Introduction to Data Science

GIRI NARASIMHAN, SCIS, FIU

Streams & Bloom Filters

Moments

- i-th Momemt
- Zeroth Moment: Count of distinct elements in stream
- First Moment: Count of elements in stream, i.e., Size of stream; sum of freq
- Second Moment: Sum of squares of frequencies
- Average = ?
- Variance = ?

$$\frac{1}{m}\sum_{s=1}^{m}\left(f_s - \frac{n}{m}\right)^2 = \frac{1}{m}\sum_{s=1}^{m}\left(f_s^2 - 2\frac{n}{m}f_s + \left(\frac{n}{m}\right)^2\right) = \left(\frac{1}{m}\sum_{s=1}^{m} f_s^2\right) - \frac{n^2}{m^2}$$

Sampling Woes

- **Stream**: Tuples (**user**, **query**, **time**); **Sampling**: 1 in 10
  - Each user has 1/10 of their queries processed
- **Query**: Fraction of typical user's queries repeated over last month
- **Correct Answer**: Suppose user has s unique queries and d queries twice and NO queries more than twice in the last month; Answer = d/(s+d)
- **Problem**: Reported fraction would be wrong
  - In the sampled stream, s/10 are unique queries and d/100 queries appear twice
  - The remainder of the queries that should appear twice will appear once 18d/100
  - We will report d/(10s + 19d) [d/100 twice and s/10 + 18d/100 once

Improved Solution for Sampling Woes

- Problem is that we are picking 1/10 of the queries
- We need to pick 1/10 of the users and pick all their queries
- If we can store 1/10 of the users, then for every query we can decide either to process or not
- **Improved Solution**: Hash user ID (actually, IP address) to 0 … 9
  - Pick only those that hash to 0
- **Sampling Question**: How to sample at rate of 1/70?
- **Sampling Question**: How to sample at rate of 23/70?

Sampling

- Sampling can be applied if the filtering test is easy (e.g., hash value = 0? Temperature > 22 degrees?)
- Sampling is harder if it involves a lookup (e.g., has this query been asked before by this user? Is this user among the top 10% of the frequent users list?)
- Other techniques are available for filtering
  - **Bloom Filters**

Example: Bloom Filters for Spam

- **White lists**: allowed email addresses
  - Assume we have **1 Billion** allowed email addresses
  - Assume black list is much larger than white list
  - If each email address is 20 bytes, this takes 20 GB to store
- **Bloom Filters**: store white lists as bit hash arrays
  - Every email address is hashed and a 1 is stored in the location if it is in white list
  - In 1 GB, we can store hash array of size 8 Billion
- Strict White Lists: use bloom filters and then verify with real white list
- Stricter White List: use cascade of bloom filters

Bloom Filters: Test for Membership

- Array of n bits, initially all 0's
- Collection of k hash functions. Each hash func maps a key to n buckets
- Given key K, compute K hash values and
  - Check that each location in bit array is a 1
  - Even if one is 0, then it fails the test

False Positive Rate

$(1\text{-}h)^{1/h} = e$ f
small h

- Assume we have **x** targets and **y** darts
- Prob a dart will hit a specific target = $1/x$
- Prob a dart does not hit a specific target = $1 - (1/x) = (x-1)/x$
- Prob that y darts miss a specific target = $((x-1)/x)^y$
- Prob that y darts miss a specific target = $e^{-y/x}$
- Let x = 8B; y = 1B; Then prob of missing a target = $e^{-1/8}$
- Prob of hitting a target = false positive rate = $1 - e^{-1/8} = 0.1175$
- If k = 2, the prob becomes $(1 - e^{-1/4})^2 = 0.0493$

False Positive Rate

- Let n = bit array length = 8B
- Let m = # of members = 1B
- Let k = # of hash functions = 1
- Prob that a white list email hashes to a location = 10^{-9}
- False positive rate is given by
  - $(1 - e^{-km/n})^k$

Counting distinct elements

- How many unique users in a give period?
- How many users (IP addresses) visited a webpage?
  - Each IP address is 4 bytes = 32 bits
  - 4 billion IP addresses are possible = 16 GB
  - If we need this for each webpage and there are thousands, then we cannot store in memory

Flajolet-Martin Algorithm

- For each element obtain a sufficiently long hash
  - Has to be more possible results of hash than elements in the universal set
  - Example, use 64 bits ($2^{64} \sim 10^{19}$) to hash URLs (4 Billion)
  - High prob that different elements get different hash values
  - Some fraction of these hash values will be "unusual"
- We will focus on the ones that have r 0s at the end of its hash value
  - Prob of hash value to end in r 0s is 2^{-r}
  - Prob that m unique items have has values that don't end in r 0s is $(1-2^{-r})^m = e^{-m2^{-r}}$

Summary

- Look at the probability = $e^{-m2^{-r}}$
- If m is much larger than 2^r, then prob approaches 1
- If m is much smaller than 2^r, then prob approaches 0
- Thus 2^R is a good choice, where R is the largest tail of 0s

Clustering

Clustering dogs using height & weight

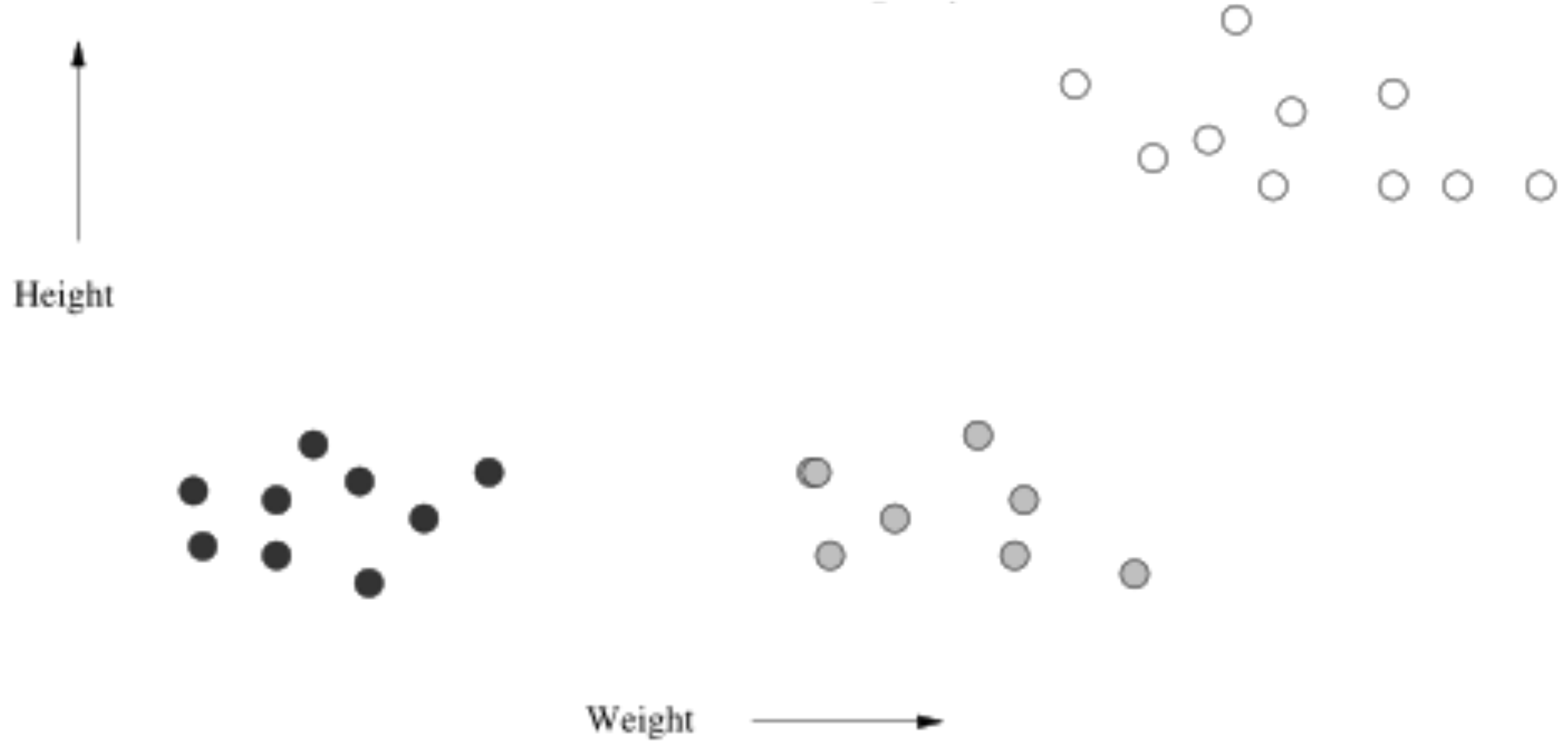


Figure 7.1: Heights and weights of dogs taken from three varieties

Clustering dogs using height & weight

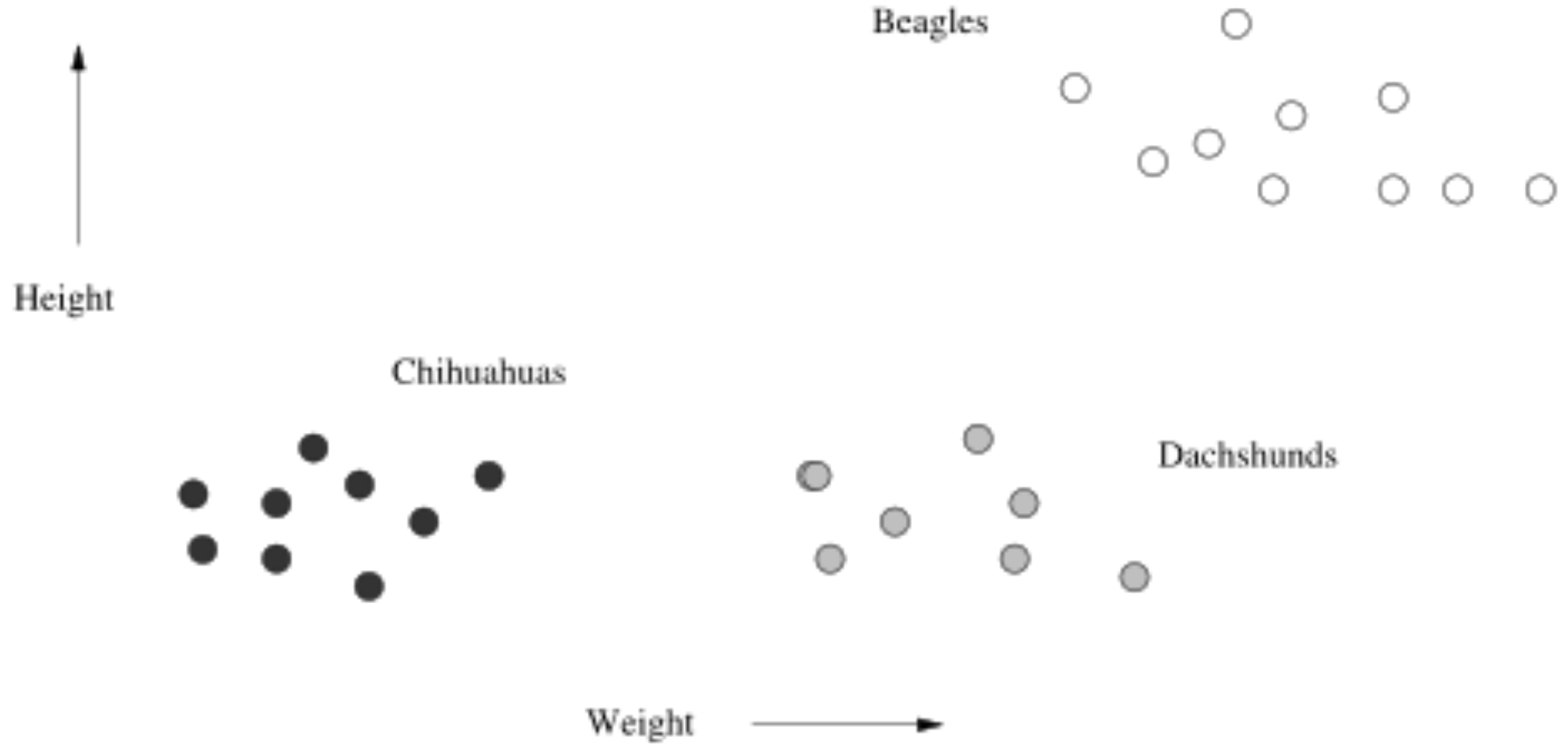


Figure 7.1: Heights and weights of dogs taken from three varieties

Clustering

- Clustering is the process of making clusters, which put **similar** things together into same cluster …
- And put **dissimilar** things into different clusters
- Need a similarity function
- Need a ~~similarity~~ **distance** function
  - Convenient to map items to points in space

Distance Functions

- Jaccard Distance
- Hamming Distance
- Euclidean Distance
- Cosine Distance
- Edit Distance
- …

- What is a **distance** function
  - D(x,y) >= 0
  - D(x,y) = D(y,x)
  - D(x,y) <= D(x,z) + D(z,y)

Clustering Strategies

- Hierarchical or Agglomerative
  - Bottom-up
- Partitioning methods
  - Top-down
- Density-based
- Cluster-based
- Iterative methods

Curse of Dimensionality

- N points in d-dimensional space
  - If d = 1, then average distance = 1/3
  - As d gets larger, what is the average distance? Distribution of distances?
    - # of **nearby** points for any a given point **vanishes.** So, clustering does not work well
    - # of points at max distance (~sqrt(d)) also vanishes. Real range actually very small
  - Angle ABC given 3 points approaches 90
    - Denominator grows linearly with d
    - Expected cos = 0 since equal points expected in all 4 quadrants

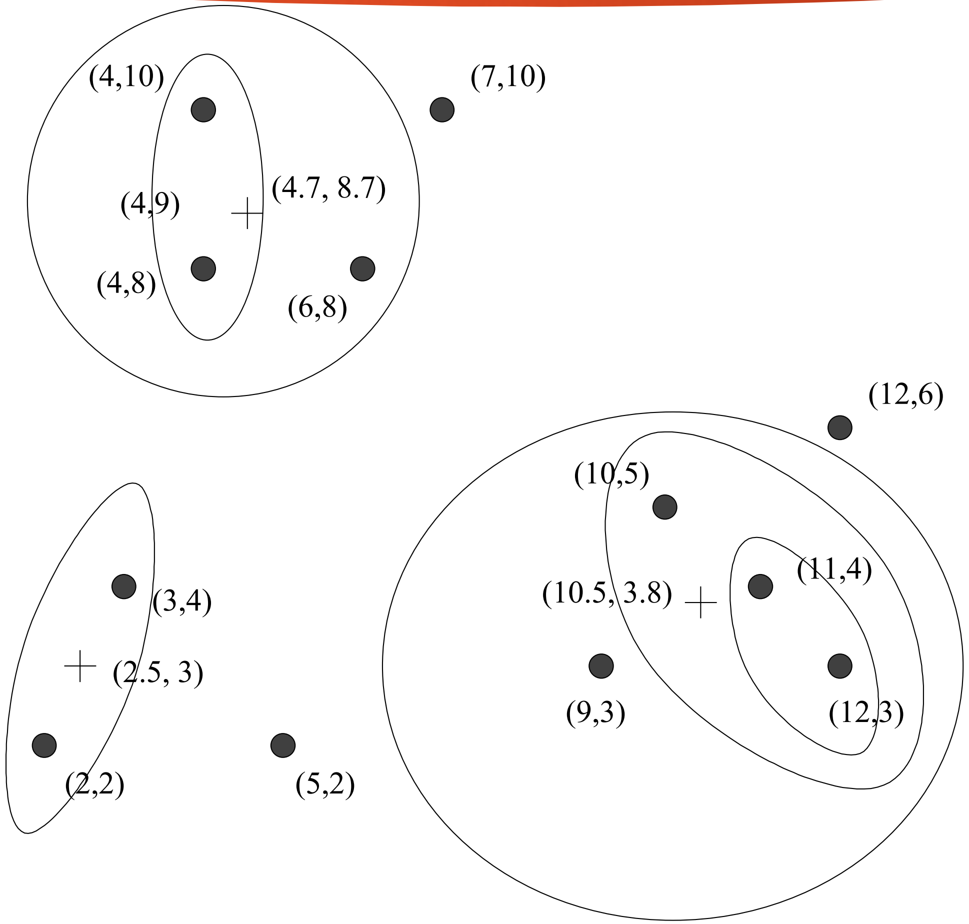
$$\frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2}\sqrt{\sum_{i=1}^{d} y_i^2}}$$

Hierarchical Clustering

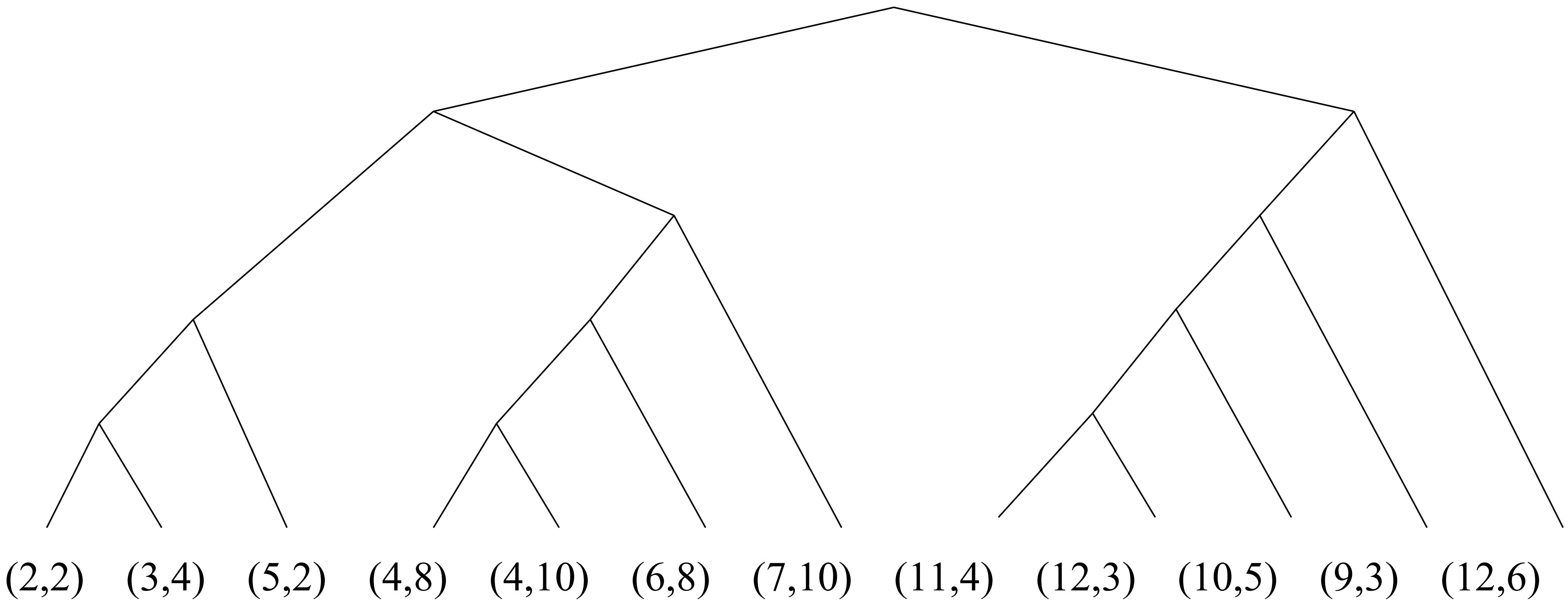
Hierarchical Clustering

- Starts with each item in different clusters
- Bottom up
- In each iteration
  - Two clusters are identified and merged into one
- Items are combined as the algorithm progresses
- **Questions**:
  - How are clusters represented
  - How to decide which ones to merge
  - What is the sopping condition
- Typical algorithm: find smallest distance between nodes of different clusters

Hierarchical Clustering



Output of Clustering: Dendrogram



Measures for a cluster

- Radius: largest distance from a centroid
- Diameter: largest distance between some pair of points in cluster
- Density: # of points per unit volume
- Volume: some power of radius or diameter

- **Good cluster**: when diameter of each cluster is much larger than its nearest cluster or nearest point outside cluster

Stopping condition for clustering

- Cluster radius or diameter crosses a threshold
- Cluster density drops below a certain threshold
- Ratio of diameter to distance to nearest cluster drops below a certain threshold