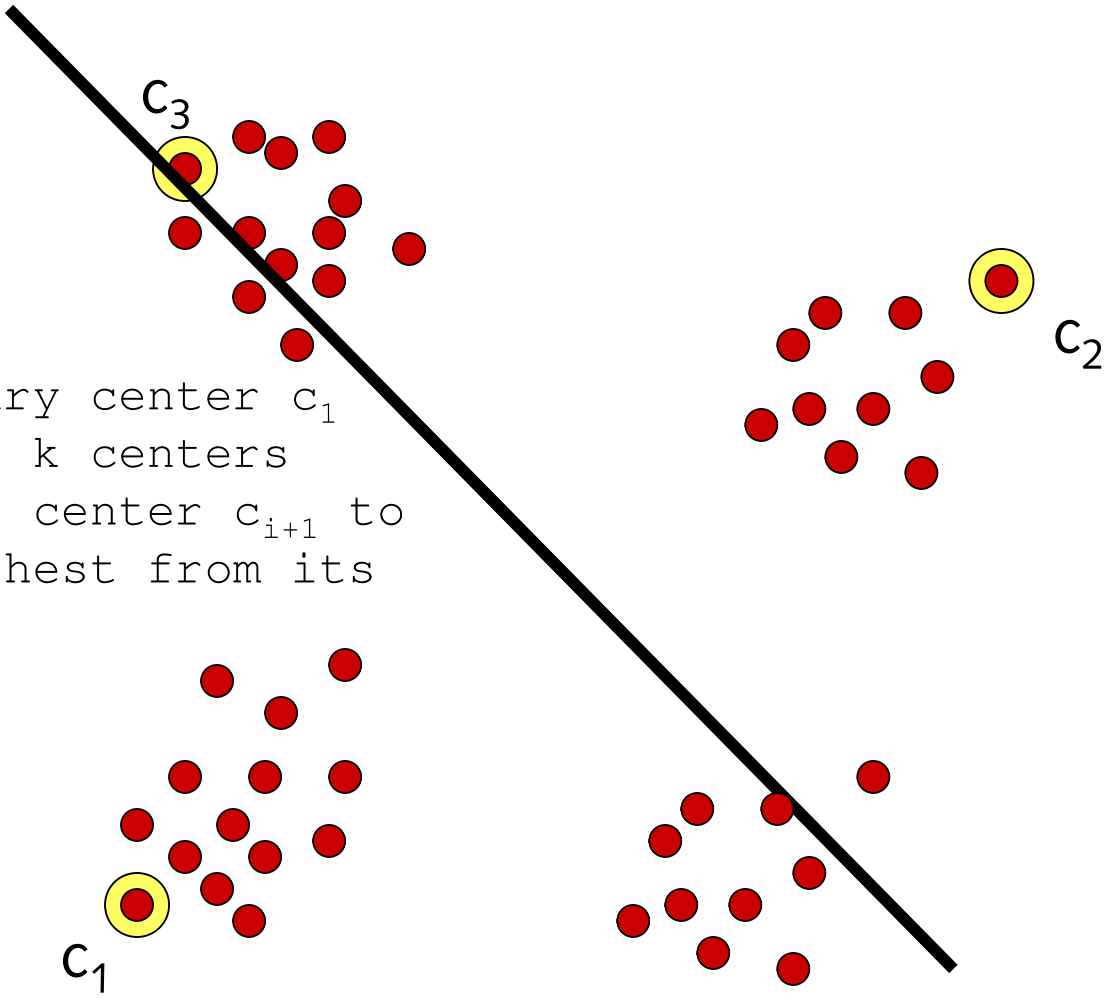
Want to minimize diameter (max dist) of each cluster.

```
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 k centers are picked
```

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 time cost is O(kn) [Gonzalez, 1985].

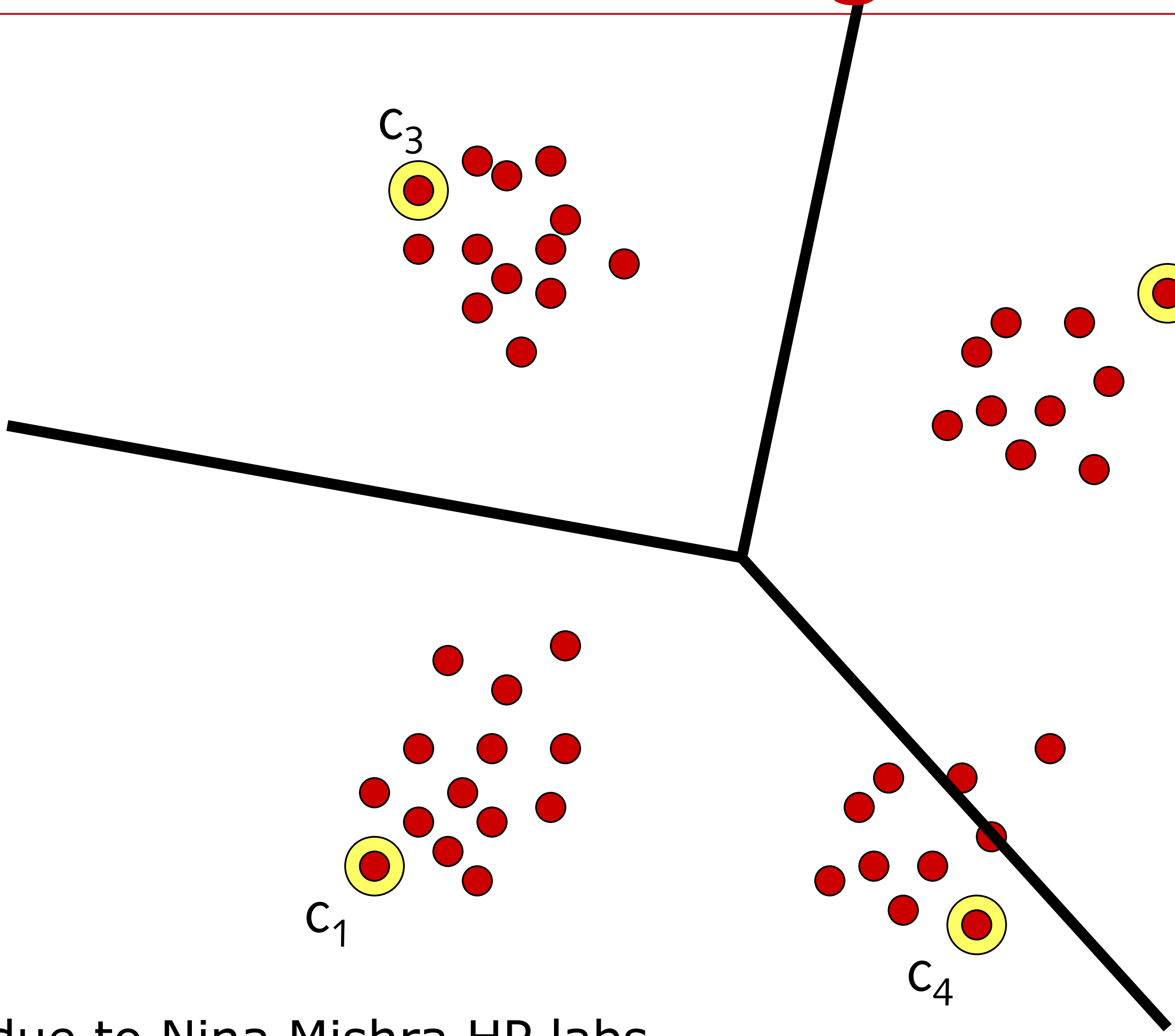
Gonzalez Clustering k=4



```
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

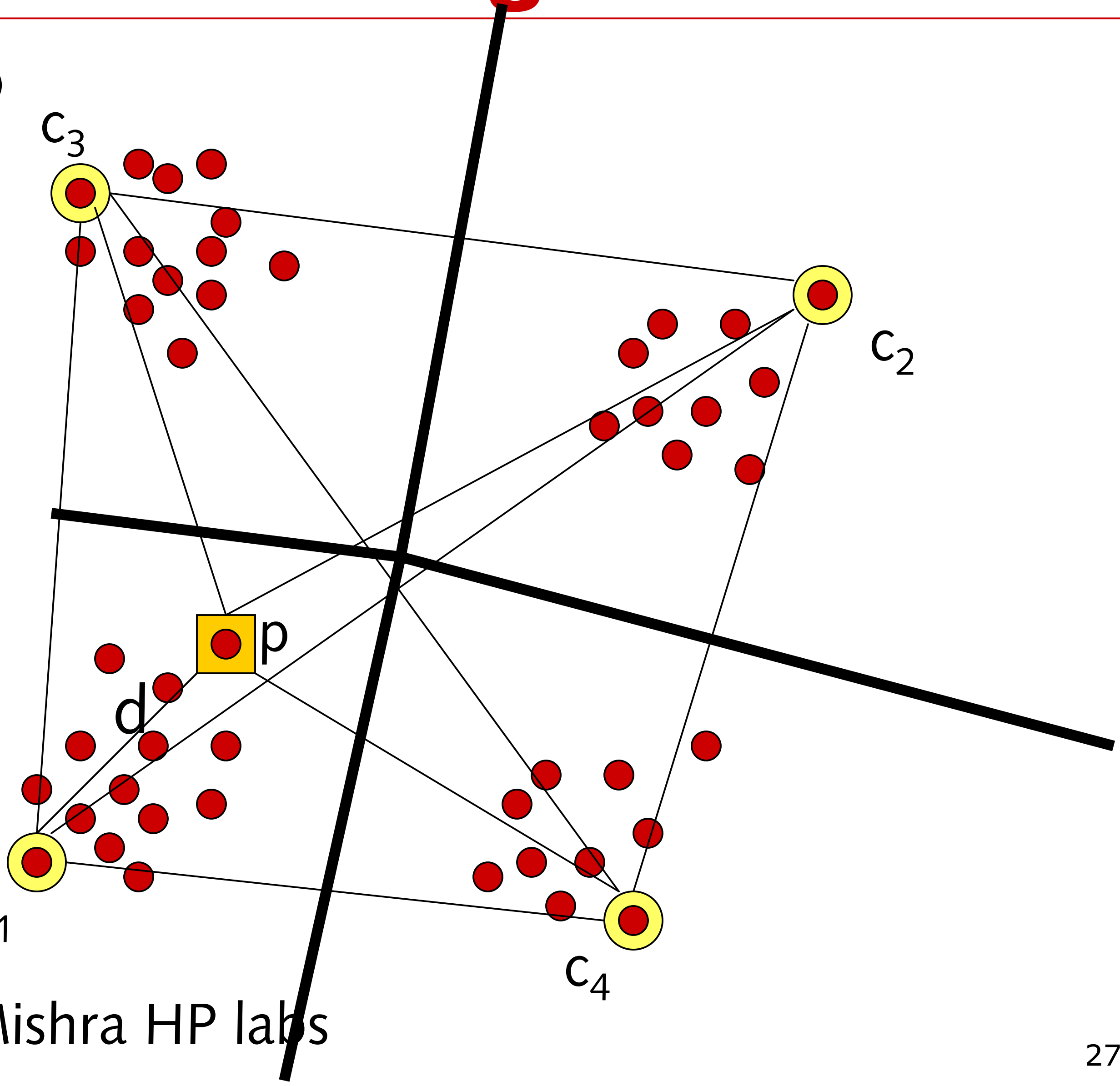
Gonzalez Clustering k=4

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.
- dist(c_i,p) $\geq$ d   for all i
- 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 $\leq$ 2OPT



Slide due to Nina Mishra HP labs

Gonzalez is 2-approximation

After picking k points to be centers, find next point that would be chosen. Let distance from 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 same set, so any k-clustering must have diameter d_{opt}

For any two points allocated to the same center, they are both at distance at most d_{opt} from their closest center

Their distance is at most $2d_{opt}$, using triangle inequality.

Diameter of any clustering must be at least d_{opt}, and is at most $2d_{opt}$ – so we have a 2 approximation.

Lower bound: NP-hard to guarantee better than 2

Available Clustering Software

- SPSS implements k-means, hierarchical and “two-step” clustering (groups items into pre-clusters, then clusters these)
- XLMiner (Excel plug-in) does k-means and hierarchical
- Clustan *ClustanGraphics offers 11 methods of hierarchical cluster analysis, plus k-means analysis, FocalPoint clustering.* Up to 120K items for average linkage, 10K items for other hierarchical methods.
- Mathematica – hierarchical clustering
- Matlab – plug-ins for k-means, hierarchical and EM based on mixture of Gaussians, fuzzy c-means

(Surprisingly?) not much variety…

Clustering Summary

There are a zillion other clustering algorithms:

- Lots of variations of EM, k-means, hierarchical
- Many “theoretical” algorithms which focus on getting good approximations to objective functions
- “Database” algorithms: BIRCH, CLARANS, DB-SCAN, CURE focus on good results and optimizing resources
- Plenty of other ad-hoc methods out there

All focus on the clustering part of the problem (clean input, model specified, clear objective)

Don’t forget the data (collection, cleaning, modeling, choosing distance, interpretation…)

2. Streaming Analysis

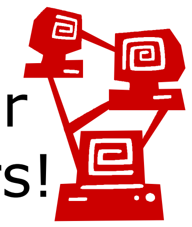
Outline

- Cluster Analysis
  - Clustering Issues
  - Clustering algorithms:
    Hierarchical Agglomerative Clustering, K-means, Expectation Maximization, Gonzalez approximation for K-center
- Data Stream Analysis
  - Massive Data Scenarios
  - Distance Estimates for High Dimensional Data:
    Count-Min Sketch for L_∞, AMS sketch for L_2, Stable sketches for L_p, Experiments on tabular data
  - Too many data points to store:
    Doubling Algorithm for k-center clustering, Hierarchical Algorithm for k-median, Gridding algorithm for k-median
- Conclusion and Summary

Data is Massive

Data is growing faster than our ability to store or process it

- There are 3 Billion Telephone Calls in US each day, 30 Billion emails daily, 1 Billion SMS, IMs.
- Scientific data: NASA's observation satellites generate billions of readings each per day.
- IP Network Traffic: up to 1 Billion packets per hour per router. Each ISP has many (hundreds) routers!
- Whole genome sequences for many species now available: each megabytes to gigabytes in size



Massive Data Analysis

Must analyze this massive data:

- Scientific research (compare viruses, species ancestry)
- System management (spot faults, drops, failures)
- Customer research (association rules, new offers)
- For revenue protection (phone fraud, service abuse)

Else, why even measure this data?

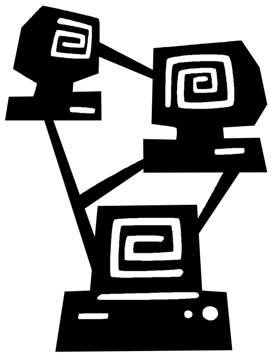
Example: Network Data

Networks are sources of massive data: the metadata per hour per router is gigabytes

Fundamental problem of data stream analysis: Too much information to store or transmit

So process data as it arrives: one pass, small space: the *data stream* approach.

Approximate answers to many questions are OK, if there are guarantees of result quality



Streaming Data Questions

Network managers ask questions that often map onto "simple" functions of the data.

- Find hosts with similar usage patterns (cluster)?
- Destinations using most bandwidth?
- Address with biggest change in traffic overnight?

The complexity comes from limited space and time.

Here, we will focus on clustering questions, which will demonstrate many techniques from streaming

Streaming And Clustering

Relate back to clustering: how to scale when data is massive?

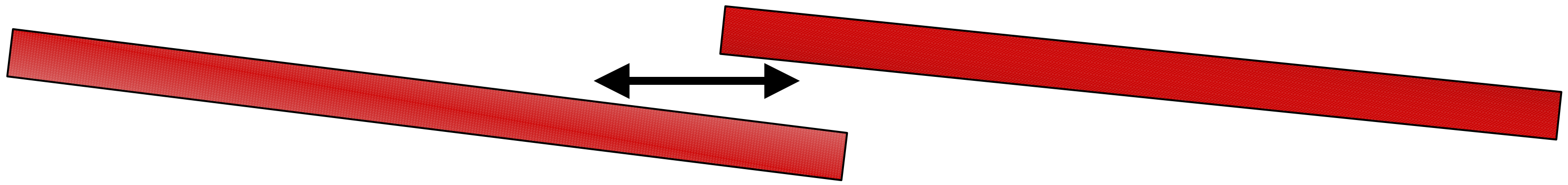
- Have already seen $O(n^4)$, $O(n^3)$, even $O(n^2)$ algorithms don’t scale with large data
- Need algorithms that are fast, look at data only once, cope smoothly with massive data

Two (almost) orthogonal problems:

- How to cope when number of points is large?
- How to cope when each point is large?

Focusing on these shows more general streaming ideas.

When each point is large…



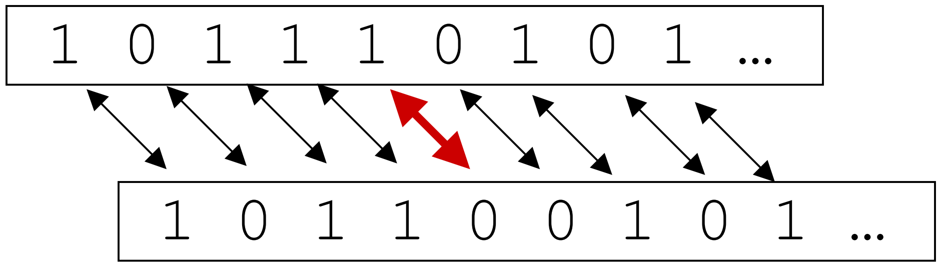
For clustering, need to compare the points. What happens when the points are very high dimensional?

- Eg. trying to compare whole genome sequences
- comparing yesterday's network traffic with today's
- clustering huge texts based on similarity

If each point is size m, m very large $\Rightarrow$ cost is very high (at least $O(m)$. $O(m^2)$ or worse for some metrics)

Can we do better? Intuition says no… randomization says yes!

Trivial Example



Simple example. Consider "equality distance":

$d_=(x,y) = 0$ iff $x=y$, 1 otherwise

To compute equality distance perfectly, must take linear effort: check every bit of x = every bit of y.

Can speed up with pre-computation and randomization: use a hash function h on x and y, test $h(x)=h(y)$

Small chance of false positive, no chance of false negative.

When x and y are seen in streaming fashion, compute $h(x)$, $h(y)$ incrementally as new bits arrive (Karp-Rabin)

Other distances

Distances we care about:

- Euclidean (L_p) distance— $\| x- y \|_2 = (\sum_i (x_i - y_i)^2)^{1/2}$
- Manhattan (L_1) distance— $\| x- y \|_1 = \sum_i |x_i - y_i|$
- Minkowski (L_p) distances— $\| x- y \|_p = (\sum_i |x_i - y_i|^p)^{1/p}$
- Maximum (L_∞) distance— $\| x-y \|_\infty = \max_i |x_i - y_i|$
- Edit distances: $d(x,y)$ = smallest number of insert/delete operations taking string x to string y
- Block edit distances: $d(x,y)$ = smallest number of indels & block moves taking string x to string y

For each distance, can we have functions h and f so that $f(h(x),h(y)) \approx d(x,y)$, and $|h(x)| \ll |x|$?

L_∞ distance

We will consider L_∞ distance.

Example: $\| [2,3,5,1] - [4,1,6,2] \|_\infty = \| [2,2,1,1] \|_\infty = 2$

Provably hard to approximate with relative error,
so will show an approximation with error $\pm \varepsilon \| x - y \|_1$

First, consider subproblem: estimate a value in a vector

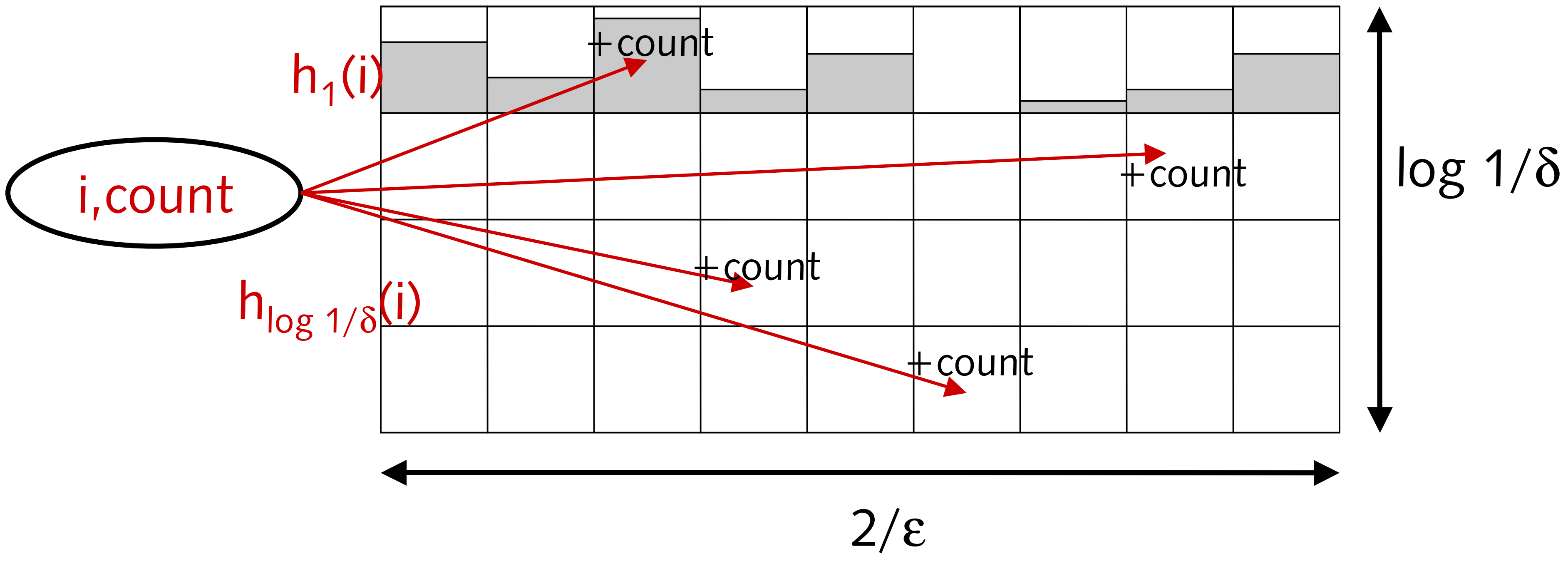
Stream defines a vector a[1..U], initially all 0
Each update change one entry, $a[i] \leftarrow a[i] + \text{count}$.
In networks $U = 2^{32}$ or 2^{64}, too big to store

Can we use less space but estimate each a[i]
reasonably accurately?

Update Algorithm

Ingredients:

- Universal hash fns $h_1..h_{\log 1/\delta}$ $\{1..U\} \rightarrow \{1..2/\varepsilon\}$
- Array of counters $CM[1..2/\varepsilon, 1..\log_2 1/\delta]$



Count-Min Sketch

Approximation

Approximate $\hat{a}[i] = \min_j CM[h_j(i),j]$

Analysis: In j'th row, $CM[h_j(i),j] = a[i] + X_{i,j}$

$X_{i,j} = \Sigma\ a[k] \mid h_j(i) = h_j(k)$

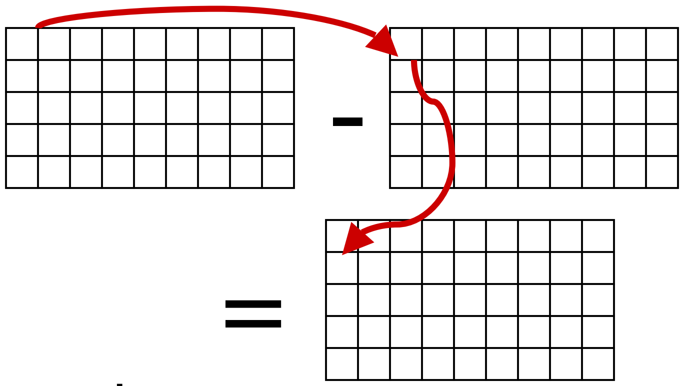
$E(X_{i,j}) = \Sigma\ a[k]*Pr[h_j(i)=h_j(k)]$
$\leq Pr[h_j(i)=h_j(k)] * \Sigma\ a[k]$
$= \varepsilon N/2$ by pairwise independence of h

$Pr[X_{i,j} \geq \varepsilon N] = Pr[X_{i,j} \geq 2E(X_{i,j})] \leq 1/2$ by Markov inequality

So, $Pr[\hat{a}[i] \geq a[i] + \varepsilon\ \|a\|_1] = Pr[\forall\ j.\ X_{i,j} > \varepsilon\ \|a\|_1] \leq 1/2^{\log 1/\delta} = \delta$

Final result:
with certainty $a[i] \leq \hat{a}[i]$ and
with probability at least $1-\delta$, $\hat{a}[i] < a[i] + \varepsilon\ \|a\|_1$

Applying to L_∞



- By linearity of sketches, we have $CM(x - y) = CM(x) - CM(y)$
- Subtract corresponding entries of the sketch to get a new sketch.
- Can now estimate $(x - y)[i]$ using sketch
- Simple algorithm for L_∞: estimate $(x-y)[i]$ for each i, take max. But too slow!
- Better: can use a group testing approach to find all i's with $(x-y)[i] > \varepsilon \| x - y \|_1$, take max to find L_∞ ❑
- Note: group testing algorithm originally proposed to find large changes in network traffic patterns.

L_2 distance

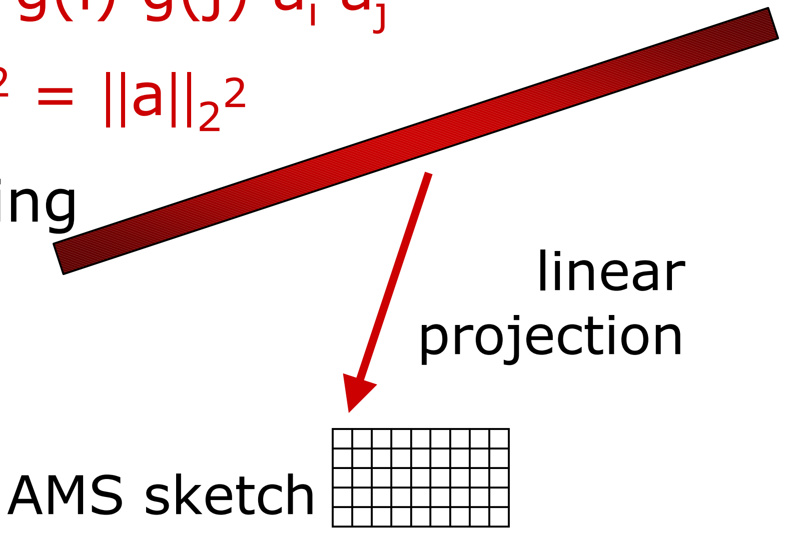
Describe a variation of the Alon-Matias-Szegedy algorithm for estimating L_2 by generalizing CM sketch.

Use extra hash functions $g_1..g_{\log 1/\delta}$ $\{1..U\} \rightarrow \{+1,-1\}$

Now, given update (i,u), set $CM[h(i),j] \mathrel{+}= u * g_j(i)$

Estimate $\|a\|_2^2 = \text{median}_j \sum_i CM[i,j]^2$

- Result is $\sum_i g(i)^2 a_i^2 + \sum_{h(i)=h(j)} 2\, g(i)\, g(j)\, a_i\, a_j$
- $g(i)^2 = -1^2 = +1^2 = 1$, and $\sum_i a_i^2 = \|a\|_2^2$
- $g(i)g(j)$ has 50/50 chance of being +1 or –1 : in expectation is 0 …



L_2 accuracy

Formally, one can show that the expectation of each estimate is exactly $||a||_2^2$ and variance is bounded by ε^2 times expectation squared.

Using Chebyshev's inequality, show that probability that each estimate is within $\pm\ \varepsilon\ ||a||_2^2$ is constant

Take median of $\log (1/\delta)$ estimates reduces probability of failure to δ (using Chernoff bounds)

Result: given sketches of size $O(1/\varepsilon^2 \log 1/\delta)$ can estimate $||a||_2^2$ so that result is in $(1\pm\varepsilon)||a||_2^2$ with probability at least $1-\delta$ ❑

[Note: same Chebyshev-Chernoff argument used many time in data stream analysis]

Sketches for L_p distance

Let X be a random variable distributed with a *stable distribution.* Stable distributions have the property that

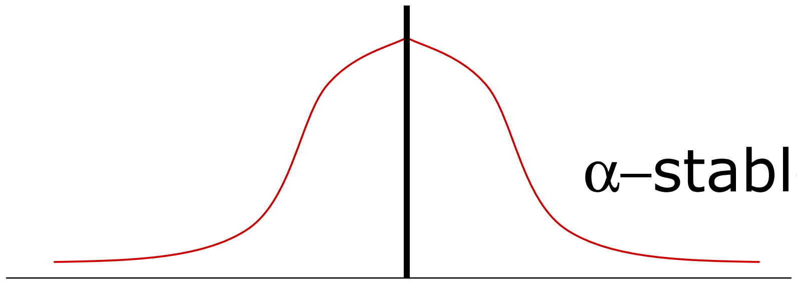
$$a_1X_1 + a_2X_2 + a_3X_3 + \ldots a_nX_n \sim \|(a_1, a_2, a_3, \ldots, a_n)\|_p X$$

if $X_1 \ldots X_n$ are stable with stability parameter p

The Gaussian distribution is stable with parameter 2

Stable distributions exist and can be simulated for all parameters $0 < p < 2$.

So, let $\mathbf{x} = x_{1,1} \ldots x_{m,n}$ be a matrix of values drawn from a stable distribution with parameter p...



α–stable distribution

Creating Sketches

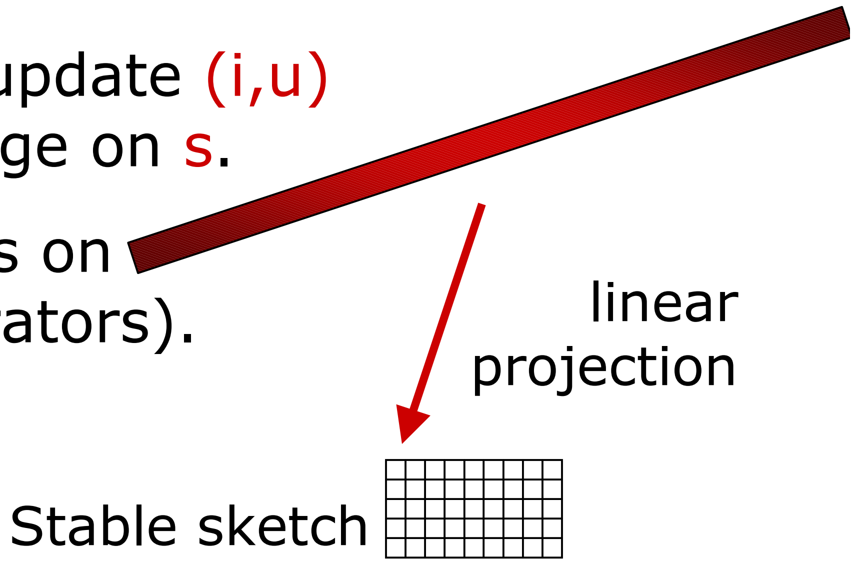
Compute $s_i = \mathbf{x_i} \cdot a$, $t_i = \mathbf{x}_i \cdot b$

$\text{median}(|s_1 - t_1|, |s_2 - t_2|, \ldots, |s_m - t_m|)/\text{median}(X)$
is an estimator for $\| a - b \|_p$

Can guarantee the accuracy of this process: will be within a factor of $1+\varepsilon$ with probability δ if
$m = O(1/\varepsilon^2 \log 1/\delta)$

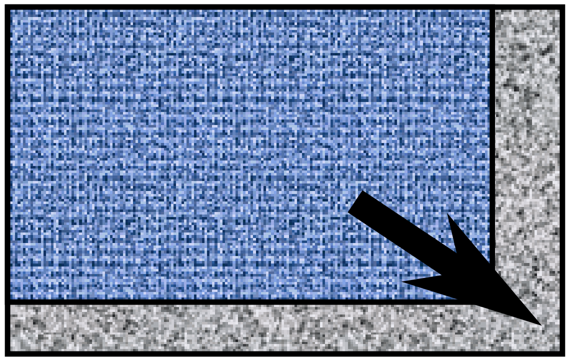
Streaming computation: when update (i,u) arrives, compute resulting change on s.

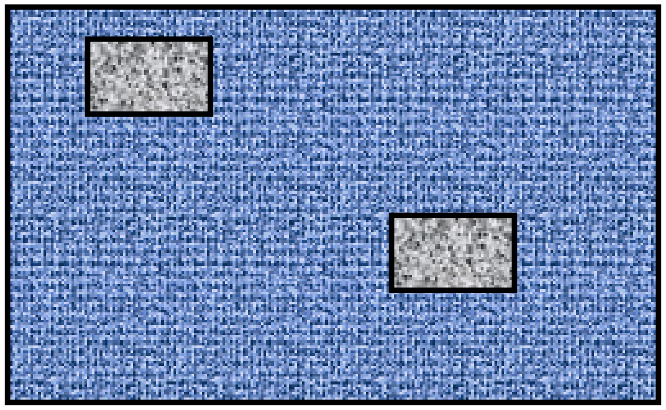
Don't store $\mathbf{x}$ -- compute entries on demand (pseudo-random generators).



Experiments with tabular data

Adding extra rows or columns increases the size by thousands or millions of readings





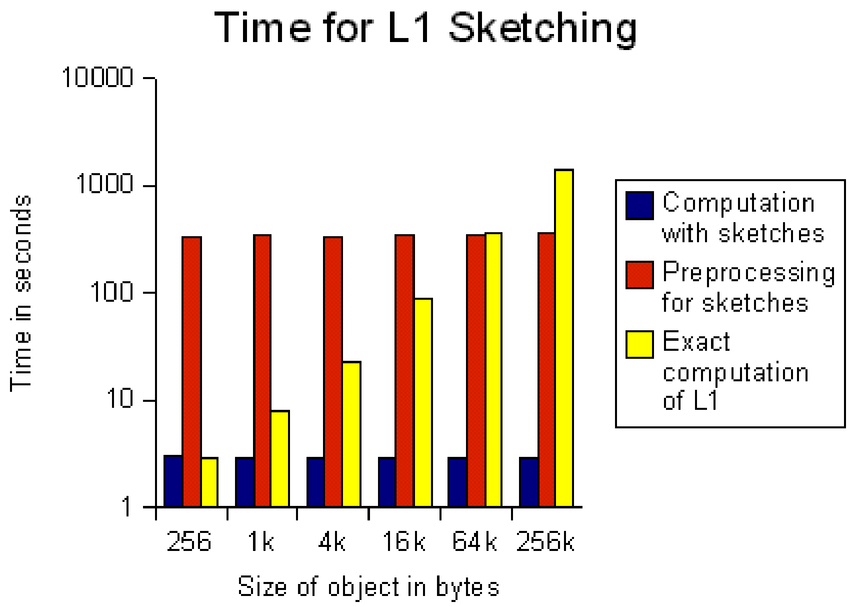
The objects of interest are *subtables* of the data

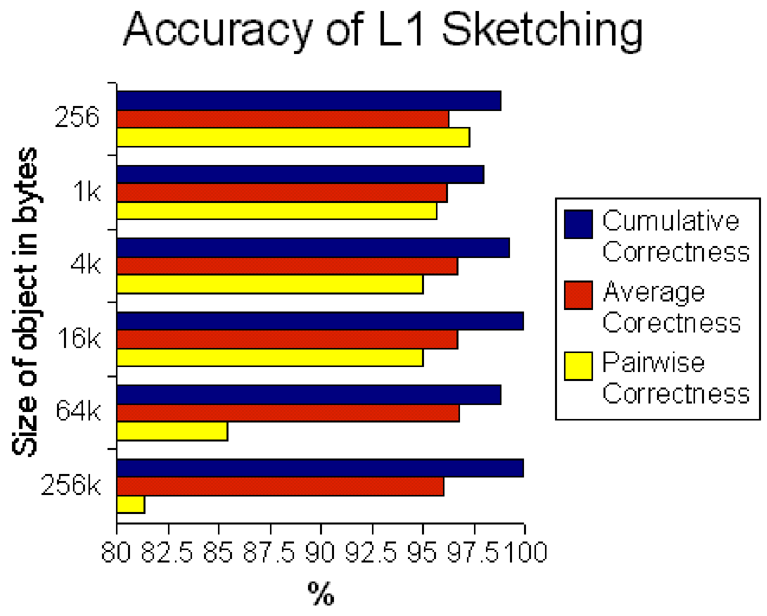
eg Compare cellphone traffic of SF with LA

These subtables are also massive!

L_1 Tests

We took 20,000 pair of subtables, and compared them using L_1 sketches. The sketch size was less than 1Kb.





- Sketches are very fast and accurate

(can be improved further by increasing sketch size)

- For large enough subtables (>64KB) the time saving

"buys back" pre-processing cost of sketch computation

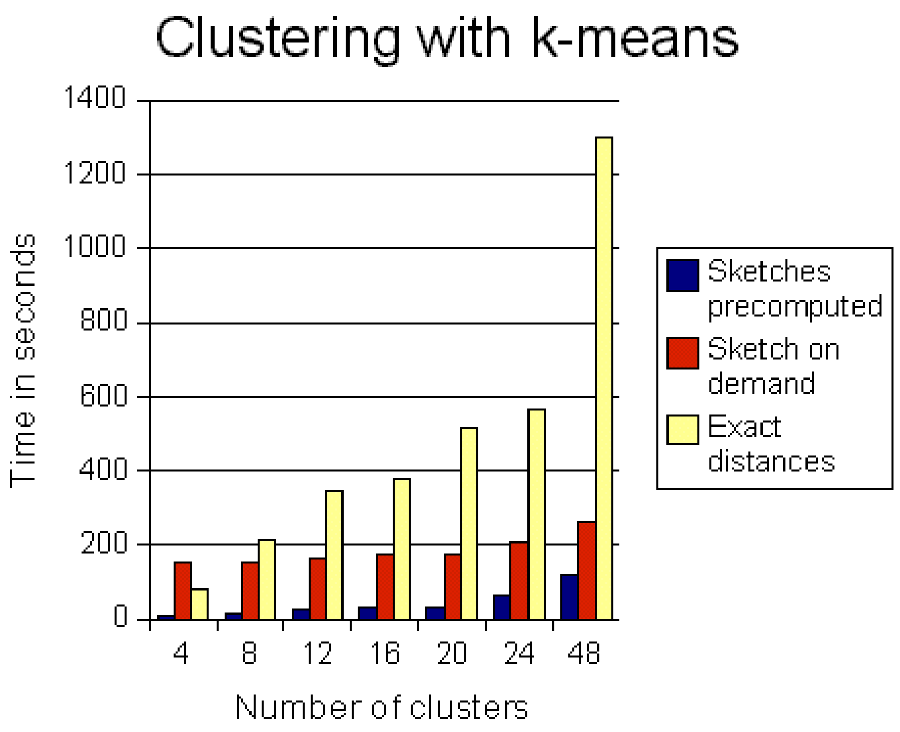
Clustering with k-means

Run k-means algorithm, replacing all distance computations with sketch computations

Sketches are much faster than exact methods, an creating sketches when needed is always faster than exact computation

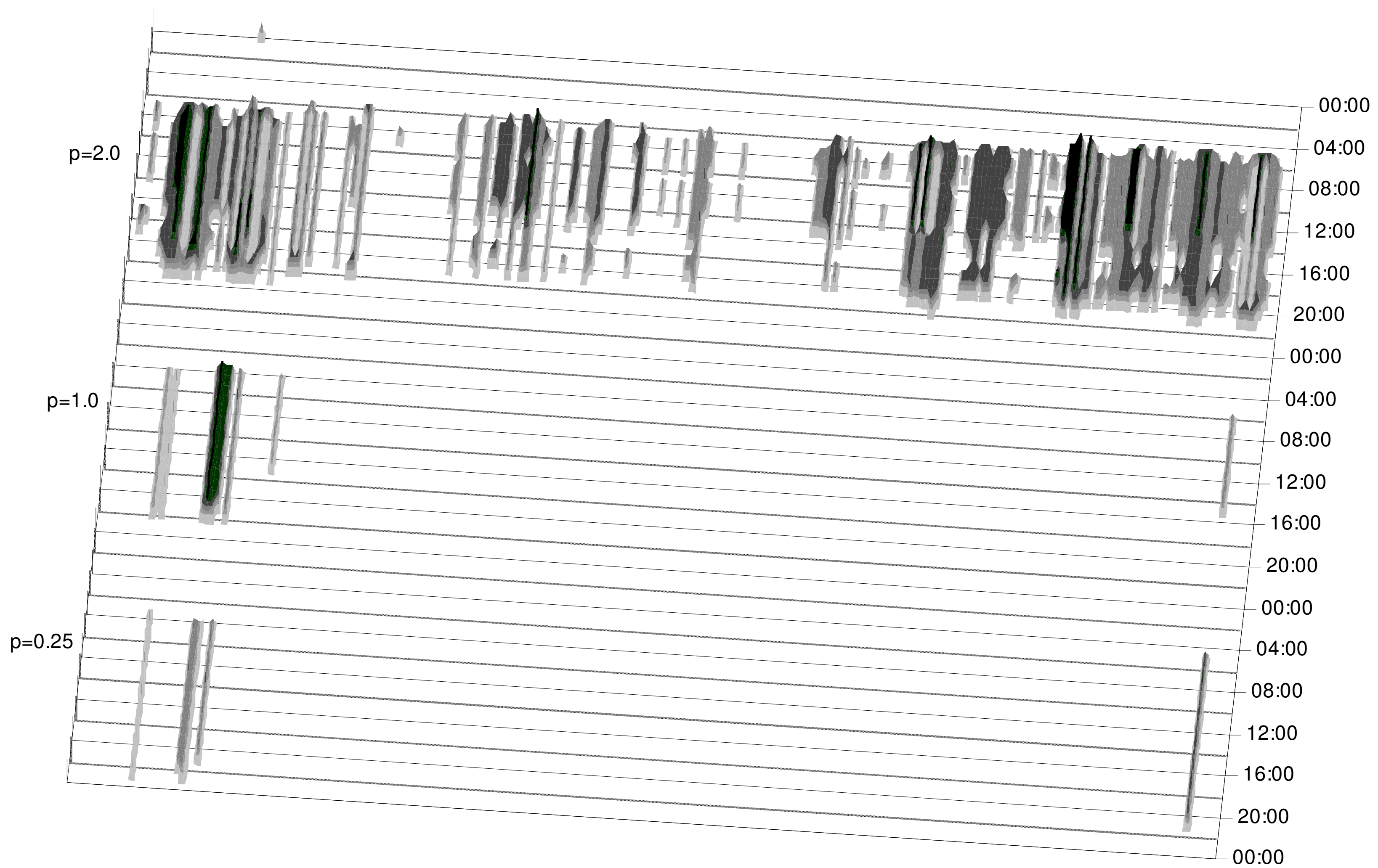
As k increases, the time saving becomes more significant.

For 8 or more clusters, creating sketches when needed is much faster.



Case study: US Call Data

One day's data clustered under p=2.0, p=1.0, p=0.25



Case study: US Call Data

We looked at the call data for the whole US for a single day

- $p = 2$ shows peak activity across the country from 8am - 5pm local time, and activity continues in similar patterns till midnight
- $p = 1$ shows key areas have similar call patterns throughout the day
- $p = 0.25$ brings out a very few locations that have highly similar calling patterns

Streaming Distance Summary

When each input data item is huge, can approximate distances using small sketches of the data

Sketches can be computed as the data streams in…

Higher level algorithms (eg, nearest neighbors, clustering) can run, replacing exact distances with approximate (sketch) distances.

Different distances require different sketches
– have covered $d_=$, L_∞, L_2 and L_p $(0<p<2)$

Partial results known for other distances, eg. edit distance/block edit distance, earth movers distance etc.

Outline

- Cluster Analysis
  - Clustering Issues
  - Clustering algorithms:
    Hierarchical Agglomerative Clustering, K-means, Expectation Maximization, Gonzalez approximation for K-center
- Data Stream Analysis
  - Massive Data Scenarios
  - Distance Estimates for High Dimensional Data:
    Count-Min Sketch for L_∞, AMS sketch for L_2, Stable sketches for L_p, Experiments on tabular data
  - Too many data points to store:
    Doubling Algorithm for k-center clustering, Hierarchical Algorithm for k-median, Gridding algorithm for k-median
- Conclusion and Summary

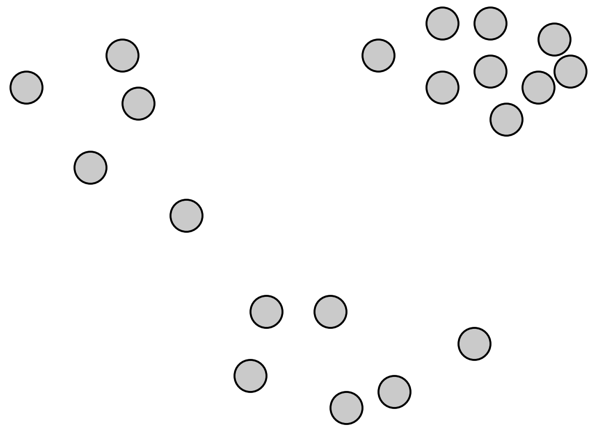
Stream Clustering Many Points

What does it mean to cluster on the stream when there are too many points to store?

We see a sequence of points one after the other, and we want to output a clustering for this observed data.

Moreover, since this clustering changes with time, for each update we maintain some summary information, and at any time can output a clustering.

Data stream restriction: data is assumed too large to store, so we do not keep all the input, or any constant fraction of it.



Clustering for the stream

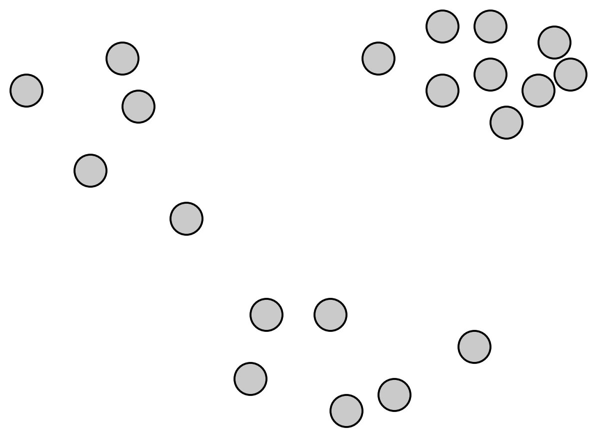
What should output of a stream clustering algorithm be?

Classification of every input point? Too large to be useful? Might this change as more input points arrive?

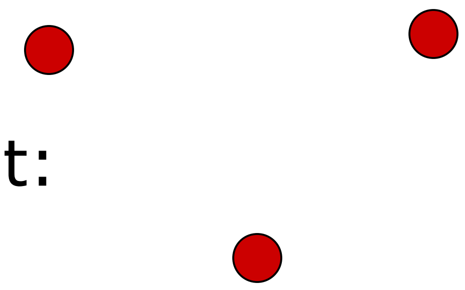
- Two points which are initially put in different clusters might end up in the same one

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

Input:



Output:



Gonzalez Restated

Suppose we knew d_{opt} (from Gonzalez algorithm for k-centers) at the start

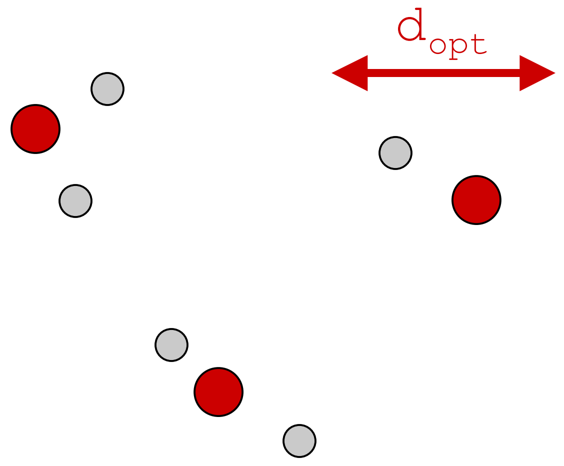
Do the following procedure:

```
Select the first point as the first center

For each point that arrives:

  – Compute d_min, the distance to the closest center

  – If d_min > d_opt then set the new point to be a new
    center
```



Analysis Restated

d_{opt} is given, so we know that there are $k+1$ points separated by $\geq d_{opt}$ and d_{opt} is as large as possible

So there are $\leq k$ points separated by $> d_{opt}$

New algorithm outputs at most k centers: only include a center when its distance is $> d_{opt}$ from all others. If $> k$ centers output, then $> k$ points separated by $> d_{opt}$, contradicting optimality of d_{opt}.

Every point not chosen as a center is $< d_{opt}$ from some center and so at most $2d_{opt}$ from any point allocated to the same center (triangle inequality)

So: given d_{opt} we find a clustering where every point is at most twice this distance from its closest center

Guessing the optimal solution

Hence, a 2-approximation -- but, we aren't given d_{opt}

Suppose we knew 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 $d_{opt} = 1, 2, 4, 8...$

We reject everything less than d_{opt}, so best guess is $< 2d_{opt}$: our output will be $< 2*2d_{opt}/d_{opt} = 4$ approx

Need $\log_2 (d_{max}/d_{smallest})$ guesses, $d_{smallest}$ is minimum distance between any pair of points, as $d_{smallest} < d_{opt}$

$O(k \log(d_{max} / d_{smallest})$ may be high, can we reduce more?

Doubling Algorithm

Doubling alg [Charikar et al 97] uses only $O(k)$ space. Each 'phase' begins with $k+1$ centers, these are merged to get fewer centers.

Initially set first $k+1$ points in stream as centers.

Merging: Given $k+1$ centers each at distance at least d_i, pick one arbitrarily, discard all centers within $2d_i$ of this center; repeat until all centers separated by at least $2d_i$

Set $d_{i+1} = 2d_i$ and go to phase $i+1$

Updating: While $< k+1$ centers, for each new point compute $d_{min.}$ If $d_{min} > d_i$, then set the new point to be a new center

Analyzing merging centers

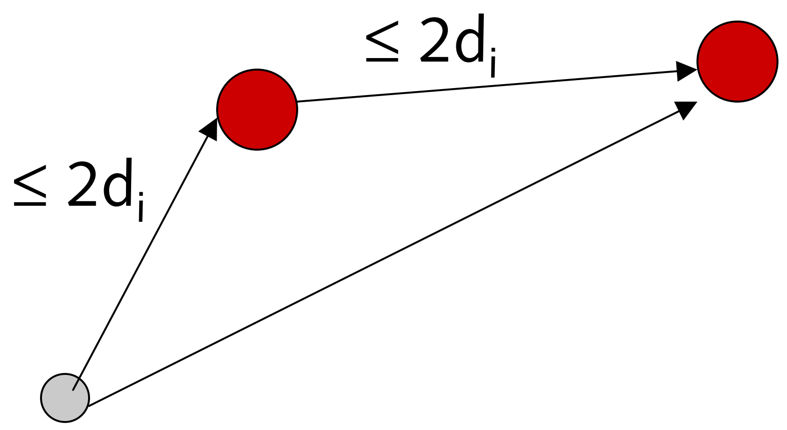
After merging, every pair of centers is separated by at least d_{i+1}

Claim: Every point that has been processed is at most $2d_{i+1}$ from its closest center

Proof by induction

Base case:
The first k+1 (distinct) points are chosen as centers
Set d_0 = minimum distance between any pair
Every point is distance 0 from its closest center
And trivially, $0 \leq 2d_0$

Finishing the Induction



Every point is at most $2d_{i+1}$ from its closest center

Inductive case: before merging, every point that has been seen 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 $= 2d_i + 2d_i = 4di = 2d_{i+1}$

Optimality Ratio

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

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

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

So a factor 8 approximation

Total time is (amortized) $O(n\ k \log k)$ using heaps ❑

K-medians

k-medians measures the quality based on the average distance between points and their closest median. So: $\Sigma_{p1}\ d(p_1,\text{median}(p_1))/n$

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

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

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

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 $C \le 2C^*$ using the previous result.

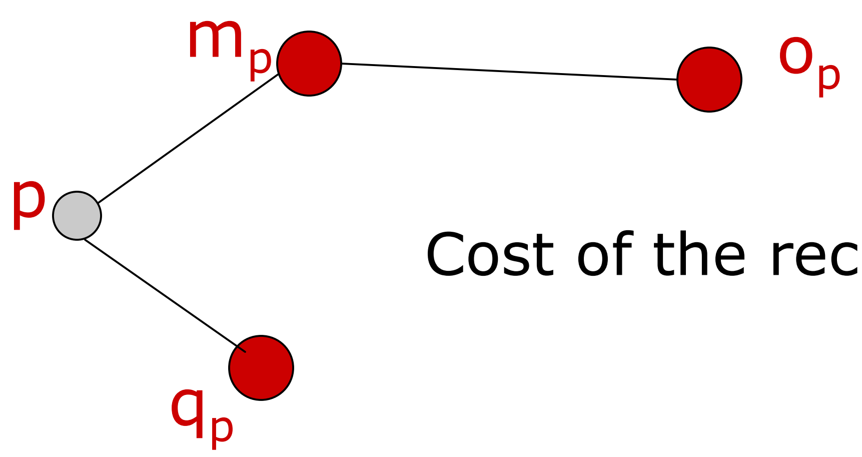
How to recluster

After processing all groups, $n^{1/2}$ sets of k-medians.

For each median, use "weight": number of points were allocated to it. Recluster using the weighted medians.

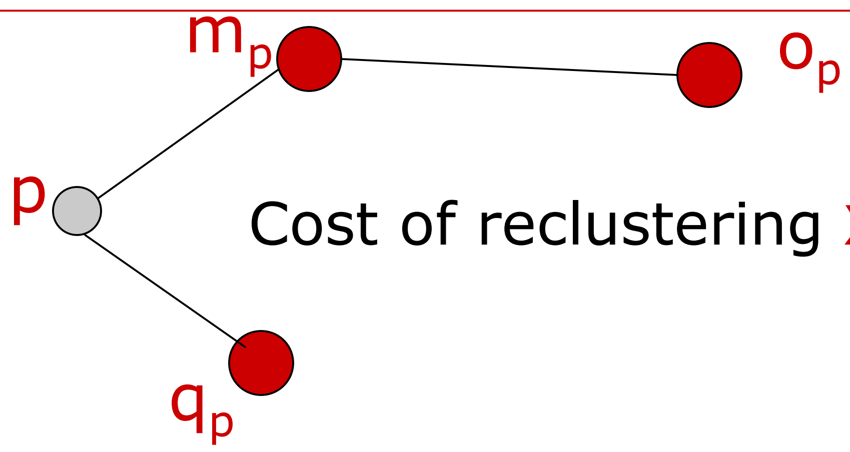
Each point p is allocated to some median m_p, which is then reclustered to some new median o_p.

Let the optimal k-median for point p be q_p



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

Cost of reclustering



Cost of reclustering $\Sigma_p\, d(m_p,o_p) \leq \Sigma_p\, d(m_p,q_p)$

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

$$\Sigma_p\, d(m_p, q_p) \leq \Sigma_p\, d(m_p, p) + d(p,q_p)$$

$$= \text{cost(1st clustering)} + \text{cost(optimal clustering)}$$
$$= C + C^*$$

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

Previous analysis assumes optimal k-median clustering. Too expensive; in practice, find c-approximation.

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

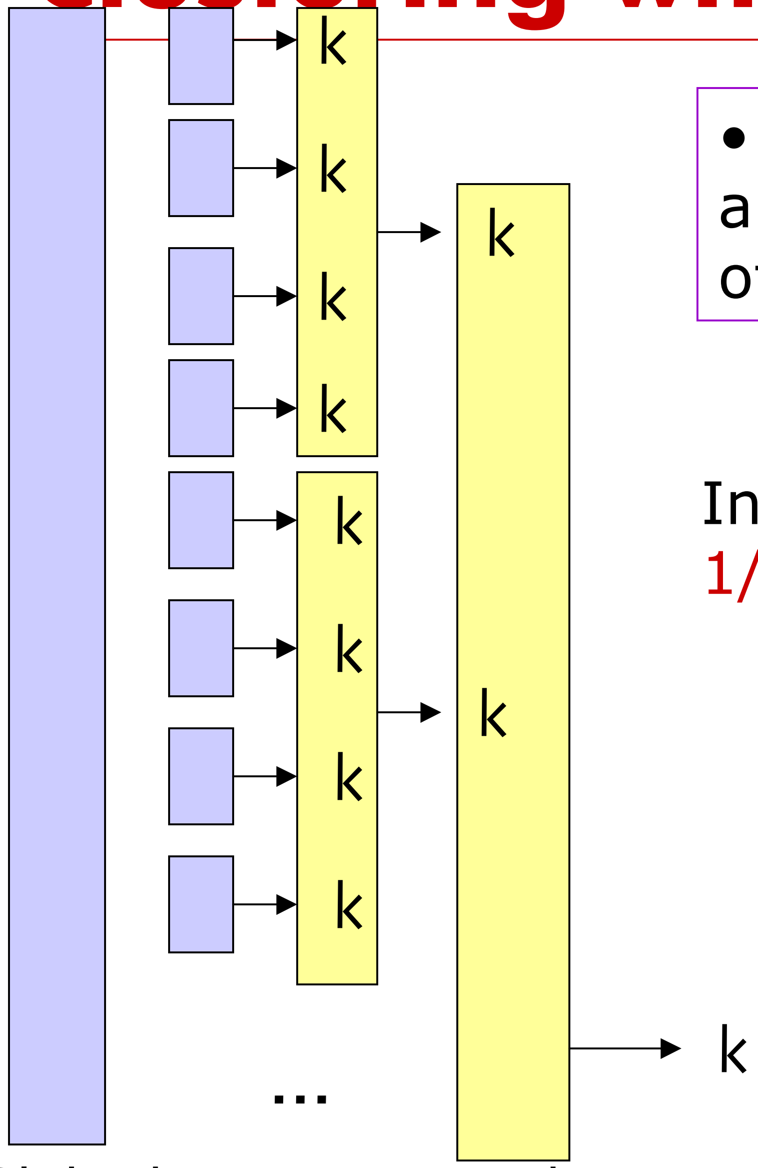
Putting this together gives a bound of

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

This uses $O(kn^{1/2})$ space, which is still a lot. Use this procedure to repeatedly merge clusterings.

Approximation factor gets worse with more levels (one level: $O(c)$, two: $O(c^2)$, i: $O(c^i)$)

Clustering with small Memory



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

In general, if |Memory|=n^{ε}, need $1/\varepsilon$ levels, approx factor $2^{O(1/\varepsilon)}$

- 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} ❑

Slide due to Nina Mishra

Gridding Approach

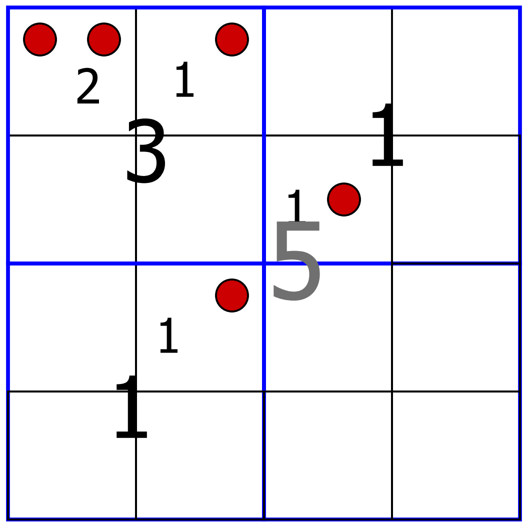
Other recent approaches use "Gridding":

Divide space into a grid, keep count of number of points in each cell.

Repeat for successively coarser grids.

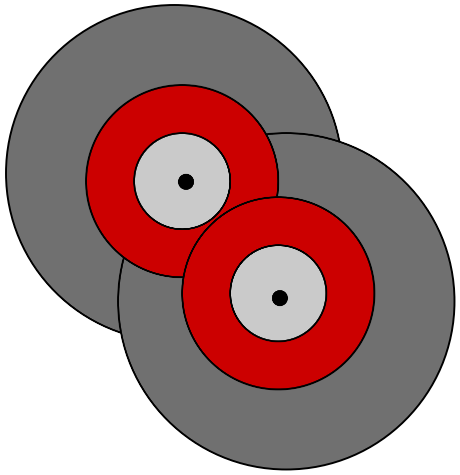
Show that by tracking information on grids can approximate clustering: (1+ε) approx for k-median in low dimensions [Indyk 04, Frahling Sohler 05]

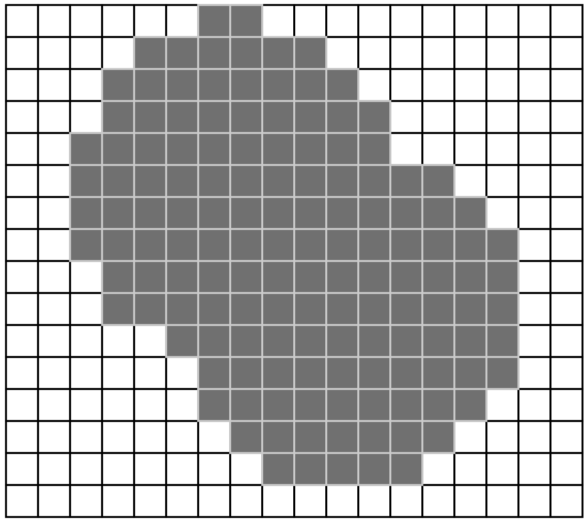
Don't store grids exactly, but use sketches to represent them (allows deletion of points as well as insertions).



Using a grid

Given a grid, can estimate the cost of a given clustering:





Cost of clustering
$\approx \sum_r$ number of points not covered by circle of radius r
$\approx \sum_r$ points not covered in grid by coarse circle

Now can search for best clustering (still quite costly) ❑

Summary of results

Have seen many of the key ideas from data streaming:

- Create small summaries that are *linear projections* of the input: ease of composability [all sketches]
- Use hash functions and randomized analysis (with limited independence properties) [L_2 sketches]
- Use probabilistic random generators to compute same "random" number many times [L_p sketches]
- Combinatorial or geometric arguments to show that easily maintained data is good approx [Doubling alg]
- Hierarchical or tree structure approach: compose summaries, summarize summaries [k-median algs]

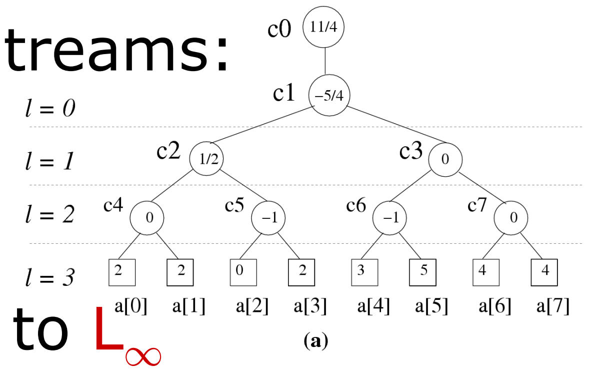
Approximates expensive computations more cheaply

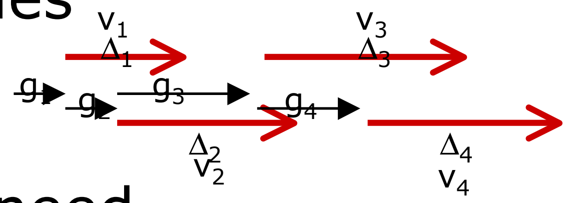
Related topics in Data Streams

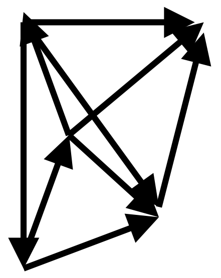
Related data mining questions from Data Streams:

- Heavy hitters, frequent items, wavelet, histograms – related to L_∞.
- Median, quantile computation – connects to L_∞
- Change detection, trend analysis – sketches
- Distinct items, F_0 – can use L_p sketches
- Decision trees, other mining primitives – need approx representations of the input to test

Have tried to show some of the key ideas from streaming, as they apply to clustering.







Streaming Conclusions

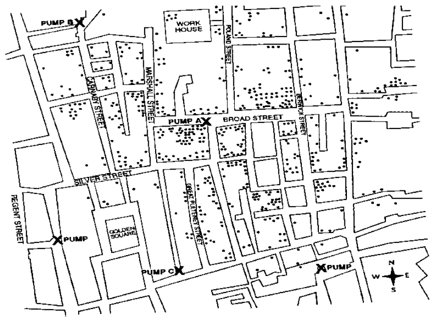
A lot of important data mining and database questions can be solved on the data stream

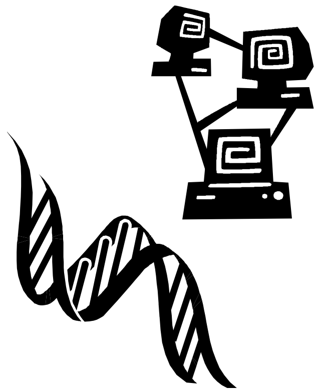
Exact answers are unlikely: instead we apply approximation and randomization to keep memory requirements low

Need tools from algorithms, statistics & database to design and analyze these methods.

Problem to ponder: what happens when each point is too high dimensional **and** too many points to store?

Closing Thoughts

From 

to… 

Clustering a hugely popular topic, but needs care.

Doesn't always scale well, need careful choice of algorithms or approximation methods to deal with huge data sets.

Sanity check: does the resultant clustering make sense?

What will you do with the clustering when you have it? Use as a tool for hypothesis generation, leading into more questions?

(A few) (biased) References

N. Alon, Y. Matias, M. Szegedy, “The Space Complexity of Approximating the Frequency Moments”, STOC 1996

N. Alon, P. Gibbons, Y. Matias, M. Szegedy, “Tracking Join and Self-Join Sizes in Limited Space”, PODS 1999

M. Charikar, C. Chekuri, T. Feder, R.Motwani, “Incremental clustering and dynamic information retrieval”, STOC 1997

G. Cormode “Some key concepts in Data Mining: Clustering” in *Discrete Methods in Epidemiology,* AMS*,* 2006

G. Cormode and S. Muthukrishnan, “An Improved Data Stream Summary: The count-min sketch and its applications” J. Algorithms, 2005;

G. Cormode and S. Muthukrishnan, “What’s new: finding significant differences in Network Data Streams” Transactions on Networking, 2005

G. Cormode, P. Indyk, N. Koudas, S. Muthukrishnan “Fast Mining of Tabular Data via Approximate Distance Computations”, ICDE 2002.

G. Frahling and C. Sohler, “Coresets in Dynamic Geometric Streams”, STOC 2005

T. Gonzalez, “Clustering to minimize the maximum intercluster distance”, Theoretical Computer Science, 1985

S. Guha, N. Mishra, R. Motwani, O’Callaghan, “Clustering Data Streams” FOCS 2000

P. Indyk “Algorithms for dynamic geometric problems over data streams”, STOC 2004

S. Muthukrishnan, “Data Streams: Algorithms and Applications”, SODA 2002