

Clustering

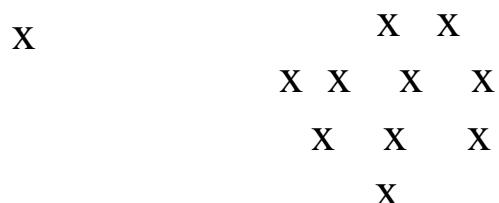
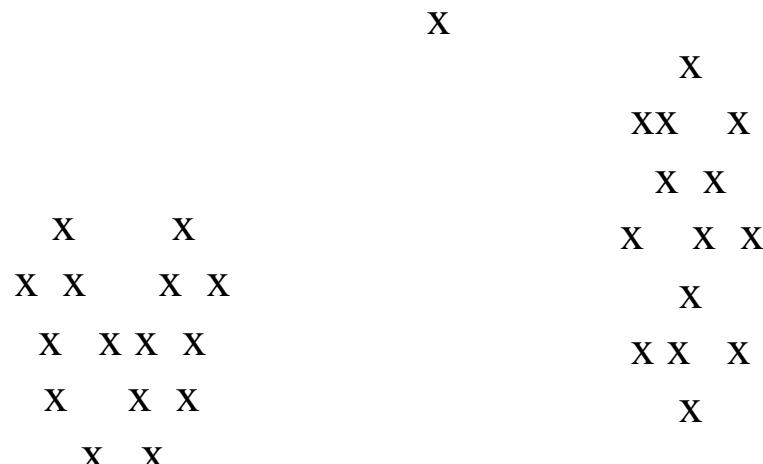
Applications
Overview of methods

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



The Problem

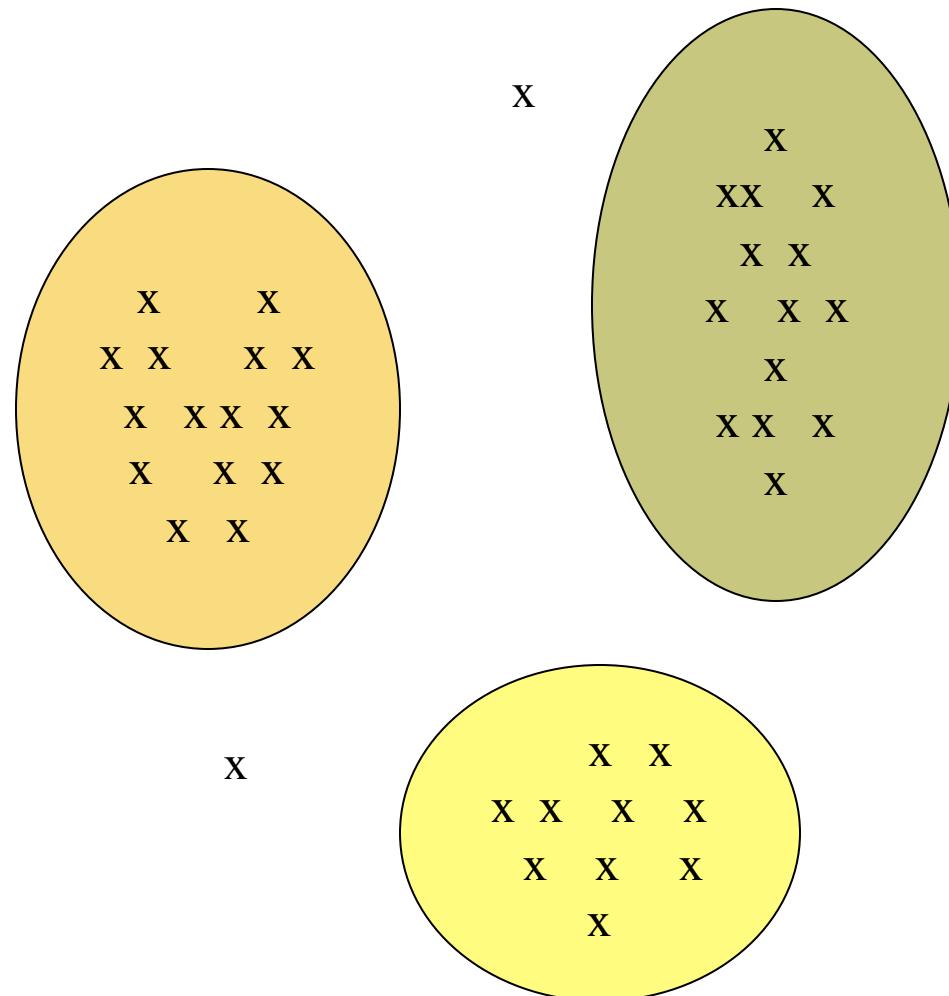
Given a cloud of data points we'd like to understand their structure



More formally:

- Given a **set of points**, with a notion of **distance** between points, **group the points** into some number of ***clusters***, so that
 - Members of a cluster are close/similar to each other
 - Members of different clusters are dissimilar
- **Usually:**
 - Points are in a high-dimensional space
 - Similarity is defined using a distance measure
 - Euclidean, Cosine, Jaccard, edit distance, ...

Example: Clusters



Clustering is a hard problem!



Why is it hard?

- Clustering in two dimensions looks easy
 - Clustering small amounts of data looks easy
 - And in most cases, looks are *not* deceiving
-
- Many applications involve not 2, but 10 or 10,000 dimensions
 - **High-dimensional spaces look different:**
Almost all pairs of points are at about the same distance

Clustering Sky Objects: SkyCat

- A catalog of 2 billion “sky objects” represents objects by their radiation in 7 dimensions (frequency bands)
- Problem: Cluster into similar objects, e.g., galaxies, stars, quasars, etc.
- Sloan Digital Sky Survey is a newer, better version of this

Example: Clustering CD's

- Intuitively: Music divides into **categories**, and customers prefer a few categories
 - But what are categories really?
- Represent a CD by a set of customers who bought it
- Similar CDs have similar sets of customers, and vice-versa

Example: Clustering Documents

- Problem: Group together documents on the same topic
- Documents with similar sets of words may be about the same topic
- Dual formulation: a topic is a group of words that co-occur in many documents
 - Cluster words instead of documents

Cosine, Jaccard, Euclidean

- Different ways of representing documents or CDs lead to different distance measures
- Document = **set** of words
 - Jaccard distance
- Document = **point** in space of words
 - (x_1, x_2, \dots, x_N) , where $x_i = 1$ iff word i appears in doc
 - Euclidean distance
- Document = **vector** in space of words
 - Vector from origin to (x_1, x_2, \dots, x_N)
 - Cosine distance

Overview: Methods of Clustering

■ Hierarchical:

- **Agglomerative** (bottom up):

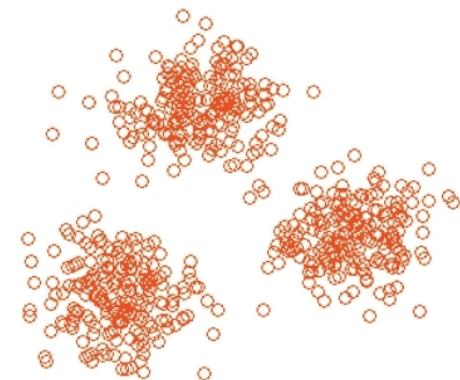
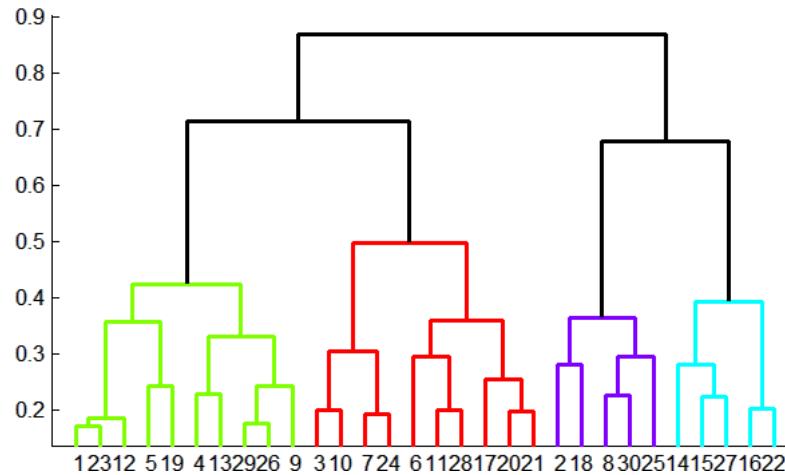
- Initially, each point is a cluster
- Repeatedly combine the two “nearest” clusters into one

- **Divisive** (top down):

- Start with one cluster and recursively split it

■ Point assignment:

- Maintain a set of clusters
- Points belong to “nearest” cluster



Clustering

Hierarchical Clustering

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Hierarchical Clustering

- **Hierarchical:**

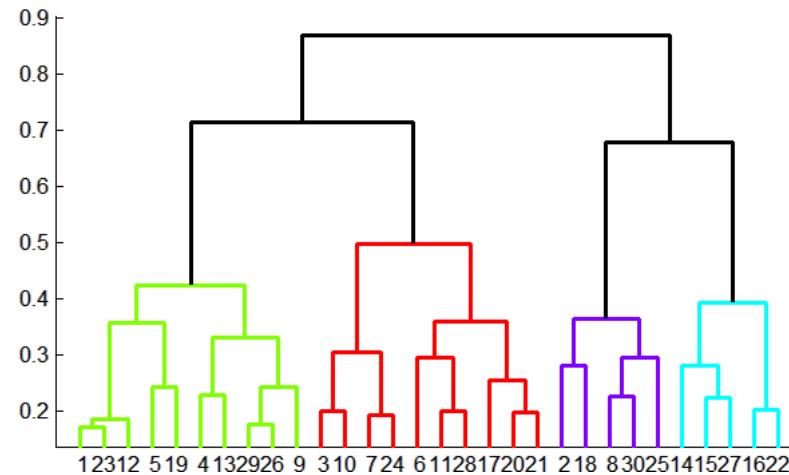
- **Agglomerative** (bottom up):

- Initially, each point is a cluster
 - Repeatedly combine the two “nearest” clusters into one

- **Divisive** (top down):

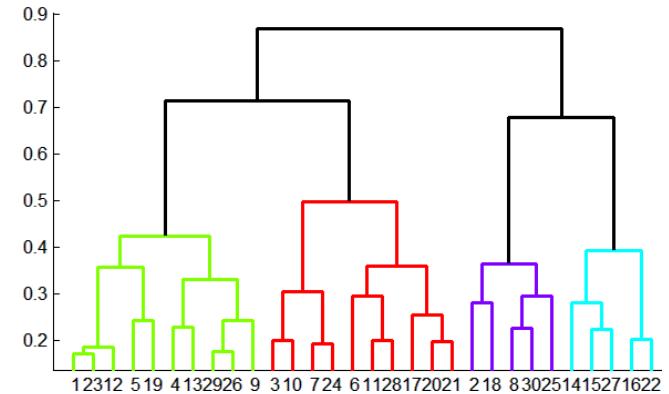
- Start with one cluster and recursively split it

- This lecture: agglomerative approach
 - Same ideas can be used for divisive



Hierarchical Clustering

- **Key operation:**
Repeatedly combine two nearest clusters

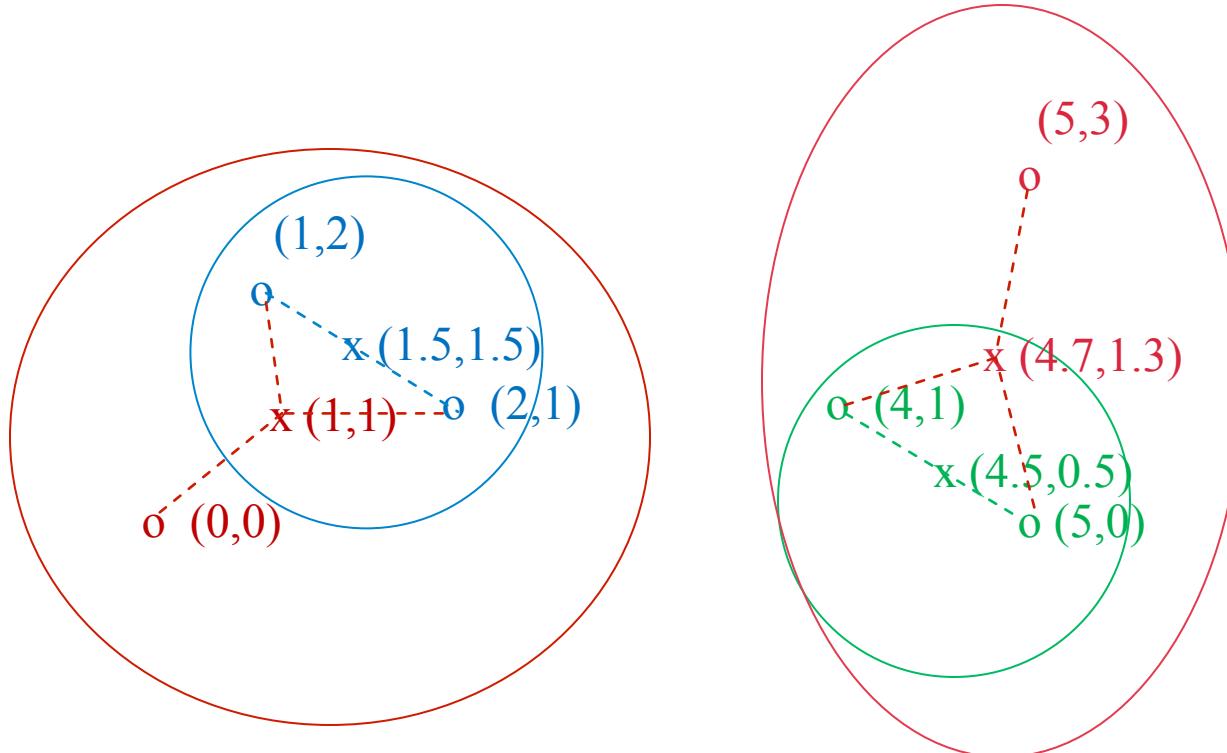


- **Three important questions:**
 - 1) How do you represent a cluster of more than one point?
 - 2) How do you determine the “nearness” of clusters?
 - 3) When to stop combining clusters?

Euclidean Space

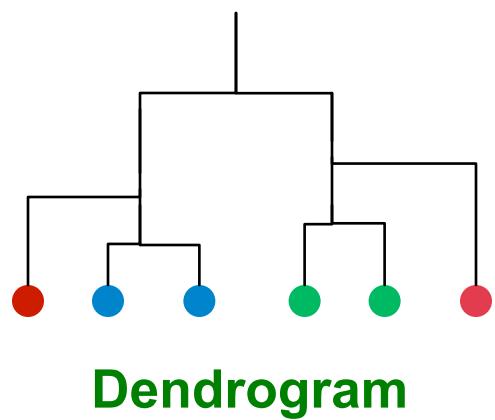
- **(1) How to represent a cluster of many points?**
 - How do you represent the location of each cluster, to tell which pair of clusters is closest?
 - Represent each cluster by its *centroid* = average of its points
- **(2) How to determine “nearness” of clusters?**
 - Measure cluster distances by distances of centroids

Example: Hierarchical clustering



Data:

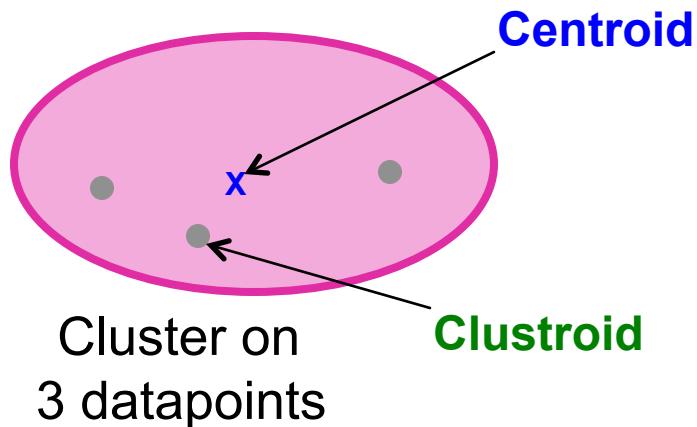
- o ... data point
- x ... centroid



Non-Euclidean Case

- The only “locations” we can talk about are the points themselves
 - i.e., there is no “average” of two points
- **(1) How to represent a cluster of many points?**
clustroid = (data)point “closest” to other points
- **(2) How do you determine the “nearness” of clusters?** Treat clustroid as if it were centroid, when computing intercluster distances

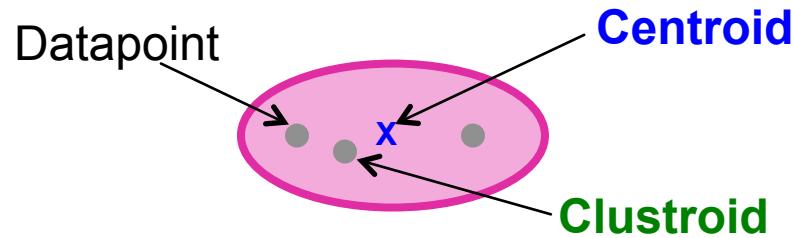
Clustroid



Centroid is the avg. of all (data)points in the cluster. This means centroid is an “artificial” point.

Clustroid is an **existing** (data)point that is “closest” to all other points in the cluster.

“Closest” Point?



- ***Clustroid*** = point “closest” to other points
- Possible meanings of “closest”:
 - Smallest maximum distance to other points
 - Smallest average distance to other points
 - Smallest sum of squares of distances to other points

Termination condition

- (3) When do you stop combining clusters?
- Approach 1: Pick a number k upfront, and stop when we have k clusters
 - Makes sense when we know that the data naturally falls into k classes
- Approach 2: Stop when the next merge would create a cluster with low “cohesion”
 - i.e, a “bad” cluster

Cohesion

- **Approach 3.1: Diameter** of the merged cluster = maximum distance between points in the cluster
- **Approach 3.2: Radius** = maximum distance of a point from centroid (or clustroid)
- **Approach 3.3: Use a density-based approach**
 - Density = number of points per unit volume
 - E.g., divide number of points in cluster by diameter or radius of the cluster
 - Perhaps use a power of the radius (e.g., square or cube)

Implementation

- **Naïve implementation of hierarchical clustering:**
 - At each step, compute pairwise distances between all pairs of clusters, then merge
 - $O(N^3)$
- Careful implementation using priority queue can reduce time to $O(N^2 \log N)$
 - **Still too expensive for really big datasets that do not fit in memory**

Clustering

k-Means Algorithm

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



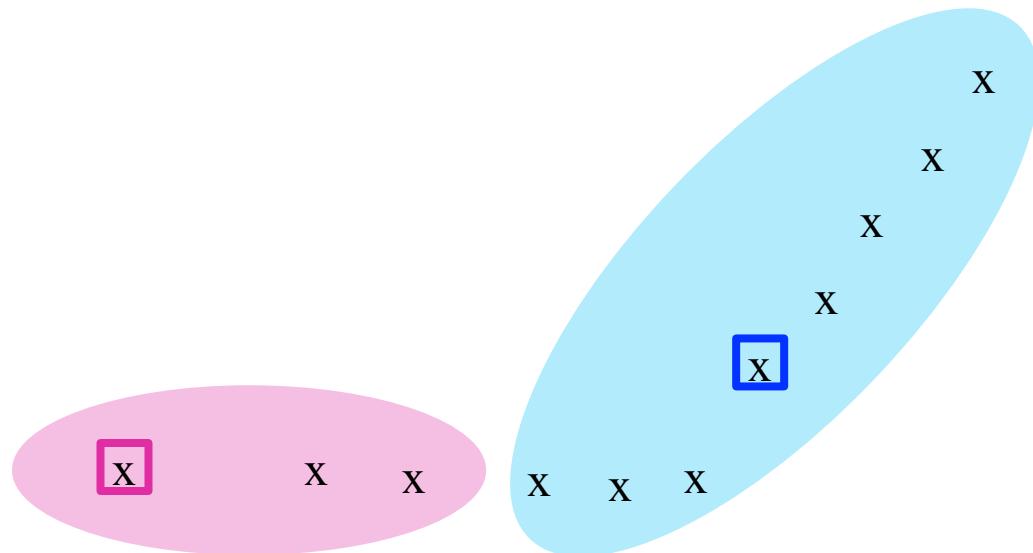
k -means Algorithm

- Assumes Euclidean space/distance
- Start by picking k , the number of clusters
- Initialize clusters by picking one point per cluster
 - For the moment, assume we pick the k points at random

Populating Clusters

- 1) For each point, place it in the cluster whose current centroid it is nearest
- 2) After all points are assigned, update the locations of centroids of the k clusters
- 3) Reassign all points to their closest centroid
 - Sometimes moves points between clusters
- **Repeat 2 and 3 until convergence**
 - **Convergence:** Points don't move between clusters and centroids stabilize

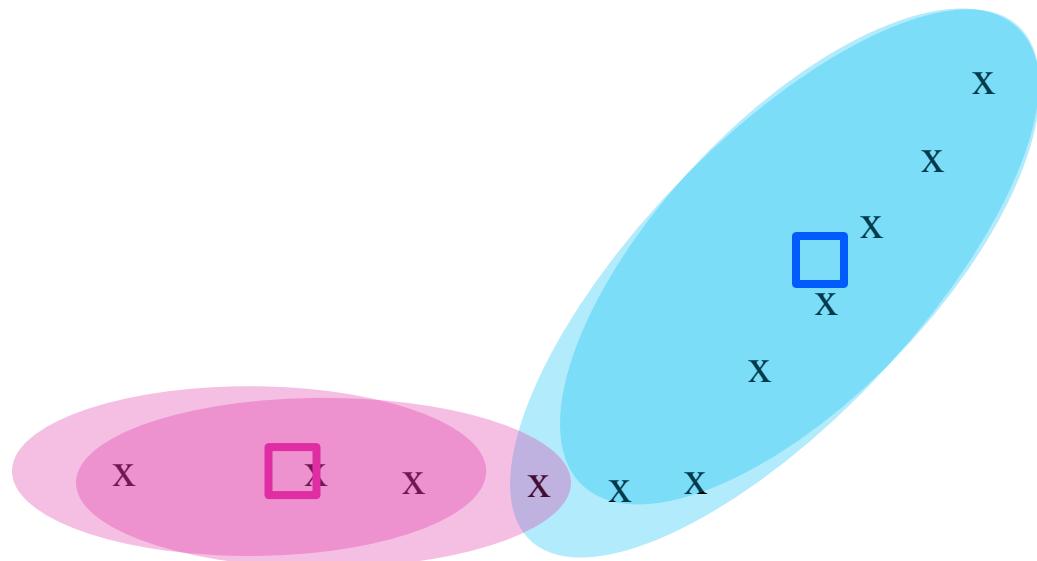
Example: $k = 2$



X ... data point
□ ... centroid

Round 1

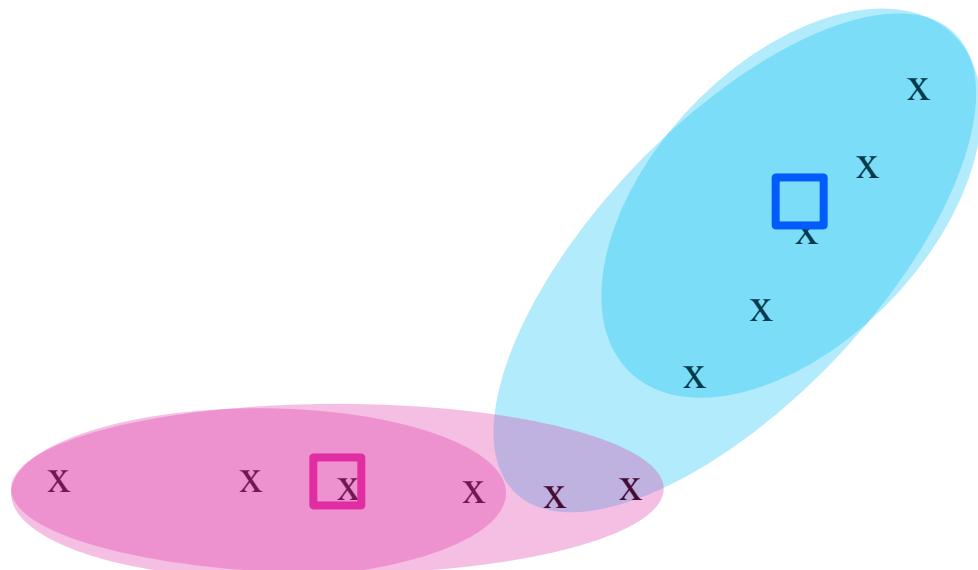
Example: Assigning Clusters



X ... data point
 \square ... centroid

Round 2

Example: Assigning Clusters



X ... data point
□ ... centroid

Round 3

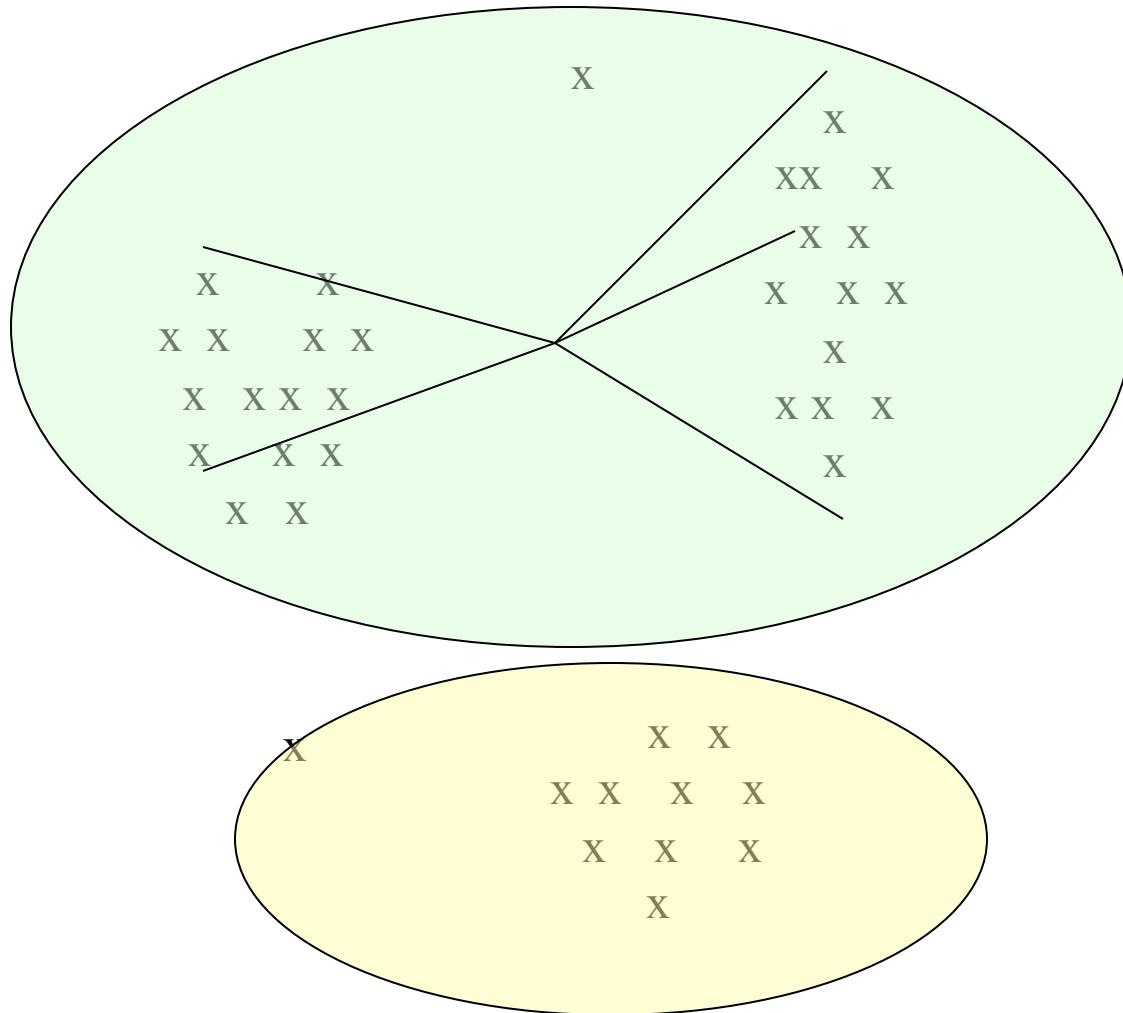
Picking the right value for k

How to select k ?

- Try different k , looking at the change in the average distance to centroid, as k increases.

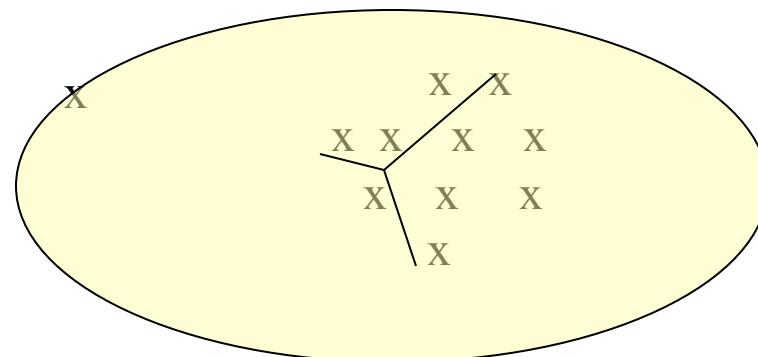
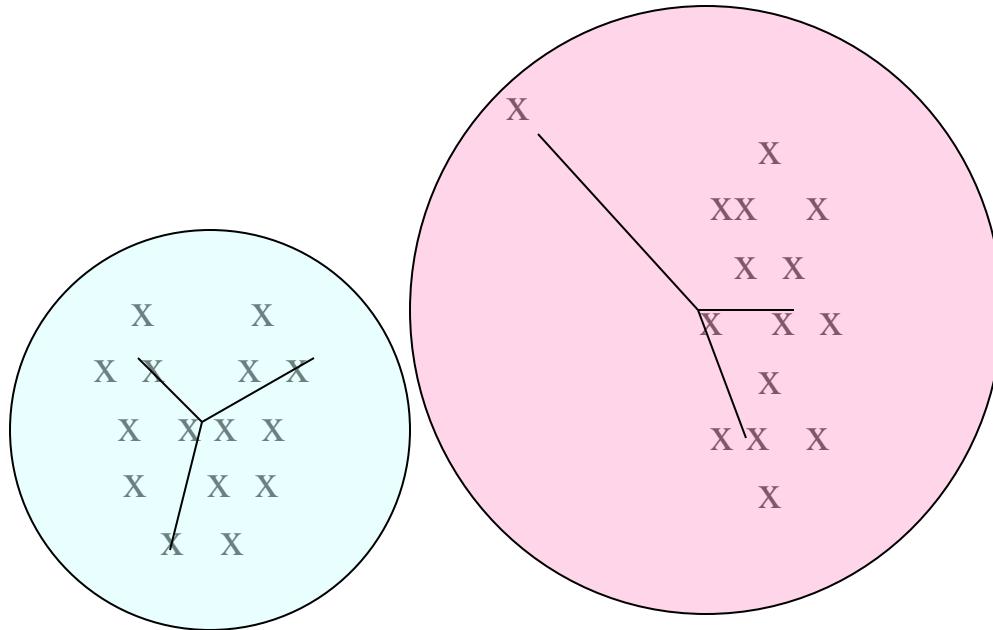
Example: Picking k

Too few;
many long
distances
to centroid.



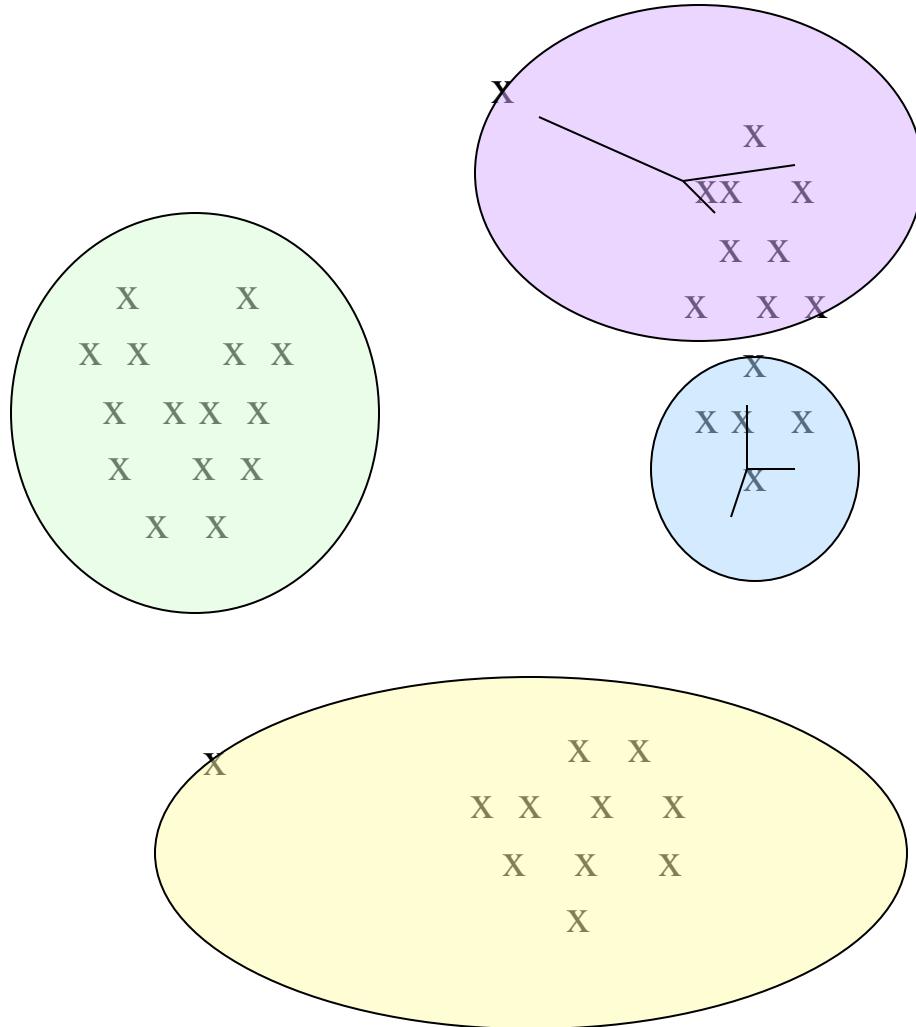
Example: Picking k

Just right;
distances
rather short.



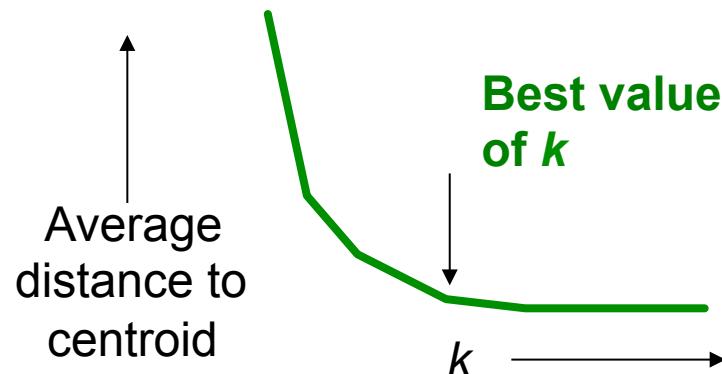
Example: Picking k

Too many;
little improvement
in average
distance.



Picking the right value for k

Average falls rapidly until right k , then falls much more slowly



Picking the initial k points

- Approach 1: Sampling
 - Cluster a sample of the data using hierarchical clustering, to obtain k clusters
 - Pick a point from each cluster (e.g. point closest to centroid)
 - Sample fits in main memory
- Approach 2: Pick “dispersed” set of points
 - Pick first point at random
 - Pick the next point to be the one whose minimum distance from the selected points is as large as possible
 - Repeat until we have k points

Complexity

- In each round, we have to examine each input point exactly once to find closest centroid
- Each round is $O(kN)$ for N points, k clusters
- But the number of rounds to convergence can be very large!
- Can we cluster in a single pass over the data?

Clustering

Bradley-Fayyad-Reina (BFR) Algorithm

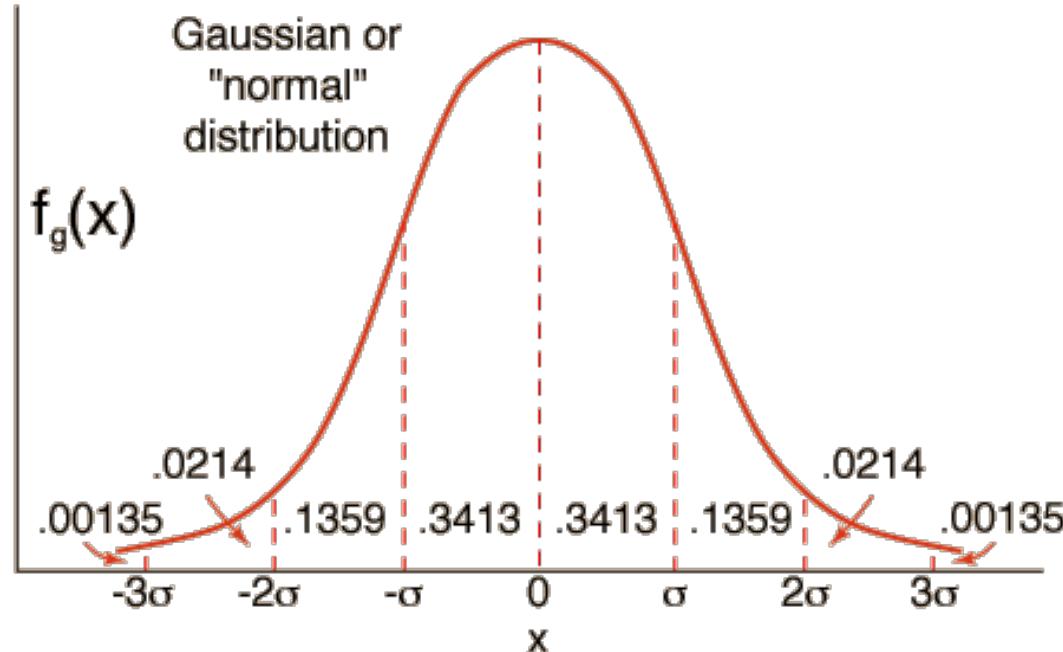
Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



BFR Algorithm

- **BFR** [Bradley-Fayyad-Reina] is a variant of k -means for very large (disk-resident) data sets
- Assumes each cluster is **normally distributed** around a centroid in Euclidean space

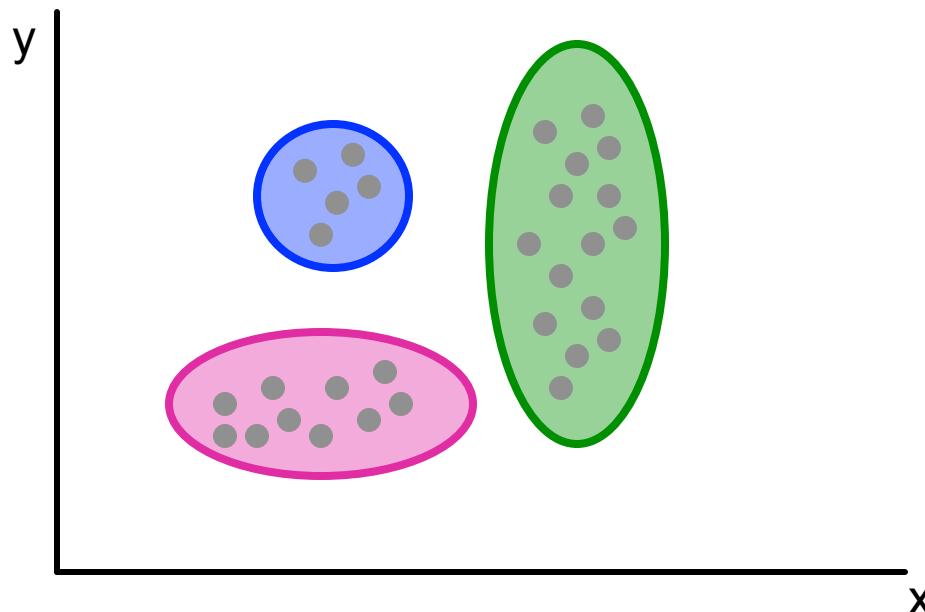
Normal Distribution



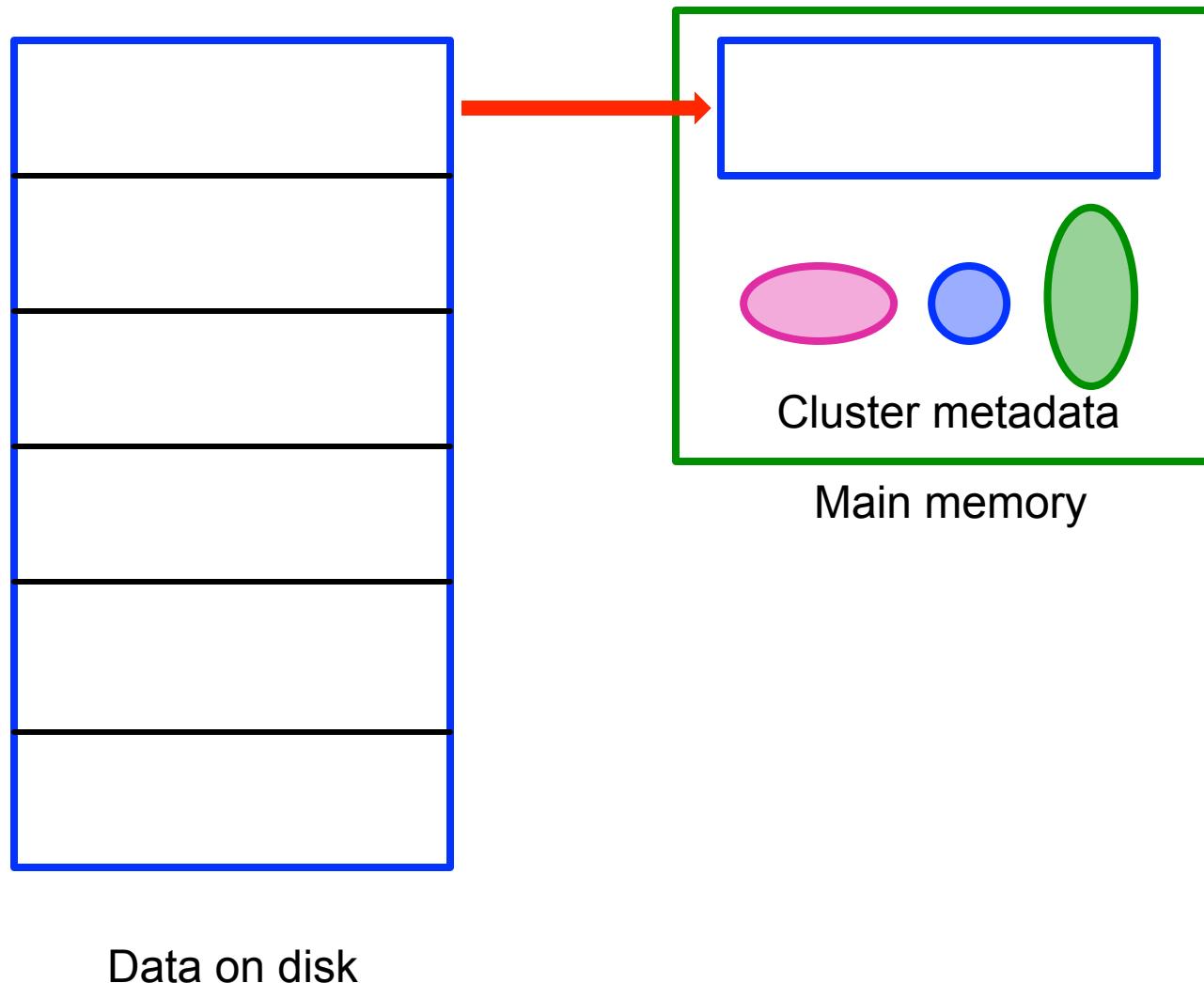
- Can quantify the likelihood of finding a point in the cluster at a given distance from the centroid along each dimension
- Standard deviations in different dimensions may vary

BFR Clusters

- Normal distribution assumption implies that clusters “look like” axis-aligned ellipses



BFR Algorithm: Overview



BFR Algorithm

- Points are read from disk one main-memory-full at a time
- Most points from previous memory loads are summarized by **simple statistics**
- To begin, from the initial load we select the initial k centroids by some sensible approach
 - Using one of the techniques from the k-Means lecture

Three Classes of Points

3 sets of points which we keep track of:

- **Discard set (DS):**

- Points close enough to a centroid to be summarized

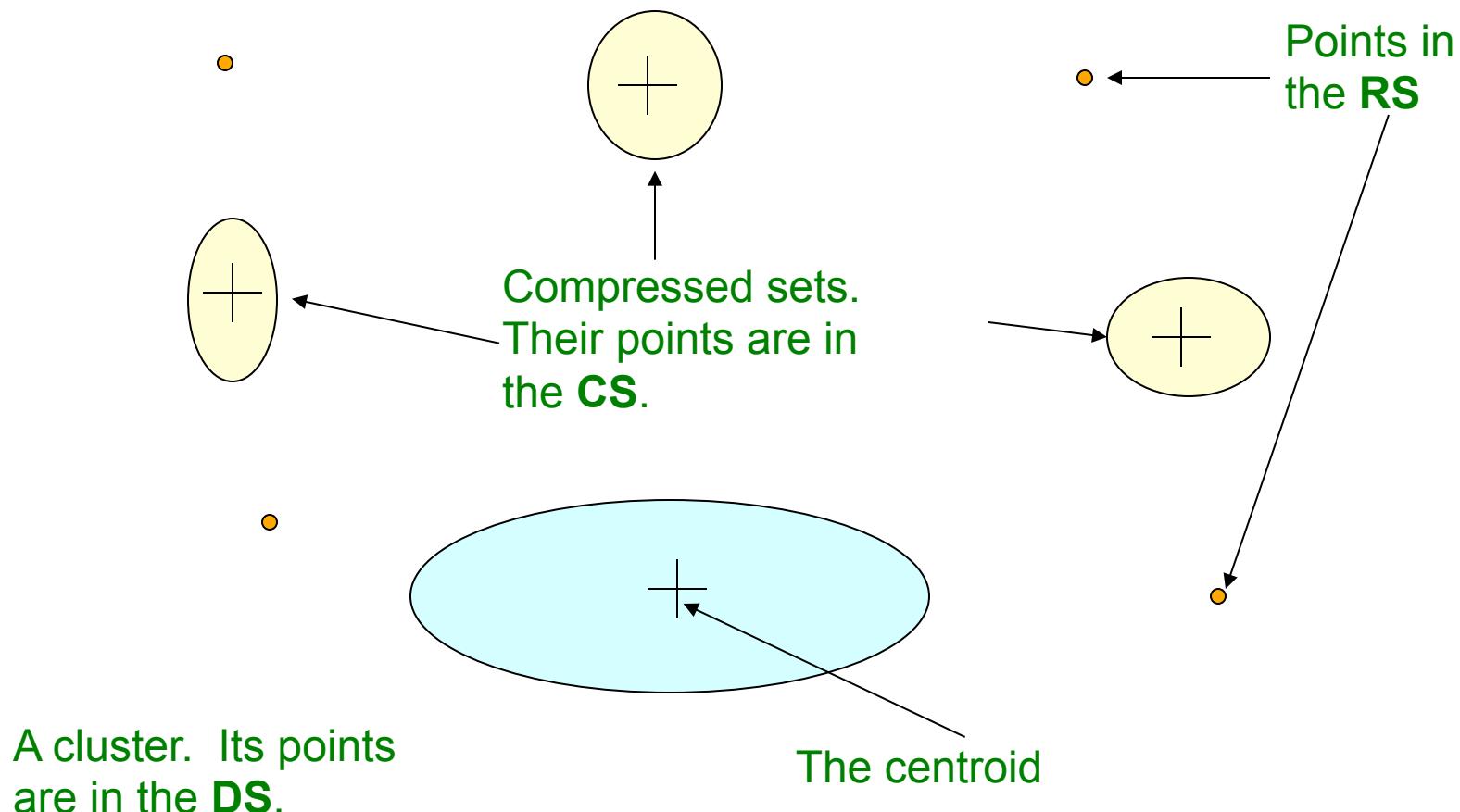
- **Compression set (CS):**

- Groups of points that are close together but not close to any existing centroid
 - These points are summarized, but not assigned to a cluster

- **Retained set (RS):**

- Isolated points waiting to be assigned to a compression set

BFR: “Galaxies” Picture

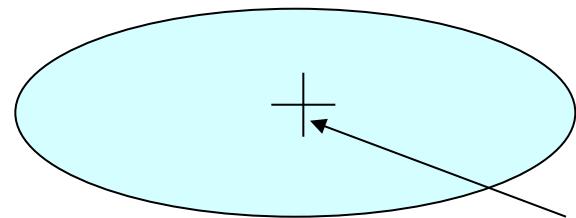


Discard set (DS): Close enough to a centroid to be summarized
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

Summarizing Sets of Points

For each cluster, the discard set (DS) is summarized by:

- The number of points, N
- The vector SUM , whose i^{th} component = sum of the coordinates of the points in the i^{th} dimension
- The vector $SUMSQ$: i^{th} component = sum of squares of coordinates in i^{th} dimension



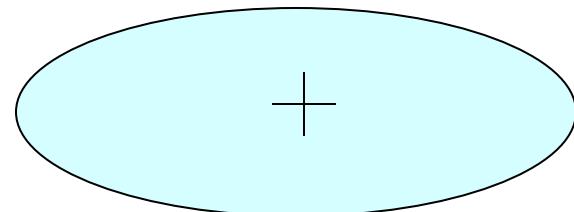
A cluster.

All its points are in the DS.

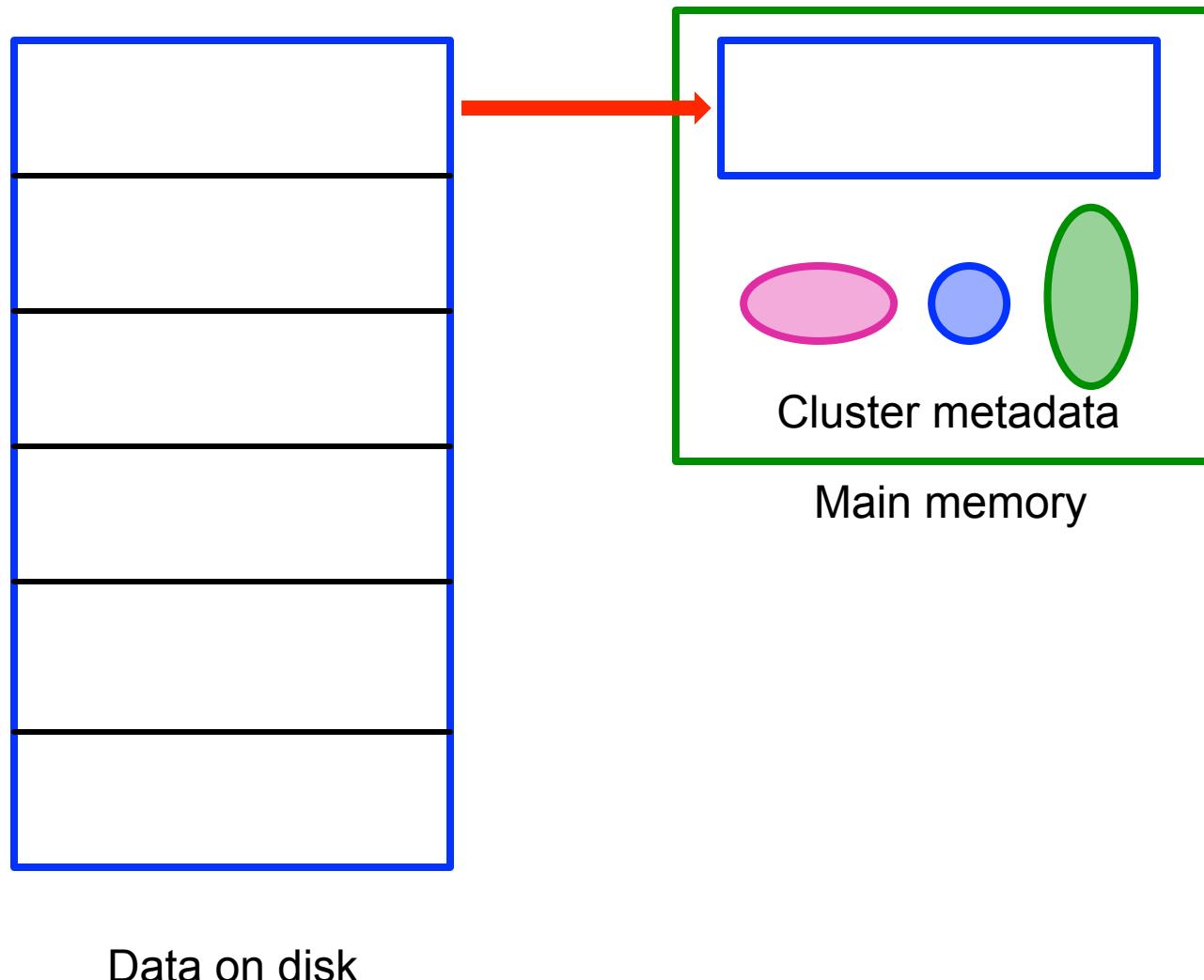
The centroid

Summarizing Points: Comments

- **$2d + 1$** values represent any size cluster
 - d = number of dimensions
- Average in **each dimension (the centroid)** can be calculated as **SUM_i / N**
 - SUM_i = i^{th} component of SUM
- Variance of a cluster's discard set in dimension i is: **$(\text{SUMSQ}_i / N) - (\text{SUM}_i / N)^2$**
 - And standard deviation is the square root of that
- **Next step: Actual clustering**



BFR Algorithm: Overview



Processing a chunk of points (1)

- Find those points that are “**sufficiently close**” to a cluster centroid
- Add those points to that cluster and the **DS**
 - Then discard the point
- **DS set:** Adjust statistics of each cluster to account for newly added points
 - Add **Ns**, **SUMs**, **SUMSQs**

Discard set (DS): Close enough to a centroid to be summarized.

Compression set (CS): Summarized, but not assigned to a cluster

Retained set (RS): Isolated points

Processing a chunk of points (2)

- The remaining points are not close to any cluster
- Use any main-memory clustering algorithm to cluster these points and the old **RS**
 - Clusters go to the **CS**; outlying points to the **RS**

Discard set (DS): Close enough to a centroid to be summarized.
Compression set (CS): Summarized, but not assigned to a cluster
Retained set (RS): Isolated points

Processing a chunk of points (3)

- Consider merging compressed sets in the **CS**
- If this is the last round, merge all compressed sets in the **CS** and all **RS** points into their nearest cluster

Discard set (DS): Close enough to a centroid to be summarized.

Compression set (CS): Summarized, but not assigned to a cluster

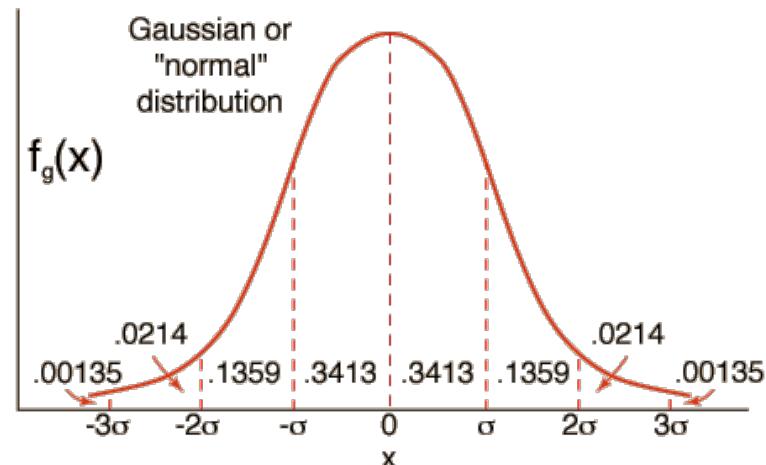
Retained set (RS): Isolated points

A Few Details...

- Q1) How do we decide if a point is “close enough” to a cluster that we will add the point to that cluster?
- Q2) How do we decide whether two compressed sets (CS) deserve to be combined into one?

How Close is Close Enough?

- Q1) We need a way to decide whether to put a new point into a cluster (and discard)
- BFR approach:
 - The **Mahalanobis distance** is less than a threshold
 - **High likelihood of the point belonging to currently nearest centroid**



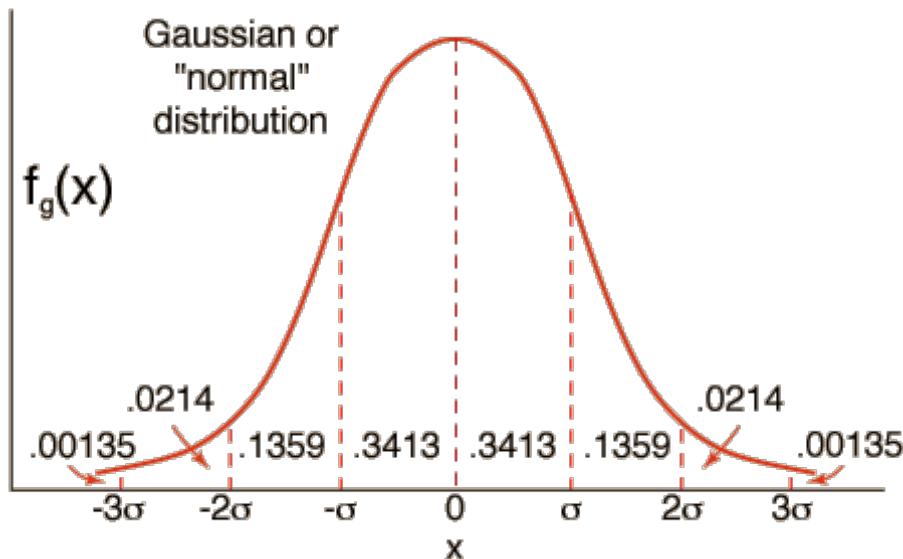
Mahalanobis distance

- Cluster C has centroid (c_1, \dots, c_d) and standard deviations $(\sigma_1, \dots, \sigma_d)$
- Point P = (x_1, \dots, x_d)
- Normalized distance in dimension i:
 $y_i = (x_i - c_i)/\sigma_i$
- MD of point P from cluster C:

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

Mahalanobis Acceptance Criterion

- Suppose point P is one standard dimension away from centroid in each dimension
 - Each $y_i = 1$ and so the MD of P is \sqrt{d}



68% of points have $MD \leq \sqrt{d}$

95% of points have $MD \leq 2\sqrt{d}$

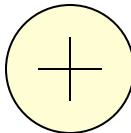
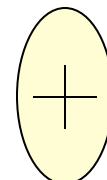
99% of points have $MD \leq 3\sqrt{d}$

Accept point P into cluster C if its MD from cluster centroid is less than a threshold e.g., $3\sqrt{d}$

Should 2 CS clusters be combined?

Q2) Should 2 CS subclusters be combined?

- Compute the variance of the combined subcluster
 - N , SUM , and $SUMSQ$ allow us to make that calculation quickly
- Combine if the combined variance is below some threshold
- **Many alternatives:** Treat dimensions differently, consider density



Clustering

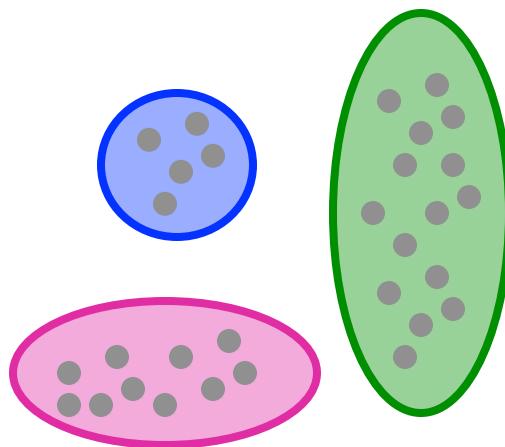
CURE Algorithm

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University

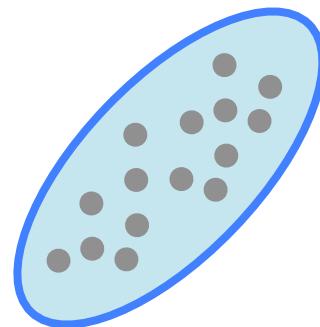


Limitations of BFR Algorithm

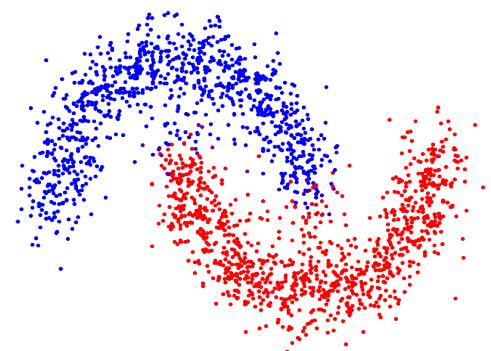
- Makes strong assumptions:
 - (1) Clusters normally distributed in each dimension
 - (2) Axes are fixed – ellipses at an angle are **not** OK



OK



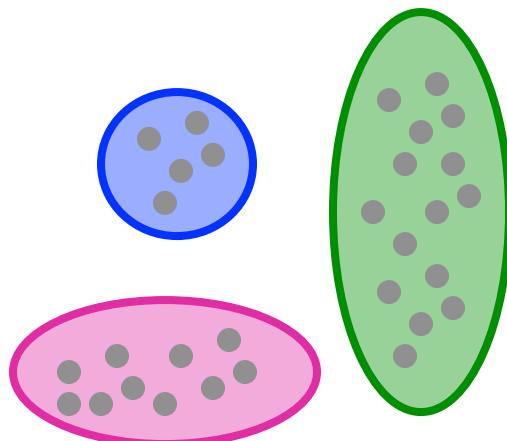
Not OK



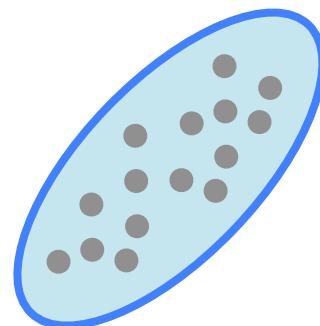
Not OK

CURE Algorithm

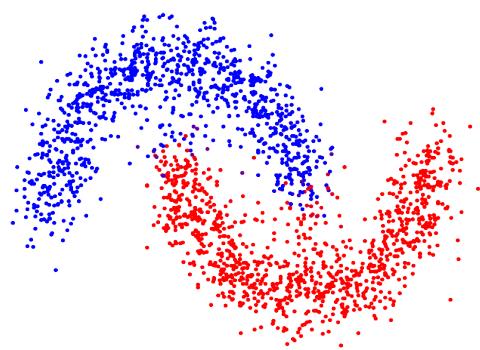
- CURE (Clustering Using REpresentatives):
 - Assumes a Euclidean distance
 - Allows clusters to assume any shape
 - **Uses a collection of representative points to represent clusters**



OK

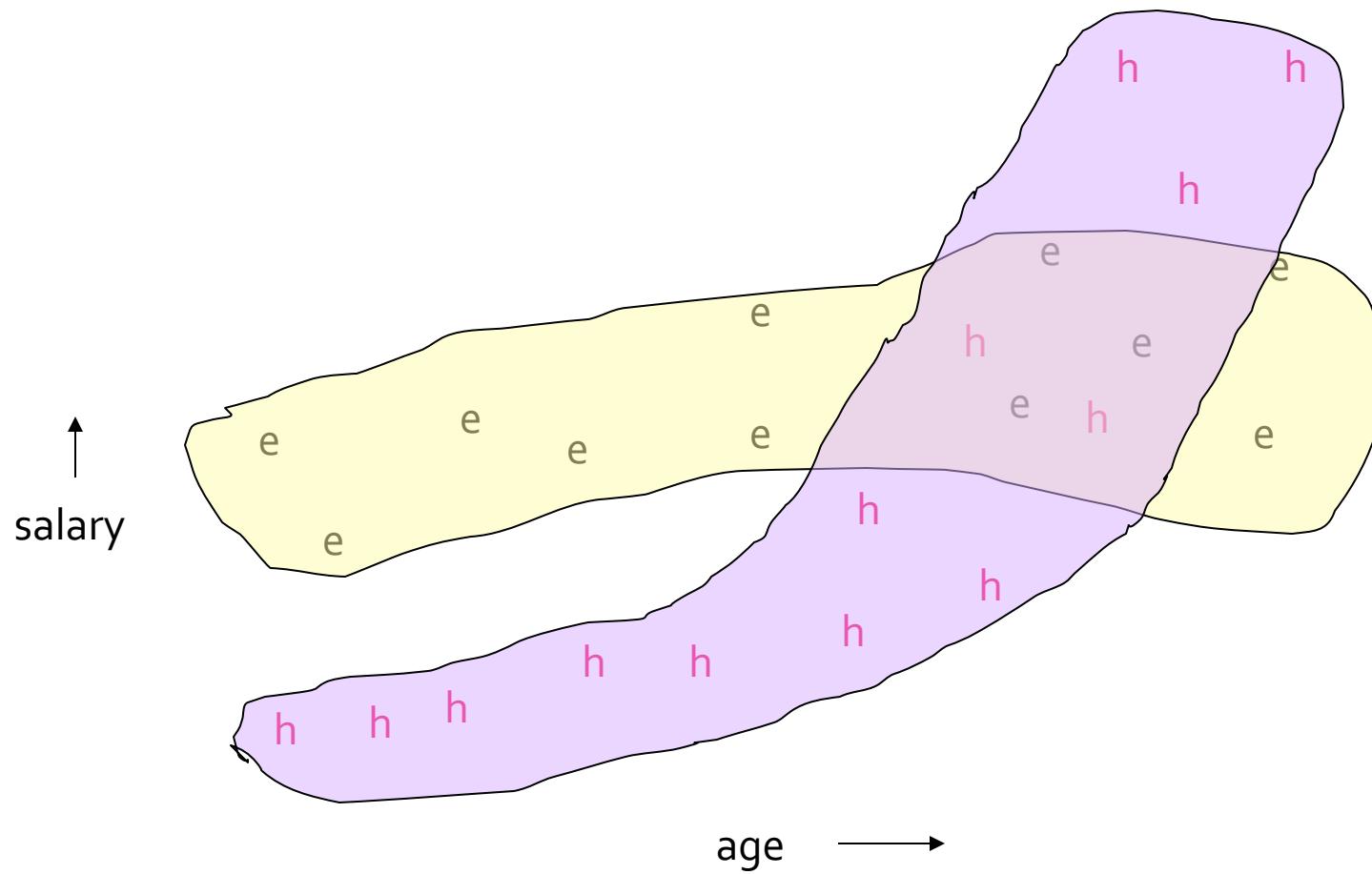


OK



OK

Example: Stanford Salaries

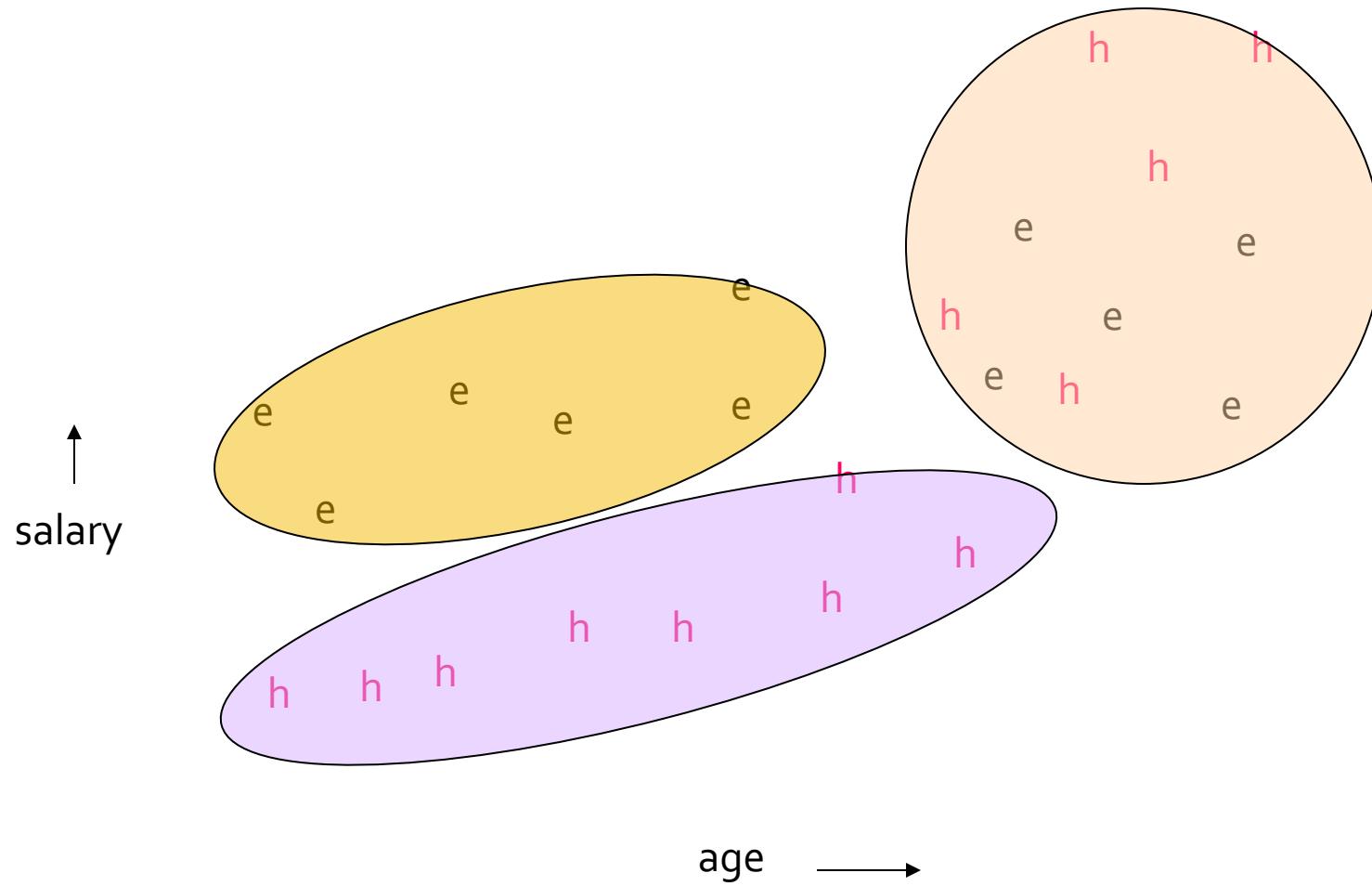


Starting CURE

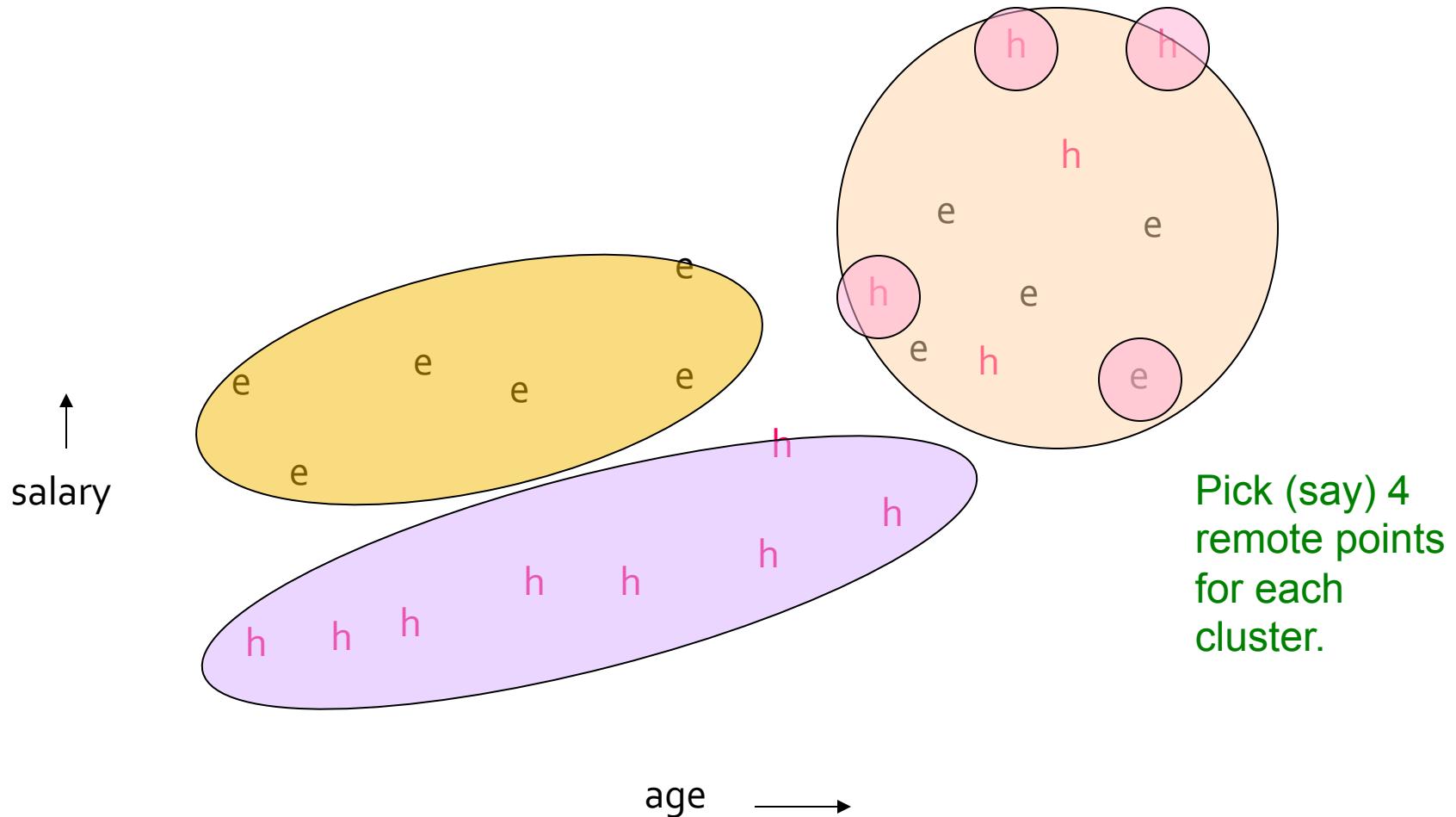
Pass 1 of 2:

- Pick a random sample of points that fit in main memory
- Cluster sample points hierarchically to create the initial clusters
- **Pick representative points:**
 - For each cluster, pick k (e.g., 4) representative points, as dispersed as possible
 - Move each representative point a fixed fraction (e.g., 20%) toward the centroid of the cluster

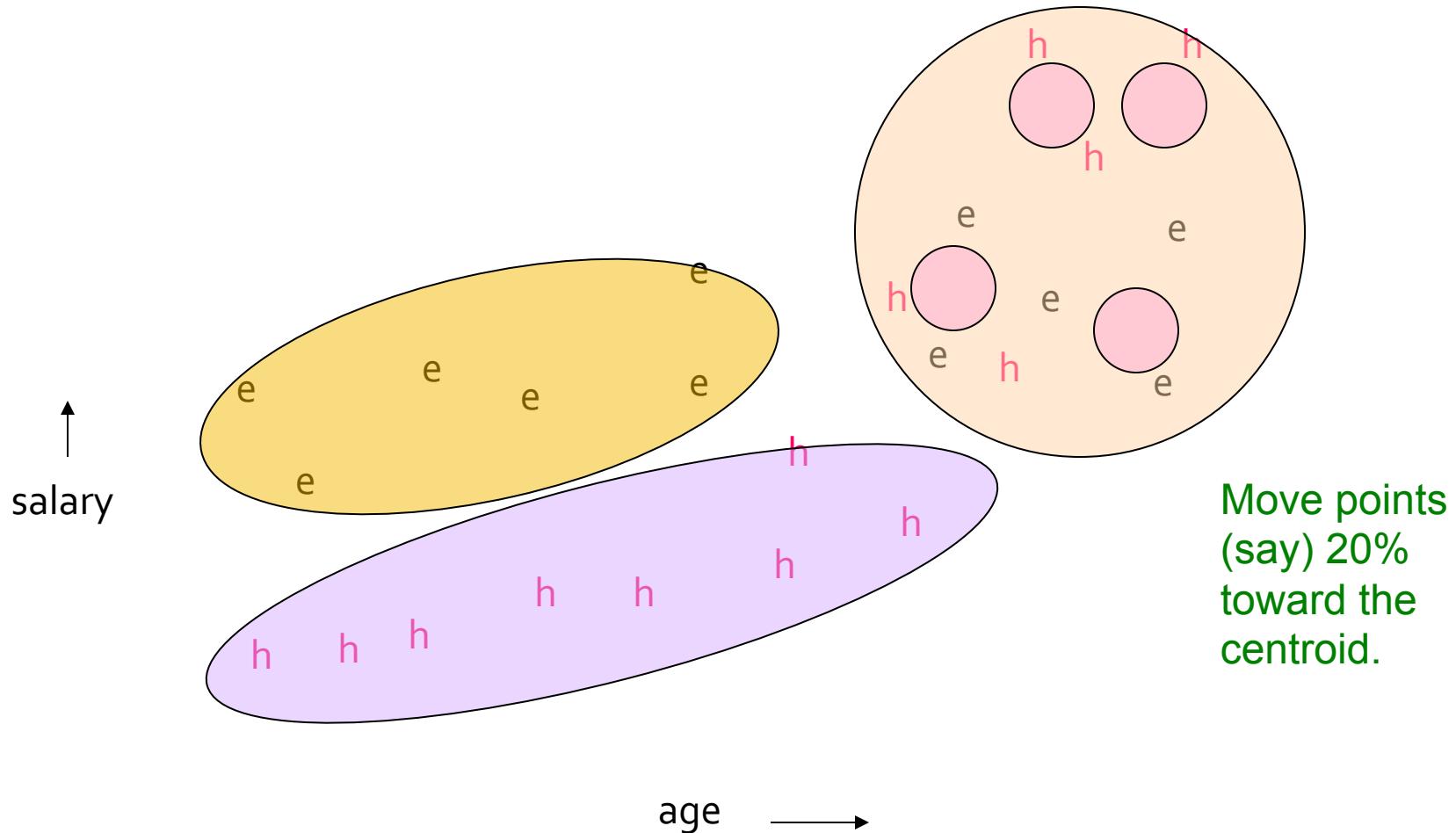
Example: Initial Clusters



Example: Pick Dispersed Points



Example: Pick Dispersed Points



Finishing CURE

Pass 2 of 2:

- Now, rescan the whole dataset and visit each point p in the data set
- Place it in the “closest cluster”
 - Normal definition of “closest”: that cluster with the closest (to p) among all the representative points of all the clusters
- And that’s it!

Summary

- **Clustering:** Given a **set of points**, with a notion of **distance** between points, **group the points** into some number of ***clusters***
- **Algorithms:**
 - Agglomerative **hierarchical clustering**
 - Centroid and clustroid
 - ***k*-means**
 - **BFR**
 - **CURE**

Frequent Itemsets

The Market-Basket Model
Association Rules
A-Priori Algorithm

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



The Market-Basket Model

- A large set of *items*, e.g., things sold in a supermarket.
- A large set of *baskets*, each of which is a small set of the items, e.g., the things one customer buys on one day.

Support

- Simplest question: find sets of items that appear “frequently” in the baskets.
- *Support* for itemset I = the number of baskets containing all items in I .
 - Sometimes given as a percentage.
- Given a *support threshold* s , sets of items that appear in at least s baskets are called *frequent itemsets*.

Example: Frequent Itemsets

- Items={milk, coke, pepsi, beer, juice}.
- Support = 3 baskets.

$$B_1 = \{m, c, b\}$$

$$B_3 = \{m, b\}$$

$$B_5 = \{m, p, b\}$$

$$B_7 = \{c, b, j\}$$

$$B_2 = \{m, p, j\}$$

$$B_4 = \{c, j\}$$

$$B_6 = \{m, c, b, j\}$$

$$B_8 = \{b, c\}$$

- Frequent itemsets: $\{m\}, \{c\}, \{b\}, \{j\}, \{m, b\}, \{b, c\}, \{c, j\}$.

Applications

- Items = products; baskets = sets of products someone bought in one trip to the store.
- Example application: given that many people buy beer and diapers together:
 - Run a sale on diapers; raise price of beer.
 - Only useful if many buy diapers & beer.
 - Essential for brick-and-mortar stores, not on-line stores.

Applications – (2)

- Baskets = sentences; items = documents containing those sentences.
- Items that appear together too often could represent plagiarism.
- Notice items do not have to be “in” baskets.
 - But it is better if baskets have small numbers of items, while items can be in large numbers of baskets.

Applications – (3)

- Baskets = documents; items = words.
- Unusual words appearing together in a large number of documents, e.g., “Brad” and “Angelina,” may indicate an interesting relationship.

Scale of the Problem

- WalMart sells 100,000 items and can store billions of baskets.
- The Web has billions of words and many billions of pages.

Association Rules

- If-then rules about the contents of baskets.
- $\{i_1, i_2, \dots, i_k\} \rightarrow j$ means: “if a basket contains all of i_1, \dots, i_k then it is *likely* to contain j .”
- *Confidence* of this association rule is the probability of j given i_1, \dots, i_k .
 - That is, the fraction of the baskets with i_1, \dots, i_k that also contain j .

Example: Confidence

$$\begin{array}{ll} + & B_1 = \{m, c, b\} \qquad B_2 = \{m, p, j\} \\ - & B_3 = \{m, b\} \qquad B_4 = \{c, j\} \\ - & B_5 = \{m, p, b\} \qquad + B_6 = \{m, c, b, j\} \\ & B_7 = \{c, b, j\} \qquad B_8 = \{b, c\} \end{array}$$

- An association rule: $\{m, b\} \rightarrow c$.
 - Confidence = $2/4 = 50\%$.

Finding Association Rules

- Question: “find all association rules with support $\geq s$ and confidence $\geq c$.”
 - Note: “support” of an association rule is the support of the set of items on the left.
- Hard part: finding the frequent itemsets.
 - Note: if $\{i_1, i_2, \dots, i_k\} \rightarrow j$ has high support and confidence, then both $\{i_1, i_2, \dots, i_k\}$ and $\{i_1, i_2, \dots, i_k, j\}$ will be “frequent.”

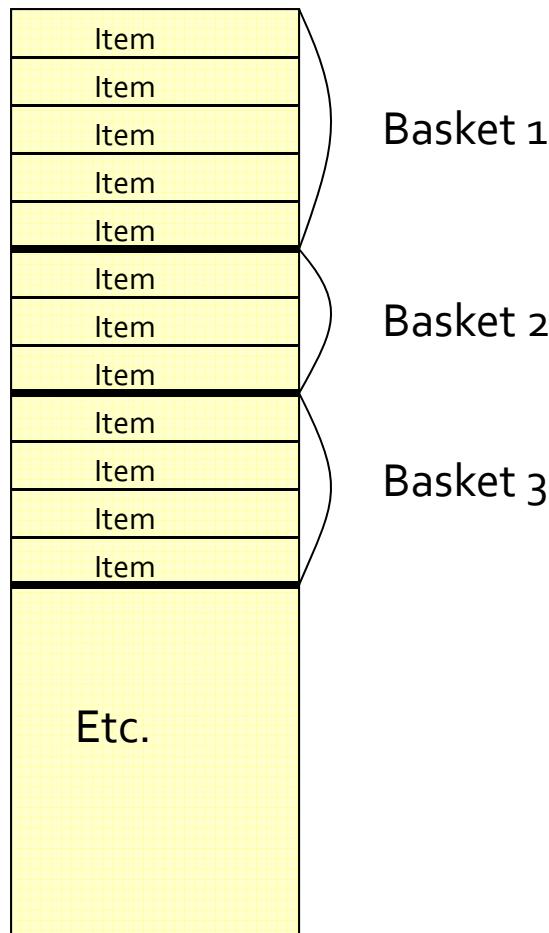
Finding Association Rules – (2)

1. Find all sets with support at least cs .
2. Find all sets with support at least s .
3. If $\{i_1, i_2, \dots, i_k, j\}$ has support at least cs , see which subsets missing one element have support at least s .
 - Take j to be the missing element.
4. $\{i_1, i_2, \dots, i_k\} \rightarrow j$ is an acceptable association rule if $\{i_1, i_2, \dots, i_k\}$ has support $s_1 \geq s$, $\{i_1, i_2, \dots, i_k, j\}$ has support $s_2 \geq cs$, and s_2/s_1 , the confidence of the rule, is at least c .

Computation Model

- Typically, data is kept in flat files.
- Stored on disk.
- Stored basket-by-basket.
- Expand baskets into pairs, triples, etc. as you read baskets.
 - Use k nested loops to generate all sets of size k .

File Organization



Example: items are positive integers, and boundaries between baskets are -1 .

Computation Model – (2)

- The true cost of mining disk-resident data is usually the **number of disk I/O's**.
- In practice, algorithms for finding frequent itemsets read the data in *passes* – all baskets read in turn.
- Thus, we measure the cost by the **number of passes** an algorithm takes.

Main-Memory Bottleneck

- For many frequent-itemset algorithms, main memory is the critical resource.
- As we read baskets, we need to count something, e.g., occurrences of pairs.
- The number of different things we can count is limited by main memory.
- Swapping counts in/out is a disaster.

Finding Frequent Pairs

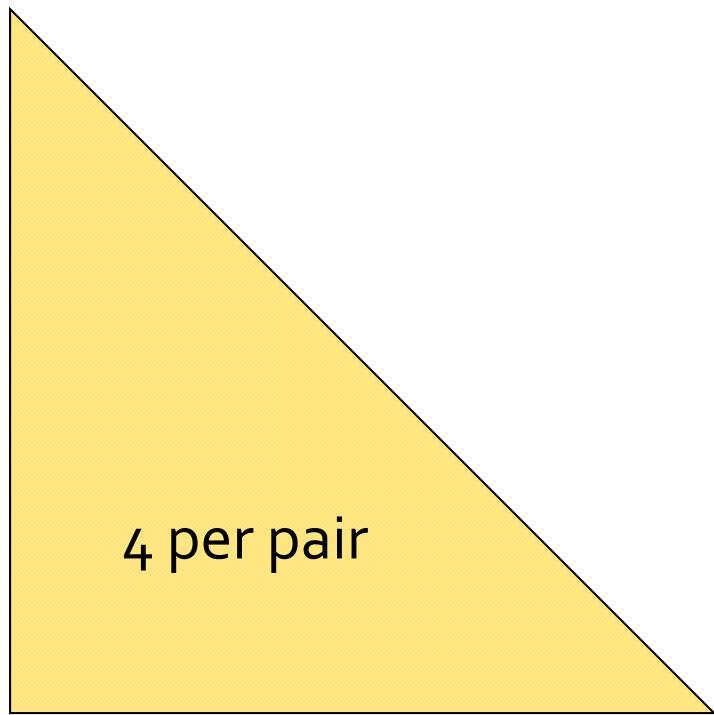
- The hardest problem often turns out to be finding the **frequent pairs**.
 - **Why?** Often frequent pairs are common, frequent triples are rare.
 - **Why?** Support threshold is usually set high enough that you don't get too many frequent itemsets.
- We'll concentrate on pairs, then extend to larger sets.

Naïve Algorithm

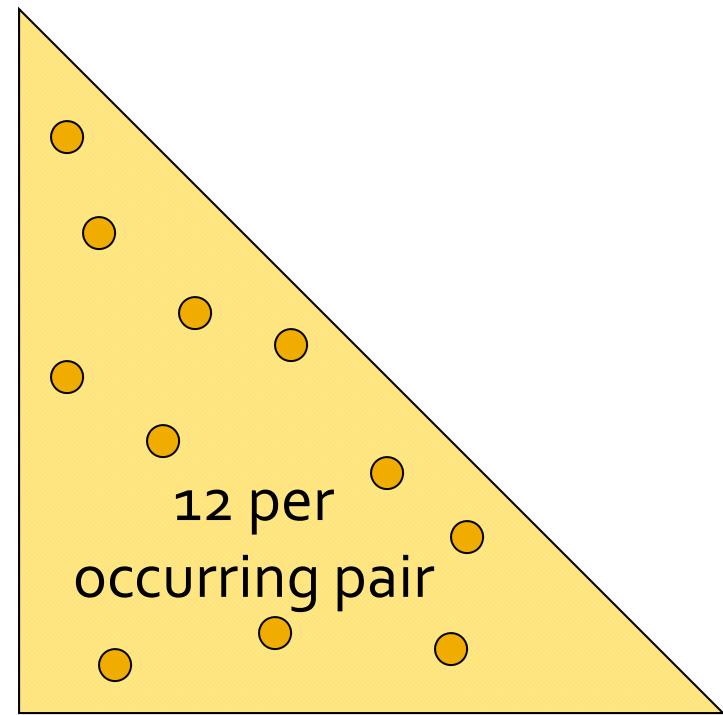
- Read file once, counting in main memory the occurrences of each pair.
 - From each basket of n items, generate its $n(n-1)/2$ pairs by two nested loops.
- Fails if $(\#items)^2$ exceeds main memory.
 - Remember: #items can be 100K (Wal-Mart) or 100B (Web pages).

Details of Main-Memory Counting

- Two approaches:
 1. Count all pairs, using a triangular matrix.
 2. Keep a table of triples $[i, j, c] = \text{"the count of the pair of items } \{i, j\} \text{ is } c\text{"}$
- (1) requires only 4 bytes/pair.
 - Note: always assume integers are 4 bytes.
 - (2) requires 12 bytes, but only for those pairs with count > 0 .



Triangular matrix



Tabular method

Triangular-Matrix Approach

- Number items 1, 2, ...
 - Requires table of size $O(n)$ to convert item names to consecutive integers.
- Count $\{i, j\}$ only if $i < j$.
- Keep pairs in the order $\{1,2\}, \{1,3\}, \dots, \{1,n\}$,
 $\{2,3\}, \{2,4\}, \dots, \{2,n\}$, $\{3,4\}, \dots, \{3,n\}, \dots, \{n-1,n\}$.

Triangular-Matrix Approach – (2)

- Find pair $\{i, j\}$, where $i < j$, at the position:
$$(i - 1)(n - i/2) + j - i$$
- Total number of pairs $n(n - 1)/2$; total bytes about $2n^2$.

Details of Tabular Approach

- Total bytes used is about $12p$, where p is the number of pairs that actually occur.
 - Beats triangular matrix if at most $1/3$ of possible pairs actually occur.
- May require extra space for retrieval structure, e.g., a hash table.

The A-Priori Algorithm

Monotonicity of “Frequent”
Candidate Pairs
Extension to Larger Itemsets

A-Priori Algorithm

- A two-pass approach called *a-priori* limits the need for main memory.
- Key idea: *monotonicity*: if a set of items appears at least s times, so does every subset of s .
- *Contrapositive for pairs*: if item i does not appear in s baskets, then no pair including i can appear in s baskets.

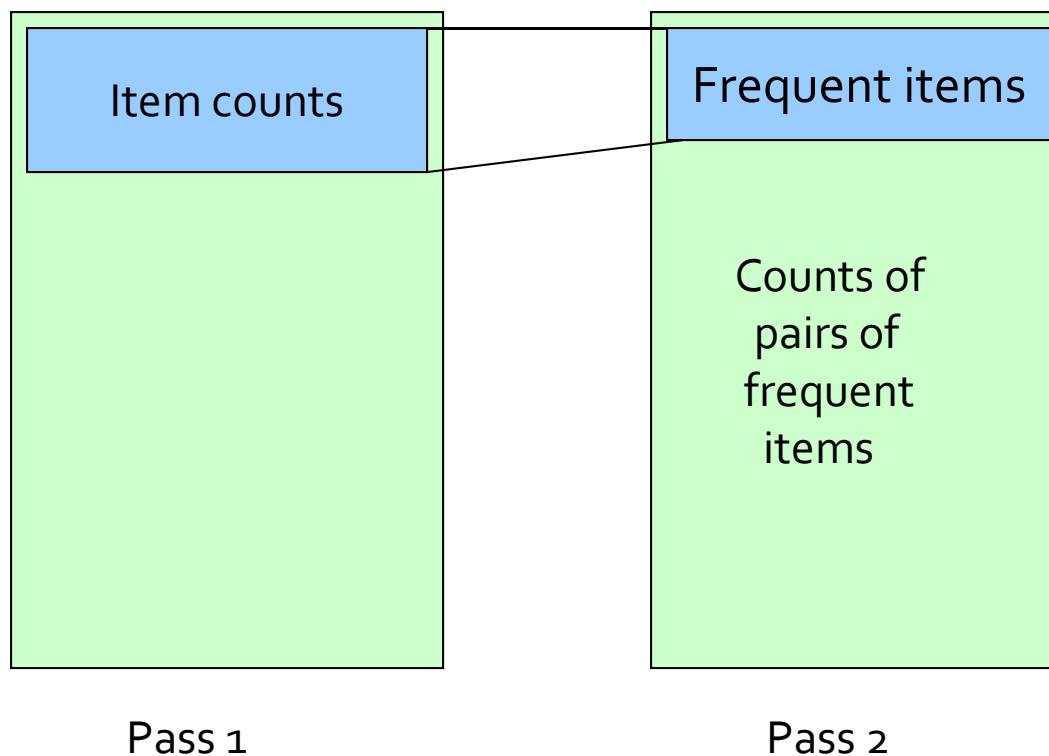
A-Priori Algorithm – (2)

- Pass 1: Read baskets and count in main memory the occurrences of each item.
 - Requires only memory proportional to #items.
- Items that appear at least s times are the *frequent items*.

A-Priori Algorithm – (3)

- **Pass 2:** Read baskets again and count in main memory only those pairs both of which were found in Pass 1 to be frequent.
- Requires memory proportional to square of *frequent* items only (for counts), plus a list of the frequent items (so you know what must be counted).

Picture of A-Priori



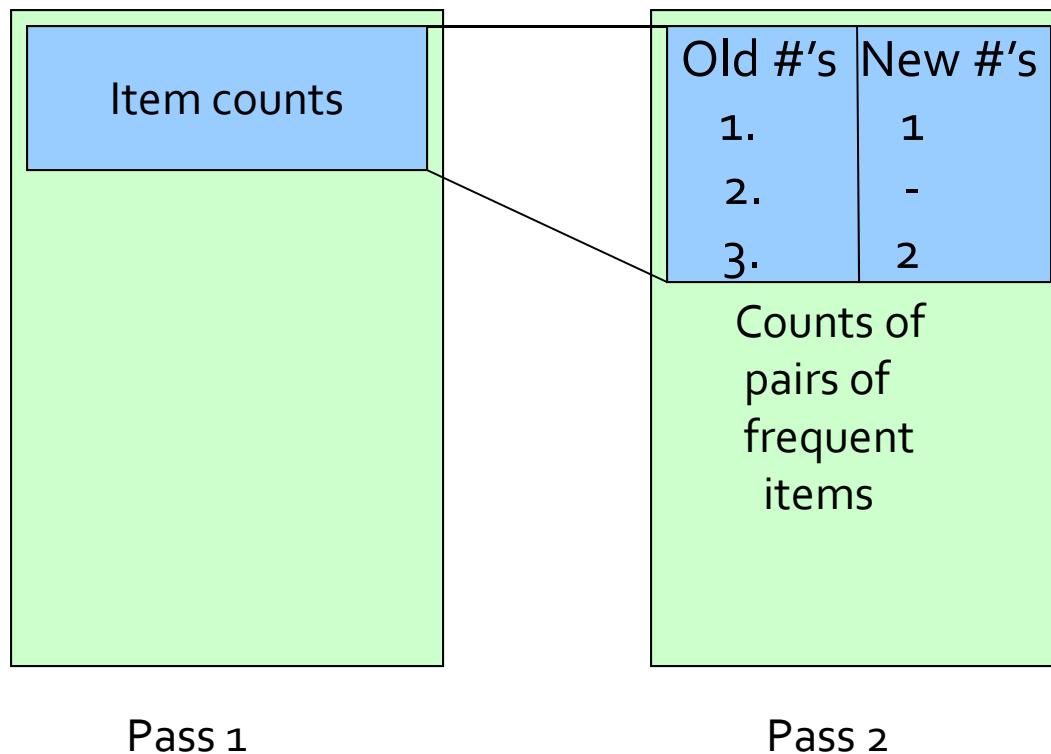
Pass 1

Pass 2

Detail for A-Priori

- You can use the triangular matrix method with $n = \text{number of frequent items}$.
 - May save space compared with storing triples.
- Trick: number frequent items 1,2,... and keep a table relating new numbers to original item numbers.

A-Priori Using Triangular Matrix

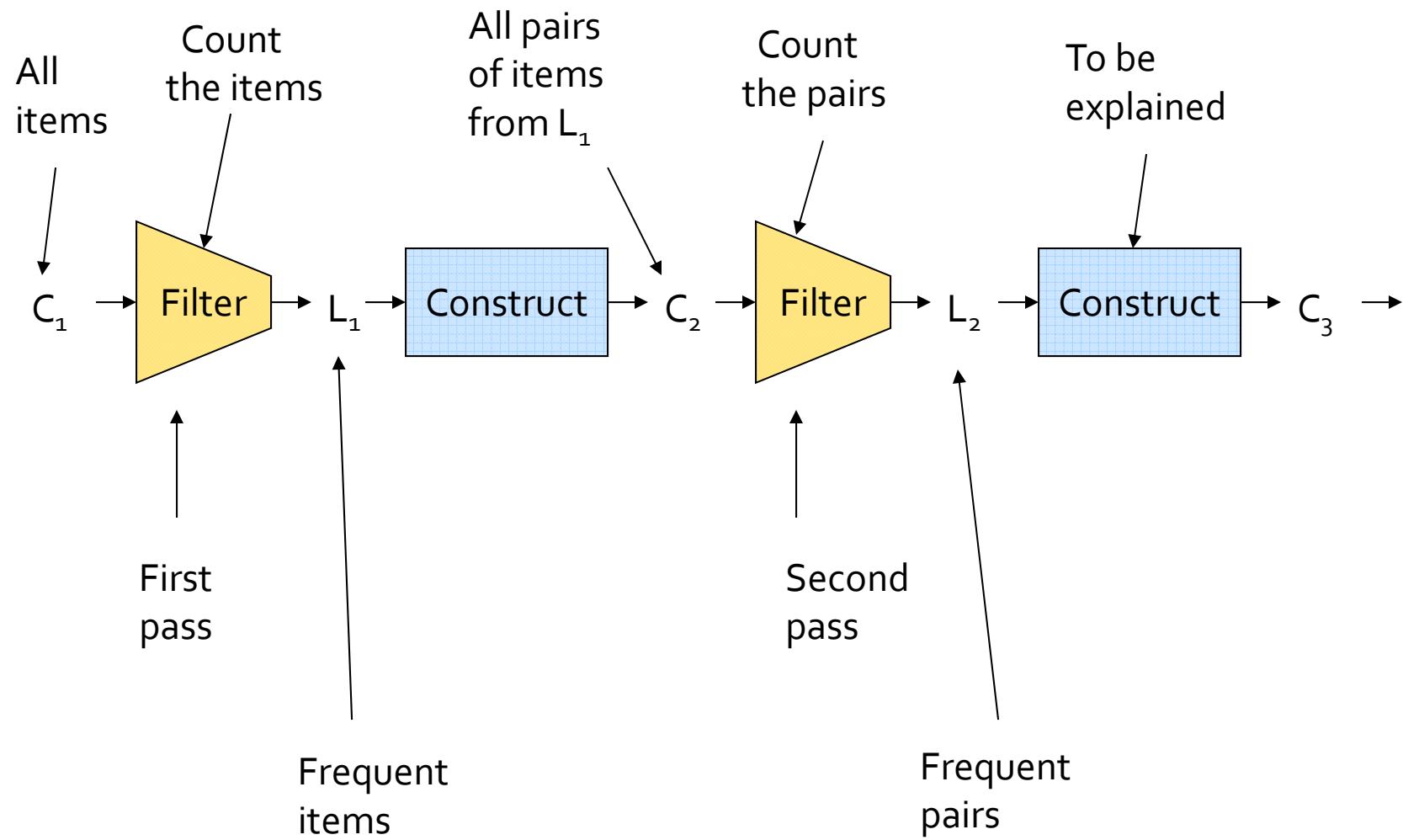


Pass 1

Pass 2

Frequent Triples, Etc.

- For each k , we construct two sets of *k-sets* (sets of size k):
 - C_k = *candidate k-sets* = those that might be frequent sets ($\text{support} \geq s$) based on information from the pass for $k - 1$.
 - L_k = the set of truly frequent k -sets.



Passes Beyond Two

- C_1 = all items
- In general, L_k = members of C_k with support $\geq s$.
 - Requires one pass.
- C_{k+1} = $(k+1)$ -sets, each k of which is in L_k .

Memory Requirements

- At the k^{th} pass, you need space to count each member of C_k .
- In realistic cases, because you need fairly high support, the number of candidates of each size drops, once you get beyond pairs.

Improvements to A-Priori

**Park-Chen-Yu Algorithm
Multistage and Multihash
Single-Pass Approximate Algorithms**

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



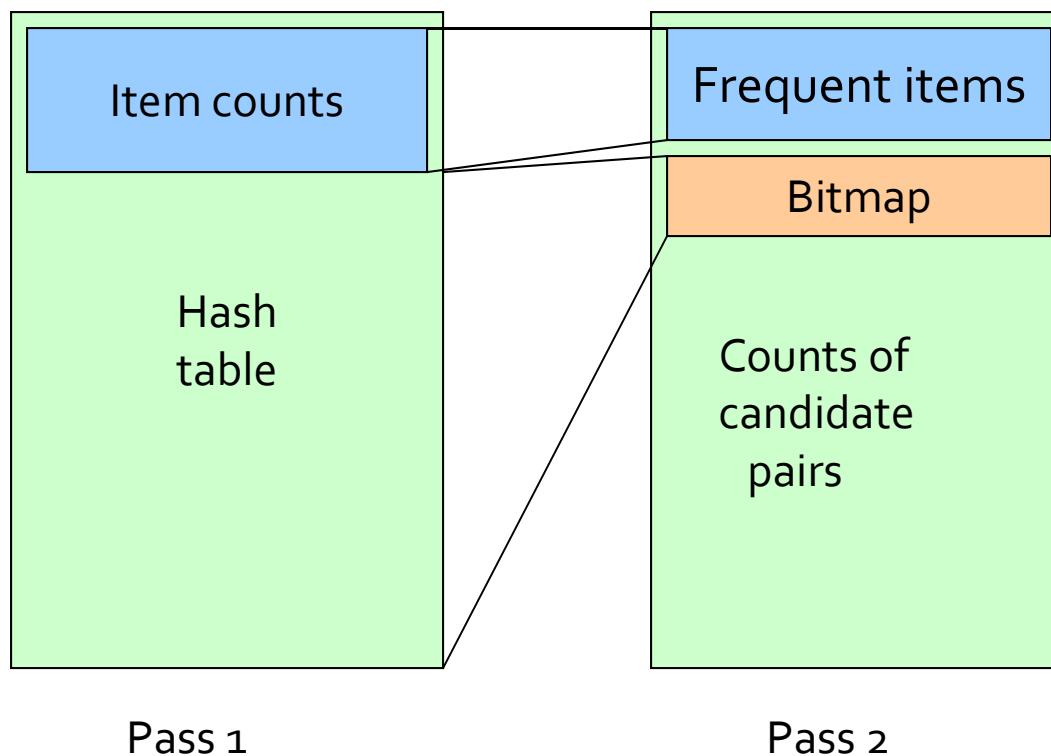
PCY Algorithm

- During Pass 1 of A-priori, most memory is idle.
- Use that memory to keep counts of buckets into which pairs of items are hashed.
 - Just the count, not the pairs themselves.
- For each basket, enumerate all its pairs, hash them, and increment the resulting bucket count by 1.

PCY Algorithm – (2)

- A bucket is *frequent* if its count is at least the support threshold.
- If a bucket is not frequent, no pair that hashes to that bucket could possibly be a frequent pair.
- On Pass 2, we only count pairs that hash to frequent buckets.

Picture of PCY



Pass 1: Memory Organization

- Space to count each item.
 - One (typically) 4-byte integer per item.
- Use the rest of the space for as many integers, representing buckets, as we can.

PCY Algorithm – Pass 1

```
FOR (each basket) {  
    FOR (each item in the basket)  
        add 1 to item's count;  
    FOR (each pair of items) {  
        hash the pair to a bucket;  
        add 1 to the count for that bucket  
    }  
}
```

Observations About Buckets

1. A bucket that a frequent pair hashes to is surely frequent.
 - We cannot use the hash table to eliminate any member of this bucket.
2. Even without any frequent pair, a bucket can be frequent.
 - Again, nothing in the bucket can be eliminated.

Observations – (2)

3. But in the best case, the count for a bucket is less than the support s .
 - Now, all pairs that hash to this bucket can be eliminated as candidates, even if the pair consists of two frequent items.

PCY Algorithm – Between Passes

- Replace the buckets by a bit-vector (the “**bitmap**”):
 - 1 means the bucket is frequent; 0 means it is not.
- 4-byte integers are replaced by bits, so the bit-vector requires 1/32 of memory.
- Also, decide which items are frequent and list them for the second pass.

PCY Algorithm – Pass 2

- Count all pairs $\{i, j\}$ that meet the conditions for being a **candidate pair**:
 1. Both i and j are frequent items.
 2. The pair $\{i, j\}$, hashes to a bucket number whose bit in the bit vector is 1.

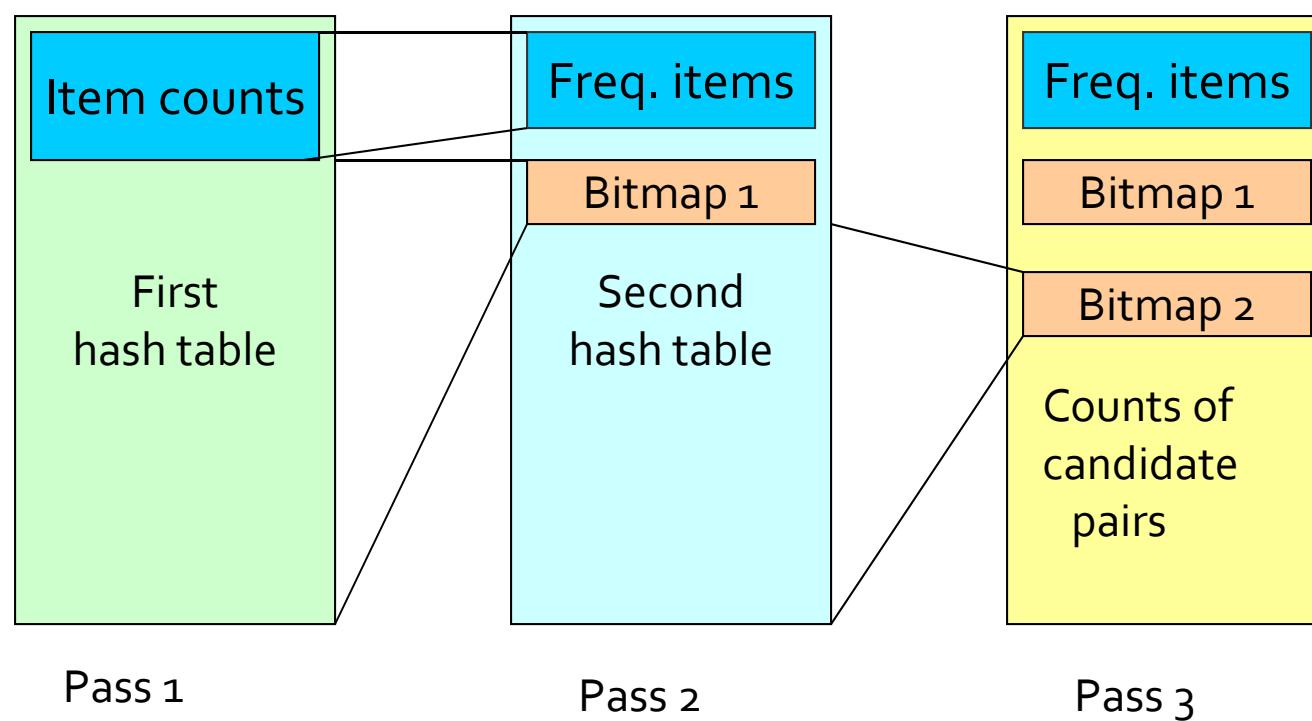
Memory Details

- Buckets require a few bytes each.
 - Note: we don't have to count past s .
 - # buckets is $O(\text{main-memory size})$.
- On second pass, a table of (item, item, count) triples is essential.
 - Thus, hash table must eliminate 2/3 of the candidate pairs for PCY to beat a-priori.

Multistage Algorithm

- **Key idea:** After Pass 1 of PCY, rehash only those pairs that qualify for Pass 2 of PCY.
- On middle pass, fewer pairs contribute to buckets, so fewer *false positives* – frequent buckets with no frequent pair.

Multistage Picture



Multistage – Pass 3

- Count only those pairs $\{i, j\}$ that satisfy these **candidate pair** conditions:
 1. Both i and j are frequent items.
 2. Using the first hash function, the pair hashes to a bucket whose bit in the first bit-vector is 1.
 3. Using the second hash function, the pair hashes to a bucket whose bit in the second bit-vector is 1.

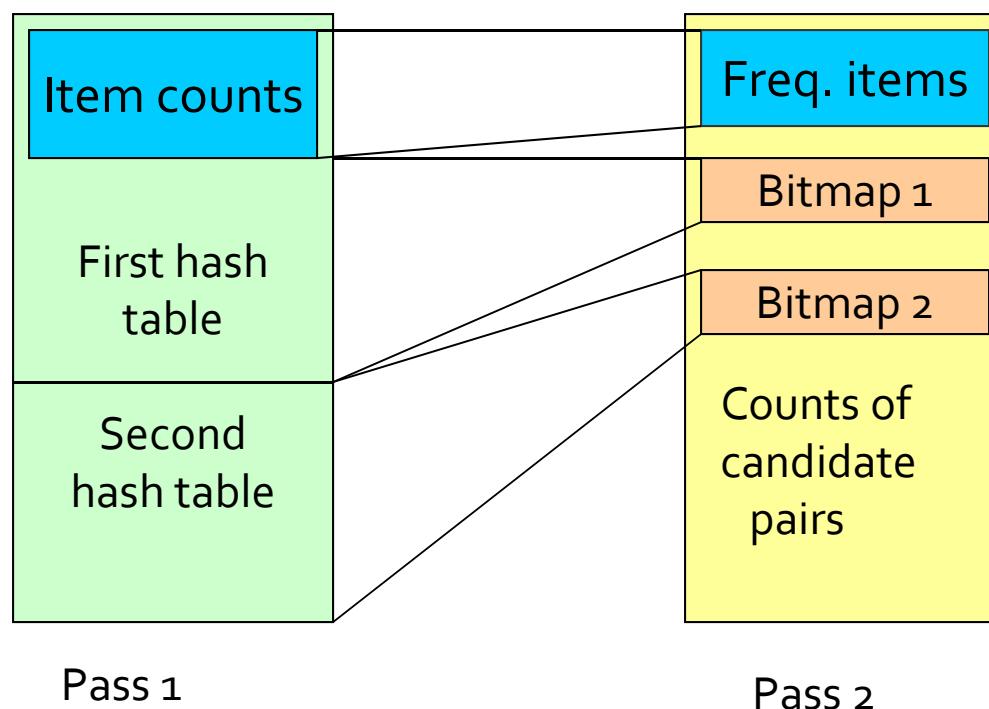
Important Points

1. The hash functions have to be independent.
2. We need to check both hashes on the third pass.
 - If not, we would wind up counting pairs of frequent items that hashed first to an infrequent bucket but happened to hash second to a frequent bucket.

Multihash

- **Key idea:** use several independent hash tables on the first pass.
- **Risk:** halving the number of buckets doubles the average count. We have to be sure most buckets will still not reach count s .
- If so, we can get a benefit like multistage, but in only 2 passes.

Multihash Picture



All (Or Most) Frequent Itemsets In ≤ 2 Passes

Simple Algorithm

Savasere-Omiecinski- Navathe (SON)

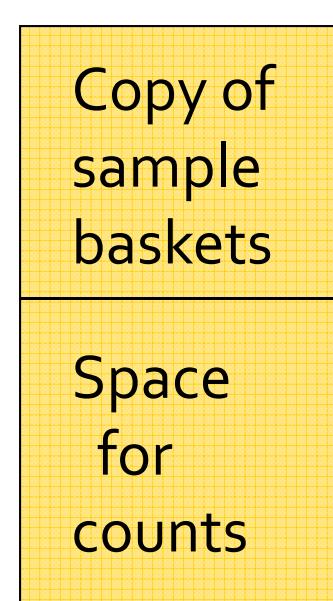
Algorithm

Toivonen's Algorithm

Simple Algorithm

- Take a random sample of the market baskets.
- Run a-priori or one of its improvements (for sets of all sizes, not just pairs) in main memory, so you don't pay for disk I/O each time you increase the size of itemsets.
- Use as your support threshold a suitable, scaled-back number.
 - **Example:** if your sample is $1/100$ of the baskets, use $s/100$ as your support threshold instead of s .

Main-Memory Picture



Simple Algorithm – Option

- Optionally, verify that your guesses are truly frequent in the entire data set by a second pass.
- But you don't catch sets frequent in the whole but not in the sample.
 - Smaller threshold, e.g., $s/125$ instead of $s/100$, helps catch more truly frequent itemsets.
 - But requires more space.

SON Algorithm

- Repeatedly read small subsets of the baskets into main memory and perform the first pass of the simple algorithm on each subset.
- An itemset becomes a candidate if it is found to be frequent in *any* one or more subsets of the baskets.

SON Algorithm – Pass 2

- On a second pass, count all the candidate itemsets and determine which are frequent in the entire set.
- Key “monotonicity” idea: an itemset cannot be frequent in the entire set of baskets unless it is frequent in at least one subset.

SON Algorithm – Distributed Version

- This idea lends itself to distributed data mining.
- If baskets are distributed among many nodes, compute *local* frequent itemsets at each node, then distribute the candidates from each node.
- Each node counts all the candidate itemsets.
- Finally, accumulate the counts of all candidates.

Toivonen's Algorithm

- Start as in the simple algorithm, but lower the threshold slightly for the sample.
 - **Example:** if the sample is 1% of the baskets, use $s/125$ as the support threshold rather than $s/100$.
 - Goal is to avoid missing any itemset that is frequent in the full set of baskets.

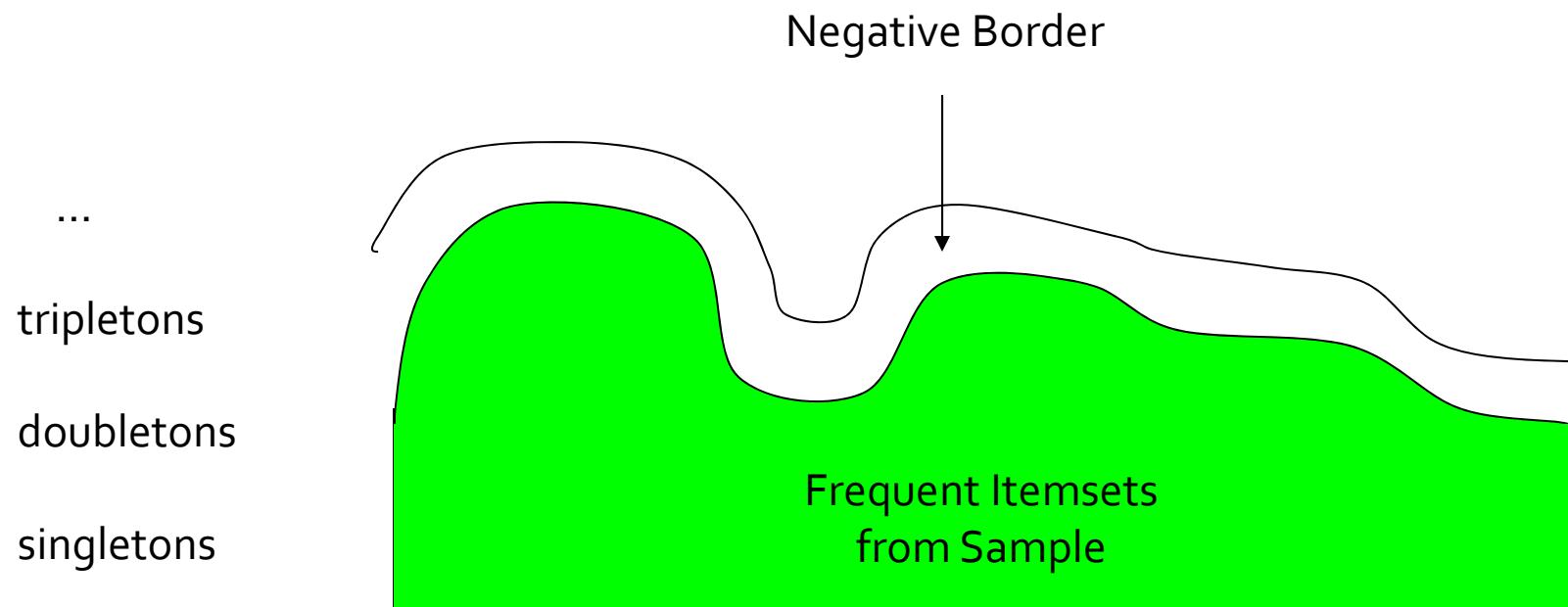
Toivonen's Algorithm – (2)

- Add to the itemsets that are frequent in the sample the *negative border* of these itemsets.
- An itemset is in the negative border if it is not deemed frequent in the sample, but *all* its immediate subsets are.

Example: Negative Border

- $\{A,B,C,D\}$ is in the negative border if and only if:
 1. It is not frequent in the sample, but
 2. All of $\{A,B,C\}$, $\{B,C,D\}$, $\{A,C,D\}$, and $\{A,B,D\}$ are.
- $\{A\}$ is in the negative border if and only if it is not frequent in the sample.
 - Because the empty set is always frequent.
 - Unless there are fewer baskets than the support threshold (silly case).

Picture of Negative Border



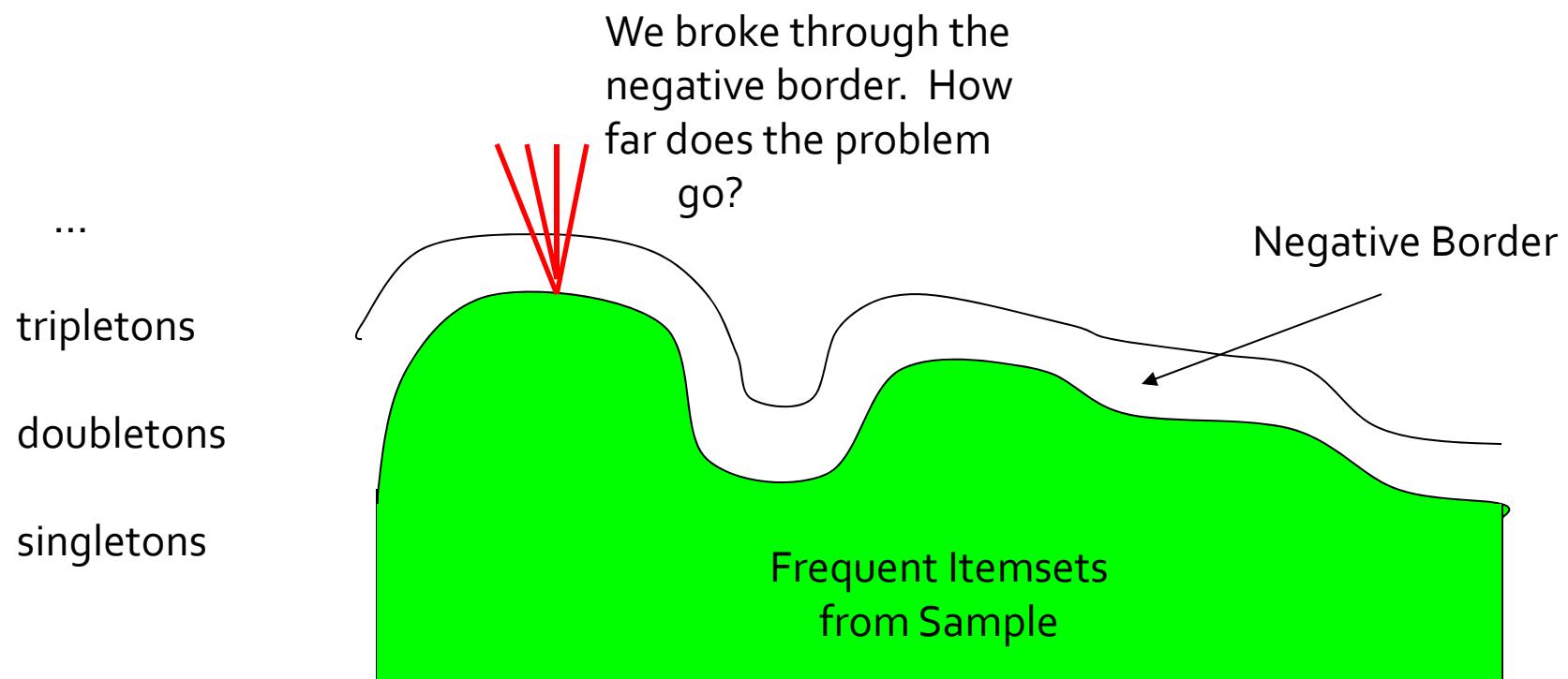
Toivonen's Algorithm – (3)

- In a second pass, count all candidate frequent itemsets from the first pass, and also count their negative border.
- If no itemset from the negative border turns out to be frequent, then the candidates found to be frequent in the whole data are *exactly* the frequent itemsets.

Toivonen's Algorithm – (4)

- What if we find that something in the negative border is actually frequent?
- We must start over again with another sample!
- Try to choose the support threshold so the probability of failure is low, while the number of itemsets checked on the second pass fits in main-memory.

If Something in the Negative Border Is Frequent . . .



Theorem:

- If there is an itemset that is frequent in the whole, but not frequent in the sample, then there is a member of the negative border for the sample that is frequent in the whole.

Proof:

- Suppose not; i.e.;
 1. There is an itemset S frequent in the whole but not frequent in the sample, and
 2. Nothing in the negative border is frequent in the whole.
- Let T be a **smallest** subset of S that is not frequent in the sample.
- T is frequent in the whole (S is frequent + monotonicity).
- T is in the negative border (else not “smallest”).

Hopcroft–Karp algorithm

From Wikipedia, the free encyclopedia

In computer science, the **Hopcroft–Karp algorithm** is an algorithm that takes as input a bipartite graph and produces as output a maximum cardinality matching – a set of as many edges as possible with the property that no two edges share an endpoint. It runs in $O(|E|\sqrt{|V|})$ time in the worst case, where E is set of edges in the graph, and V is set of vertices of the graph. In the case of dense graphs the time bound becomes $O(|V|^{2.5})$, and for random graphs it runs in near-linear time.

The algorithm was found by John Hopcroft and Richard Karp (1973). As in previous methods for matching such as the Hungarian algorithm and the work of Edmonds (1965), the Hopcroft–Karp algorithm repeatedly increases the size of a partial matching by finding augmenting paths. However, instead of finding just a single augmenting path per iteration, the algorithm finds a maximal set of shortest augmenting paths. As a result only $O(\sqrt{n})$ iterations are needed. The same principle has also been used to develop more complicated algorithms for non-bipartite matching with the same asymptotic running time as the Hopcroft–Karp algorithm.

Contents

- 1 Augmenting paths
- 2 Algorithm
- 3 Analysis
- 4 Comparison with other bipartite matching algorithms
- 5 Non-bipartite graphs
- 6 Pseudocode
- 7 Notes
- 8 References

Augmenting paths

A vertex that is not the endpoint of an edge in some partial matching M is called a *free vertex*. The basic concept that the algorithm relies on is that of an *augmenting path*, a path that starts at a free vertex, ends at a free vertex, and alternates between unmatched and matched edges within the path. If M is a matching, and P is an augmenting path relative to M , then the symmetric difference of the two sets of edges, $M \oplus P$, would form a matching with size $|M| + 1$. Thus, by finding augmenting paths, an algorithm may increase the size of the matching.

Conversely, suppose that a matching M is not optimal, and let P be the symmetric difference $M \oplus M^*$ where M^* is an optimal matching. Then P must form a collection of disjoint augmenting paths and cycles or paths in which matched and unmatched edges are of equal number; the difference in size between M and M^* is the number of augmenting paths in P . Thus, if no augmenting path can be found, an algorithm may safely terminate, since in this case M must be optimal.

An augmenting path in a matching problem is closely related to the augmenting paths arising in maximum flow problems, paths along which one may increase the amount of flow between the terminals of the flow. It is possible to transform the bipartite matching problem into a maximum flow instance, such that the alternating paths of the matching problem become augmenting paths of the flow problem.^[1] In fact, a generalization of the technique used in Hopcroft–Karp algorithm to arbitrary flow networks is known as Dinic's algorithm.

Input: Bipartite graph $G(U \cup V, E)$

Output: Matching $M \subseteq E$

$M \leftarrow \emptyset$

repeat

$\mathcal{P} \leftarrow \{P_1, P_2, \dots, P_k\}$ maximal set of vertex-disjoint shortest augmenting paths
 $M \leftarrow M \oplus (P_1 \cup P_2 \cup \dots \cup P_k)$

until $\mathcal{P} = \emptyset$

Algorithm

Let U and V be the two sets in the bipartition of G , and let the matching from U to V at any time be represented as the set M .

The algorithm is run in phases. Each phase consists of the following steps.

- A breadth-first search partitions the vertices of the graph into layers. The free vertices in U are used as the starting vertices of this search, and form the first layer of the partition. At the first level of the search, only unmatched edges may be traversed (since the free vertices in U are by definition not adjacent to any matched edges); at subsequent levels of the search, the traversed edges are required to alternate between matched and unmatched. That is, when searching for successors from a vertex in U , only unmatched edges may be traversed, while from a vertex in V only matched edges may be traversed. The search terminates at the first layer k where one or more free vertices in V are reached.
- All free vertices in V at layer k are collected into a set F . That is, a vertex v is put into F if and only if it ends a shortest augmenting path.
- The algorithm finds a maximal set of *vertex disjoint* augmenting paths of length k . This set may be computed by depth first search from F to the free vertices in U , using the breadth first layering to guide the search: the depth first search is only allowed to follow edges that lead to an unused vertex in the previous layer, and paths in the depth first search tree must alternate between matched and unmatched edges. Once an augmenting path is found that involves one of the vertices in F , the depth first search is continued from the next starting vertex.
- Every one of the paths found in this way is used to enlarge M .

The algorithm terminates when no more augmenting paths are found in the breadth first search part of one of the phases.

Analysis

Each phase consists of a single breadth first search and a single depth first search. Thus, a single phase may be implemented in linear time. Therefore, the first $\sqrt{|V|}$ phases, in a graph with $|V|$ vertices and $|E|$ edges, take time $O(|E|\sqrt{|V|})$.

It can be shown that each phase increases the length of the shortest augmenting path by at least one: the phase finds a maximal set of augmenting paths of the given length, so any remaining augmenting path must be longer. Therefore, once the initial $\sqrt{|V|}$ phases of the algorithm are complete, the shortest remaining augmenting path has at least $\sqrt{|V|}$ edges in it. However, the symmetric difference of the eventual optimal matching and of the partial matching M found by the initial phases forms a collection of vertex-disjoint augmenting paths and alternating cycles. If each of the paths in this collection has length at least $\sqrt{|V|}$, there can be at most $\sqrt{|V|}$ paths in the collection, and the size of the optimal matching can differ from the size of M by at most $\sqrt{|V|}$ edges. Since each phase of the algorithm increases the size of the matching by at least one, there can be at most $\sqrt{|V|}$ additional phases before the algorithm terminates.

Since the algorithm performs a total of at most $2\sqrt{|V|}$ phases, it takes a total time of $O(|E|\sqrt{|V|})$ in the worst case.

In many instances, however, the time taken by the algorithm may be even faster than this worst case analysis indicates. For instance, in the average case for sparse bipartite random graphs, Bast et al. (2006) (improving a previous result of Motwani 1994) showed that with high probability all non-optimal matchings have augmenting paths of logarithmic length. As a consequence, for these graphs, the Hopcroft–Karp algorithm takes $O(\log |V|)$ phases and $O(|E| \log |V|)$ total time.

Comparison with other bipartite matching algorithms

For sparse graphs, the Hopcroft–Karp algorithm continues to have the best known worst-case performance, but for dense graphs a more recent algorithm by Alt et al. (1991) achieves a slightly better time bound, $O\left(n^{1.5}\sqrt{\frac{m}{\log n}}\right)$. Their algorithm is based on using a push-relabel maximum flow algorithm and then, when the matching created by this algorithm becomes close to optimum, switching to the Hopcroft–Karp method.

Several authors have performed experimental comparisons of bipartite matching algorithms. Their results in general tend to show that the Hopcroft–Karp method is not as good in practice as it is in theory: it is outperformed both by simpler breadth-first and depth-first strategies for finding augmenting paths, and by push-relabel techniques.^[2]

Non-bipartite graphs

The same idea of finding a maximal set of shortest augmenting paths works also for finding maximum cardinality matchings in non-bipartite graphs, and for the same reasons the algorithms based on this idea take $O(\sqrt{|V|})$ phases. However, for non-bipartite graphs, the task of finding the augmenting paths within each phase is more difficult. Building on the work of several slower predecessors, Micali & Vazirani (1980) showed how to implement a phase in linear time, resulting in a non-bipartite matching algorithm with the same time bound as the Hopcroft–Karp algorithm for bipartite graphs. The Micali–Vazirani technique is complex, and its authors did not provide full proofs of their results; subsequently, a "clear exposition" was published by Peterson & Loui (1988) and alternative methods were described by other authors.^[3] In 2012, Vazirani offered a new simplified proof of the Micali-Vazirani algorithm.^[4]

Pseudocode

```
/*
G = G1 ∪ G2 ∪ {NIL}
where G1 and G2 are partition of graph and NIL is a special null vertex
*/
function BFS ()
    for v in G1
        if Pair_G1[v] == NIL
            Dist[v] = 0
            Enqueue(Q,v)
        else
            Dist[v] = ∞
    Dist[NIL] = ∞
    while Empty(Q) == false
        v = Dequeue(Q)
        if Dist[v] < Dist[NIL]
            for each u in Adj[v]
                if Dist[ Pair_G2[u] ] == ∞
                    Dist[ Pair_G2[u] ] = Dist[v] + 1
                    Enqueue(Q,Pair_G2[u])
    return Dist[NIL] != ∞

function DFS (v)
    if v != NIL
        for each u in Adj[v]
            if Dist[ Pair_G2[u] ] == Dist[v] + 1
                if DFS(Pair_G2[u]) == true
                    Pair_G2[u] = v
                    Pair_G1[v] = u
                    return true
            Dist[v] = ∞
            return false
    return true

function Hopcroft-Karp
    for each v in G
        Pair_G1[v] = NIL
        Pair_G2[v] = NIL
    matching = 0
    while BFS() == true
        for each v in G1
            if Pair_G1[v] == NIL
                if DFS(v) == true
                    matching = matching + 1
    return matching
```

Notes

1. ^ Ahuja, Magnanti & Orlin (1993), section 12.3, bipartite cardinality matching problem, pp. 469–470.
2. ^ Chang & McCormick (1990); Darby-Dowman (1980); Setubal (1993); Setubal (1996).

3. ^ Gabow & Tarjan (1989) and Blum (2001).
4. ^ Vazirani (2012)

References

- Ahuja, Ravindra K.; Magnanti, Thomas L.; Orlin, James B. (1993), *Network Flows: Theory, Algorithms and Applications*, Prentice-Hall.
- Alt, H.; Blum, N.; Mehlhorn, K.; Paul, M. (1991), *Computing a maximum cardinality matching in a bipartite graph in time $O\left(n^{1.5} \sqrt{\frac{m}{\log n}}\right)$* , *Information Processing Letters* **37** (4): 237–240, doi:10.1016/0020-0190(91)90195-N (<http://dx.doi.org/10.1016%2F0020-0190%2891%2990195-N>).
- Bast, Holger; Mehlhorn, Kurt; Schafer, Guido; Tamaki, Hisao (2006), *Matching algorithms are fast in sparse random graphs*, *Theory of Computing Systems* **39** (1): 3–14, doi:10.1007/s00224-005-1254-y (<http://dx.doi.org/10.1007%2Fs00224-005-1254-y>).
- Blum, Norbert (2001), *A Simplified Realization of the Hopcroft-Karp Approach to Maximum Matching in General Graphs* (<http://theory.cs.uni-bonn.de/ftp/reports/cs-reports/2001/85232-CS.ps.gz>), Tech. Rep. 85232-CS, Computer Science Department, Univ. of Bonn.
- Chang, S. Frank; McCormick, S. Thomas (1990), *A faster implementation of a bipartite cardinality matching algorithm*, Tech. Rep. 90-MSC-005, Faculty of Commerce and Business Administration, Univ. of British Columbia. As cited by Setubal (1996).
- Darby-Dowman, Kenneth (1980), *The exploitation of sparsity in large scale linear programming problems – Data structures and restructuring algorithms*, Ph.D. thesis, Brunel University. As cited by Setubal (1996).
- Edmonds, Jack (1965), *Paths, Trees and Flowers*, *Canadian J. Math* **17**: 449–467, doi:10.4153/CJM-1965-045-4 (<http://dx.doi.org/10.4153%2FCJM-1965-045-4>), MR 0177907 (<https://www.ams.org/mathscinet-getitem?mr=0177907>).
- Gabow, Harold N.; Tarjan, Robert E. (1991), *Faster scaling algorithms for general graph matching problems*, *Journal of the ACM* **38** (4): 815–853, doi:10.1145/115234.115366 (<http://dx.doi.org/10.1145%2F115234.115366>).
- Hopcroft, John E.; Karp, Richard M. (1973), *An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs*, *SIAM Journal on Computing* **2** (4): 225–231, doi:10.1137/0202019 (<http://dx.doi.org/10.1137%2F0202019>).
- Micali, S.; Vazirani, V. V. (1980), "An $O(\sqrt{|V||E|})$ algorithm for finding maximum matching in general graphs", *Proc. 21st IEEE Symp. Foundations of Computer Science*, pp. 17–27, doi:10.1109/SFCS.1980.12 (<http://dx.doi.org/10.1109%2FSFCS.1980.12>).
- Peterson, Paul A.; Loui, Michael C. (1988), *The general maximum matching algorithm of Micali and Vazirani*, *Algorithmica* **3** (1-4): 511–533, doi:10.1007/BF01762129 (<http://dx.doi.org/10.1007%2FBF01762129>).
- Motwani, Rajeev (1994), *Average-case analysis of algorithms for matchings and related*

- problems, *Journal of the ACM* **41** (6): 1329–1356, doi:10.1145/195613.195663 (<http://dx.doi.org/10.1145%2F195613.195663>).
- Setubal, João C. (1993), "New experimental results for bipartite matching", *Proc. Netflow93*, Dept. of Informatics, Univ. of Pisa, pp. 211–216. As cited by Setubal (1996).
 - Setubal, João C. (1996), *Sequential and parallel experimental results with bipartite matching algorithms* (<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.48.3539>), Tech. Rep. IC-96-09, Inst. of Computing, Univ. of Campinas.
 - Vazirani, Vijay (2012), *An Improved Definition of Blossoms and a Simpler Proof of the MV Matching Algorithm* (<http://arxiv.org/abs/1210.4594>), CoRR abs/1210.4594.

Retrieved from "http://en.wikipedia.org/w/index.php?title=Hopcroft-Karp_algorithm&oldid=593898016"

Categories: Graph algorithms | Matching

-
- This page was last modified on 4 February 2014 at 15:28.
 - Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

Finding Similar Sets

Applications

Shingling

Minhashing

Locality-Sensitive Hashing

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Applications of Set-Similarity

Many data-mining problems can be expressed as finding “similar” sets:

1. Pages with similar words, e.g., for classification by topic.
2. NetFlix users with similar tastes in movies, for recommendation systems.
3. Dual: movies with similar sets of fans.
4. Entity resolution.

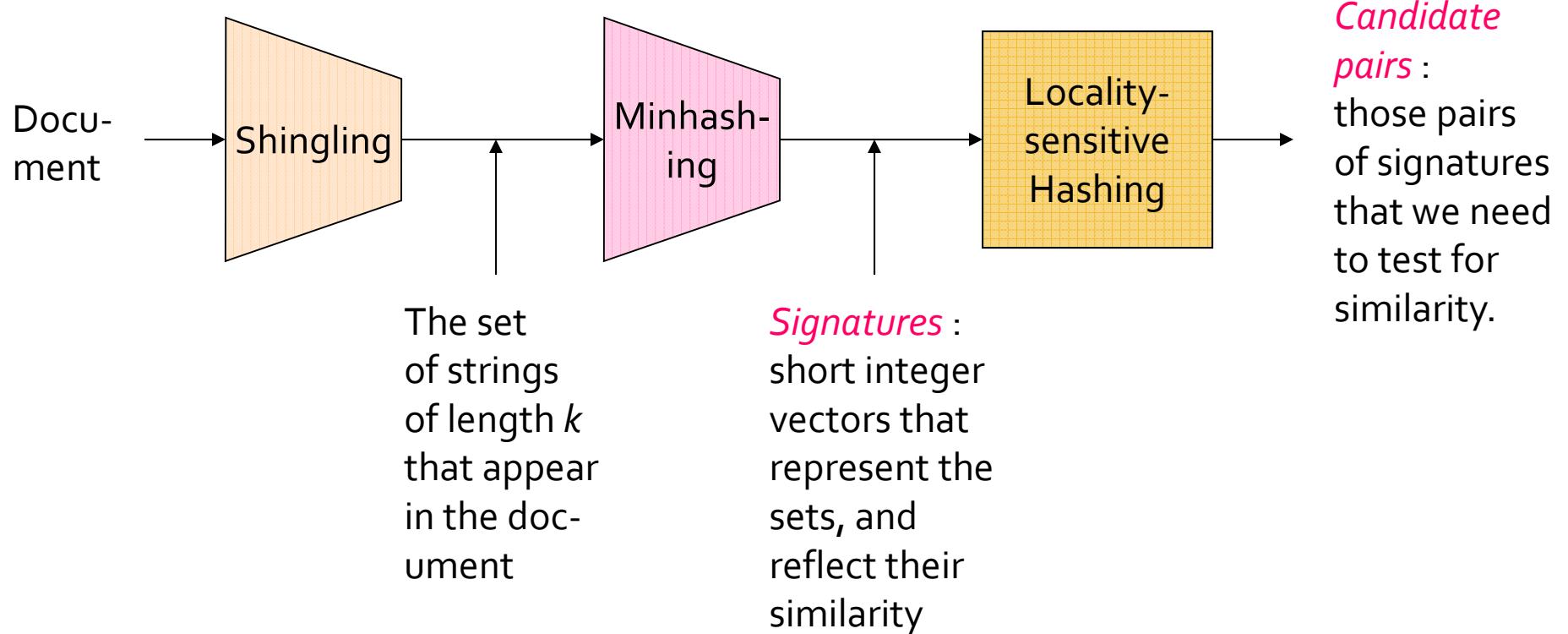
Similar Documents

- Given a body of documents, e.g., the Web, find pairs of documents with a lot of text in common, such as:
 - Mirror sites, or approximate mirrors.
 - Application: Don't want to show both in a search.
 - Plagiarism, including large quotations.
 - Similar news articles at many news sites.
 - Application: Cluster articles by "same story."

Three Essential Techniques for Similar Documents

1. *Shingling* : convert documents, emails, etc., to sets.
2. *Minhashing* : convert large sets to short signatures, while preserving similarity.
3. *Locality-sensitive hashing* : focus on pairs of signatures likely to be similar.

The Big Picture



Shingles

- A k -shingle (or k -gram) for a document is a sequence of k characters that appears in the document.
- Example: $k=2$; doc = abcab. Set of 2-shingles = {ab, bc, ca}.
- Represent a doc by its set of k -shingles.

Shingles and Similarity

- Documents that are intuitively similar will have many shingles in common.
- Changing a word only affects k-shingles within distance k from the word.
- Reordering paragraphs only affects the 2k shingles that cross paragraph boundaries.
- Example: k=3, “The dog which chased the cat” versus “The dog that chased the cat”.
 - Only 3-shingles replaced are g_w, _wh, whi, hic, ich, ch_, and h_c.

Shingles: Compression Option

- To compress long shingles, we can hash them to (say) 4 bytes.
 - Called *tokens*.
- Represent a doc by its tokens, that is, the set of hash values of its k -shingles.
- Two documents could (rarely) appear to have shingles in common, when in fact only the hash-values were shared.

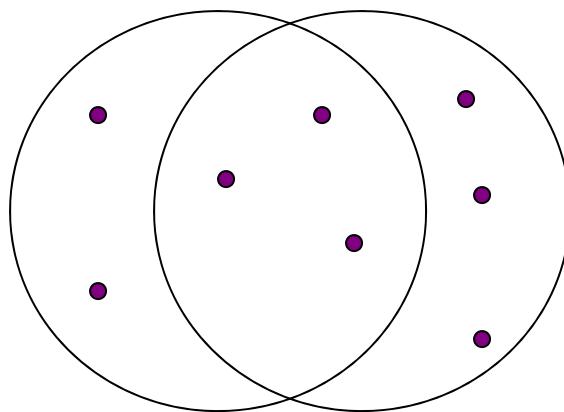
Minhashing

Jaccard Similarity Measure
Constructing Signatures

Jaccard Similarity

- The *Jaccard similarity* of two sets is the size of their intersection divided by the size of their union.
- $Sim(C_1, C_2) = |C_1 \cap C_2| / |C_1 \cup C_2|$.

Example: Jaccard Similarity



3 in intersection.
8 in union.
Jaccard similarity
 $= 3/8$

From Sets to Boolean Matrices

- **Rows** = elements of the universal set.
 - **Example:** the set of all k-shingles.
- **Columns** = sets.
- 1 in row e and column S if and only if e is a member of S .
- Column similarity is the Jaccard similarity of the sets of their rows with 1.
- Typical matrix is sparse.

Example: Column Similarity

C₁ C₂

0 1 *

1 0 *

1 1 * * Sim(C₁, C₂) =
0 0 2/5 = 0.4

1 1 * *

0 1 *

Four Types of Rows

- Given columns C_1 and C_2 , rows may be classified as:

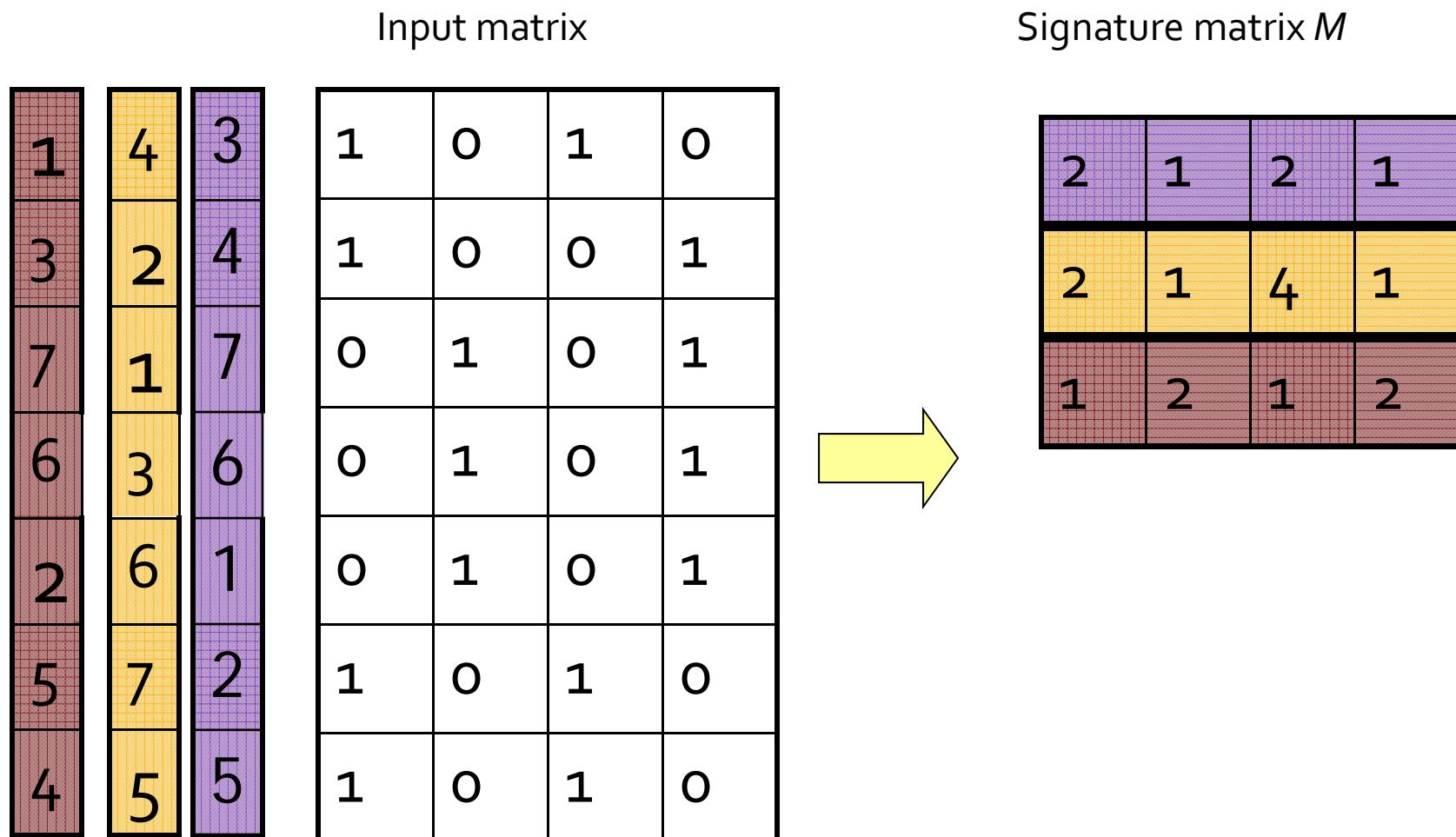
	<u>C_1</u>	<u>C_2</u>
a	1	1
b	1	0
c	0	1
d	0	0

- Also, $a = \# \text{ rows of type } a$, etc.
- Note $\text{Sim}(C_1, C_2) = a/(a + b + c)$.

Minhashing

- Imagine the rows permuted randomly.
- Define *minhash function* $h(C)$ = the number of the first (in the permuted order) row in which column C has 1.
- Use several (e.g., 100) independent hash functions to create a signature for each column.
- The signatures can be displayed in another matrix – the *signature matrix* – whose columns represent the sets and the rows represent the minhash values, in order for that column.

Minhashing Example



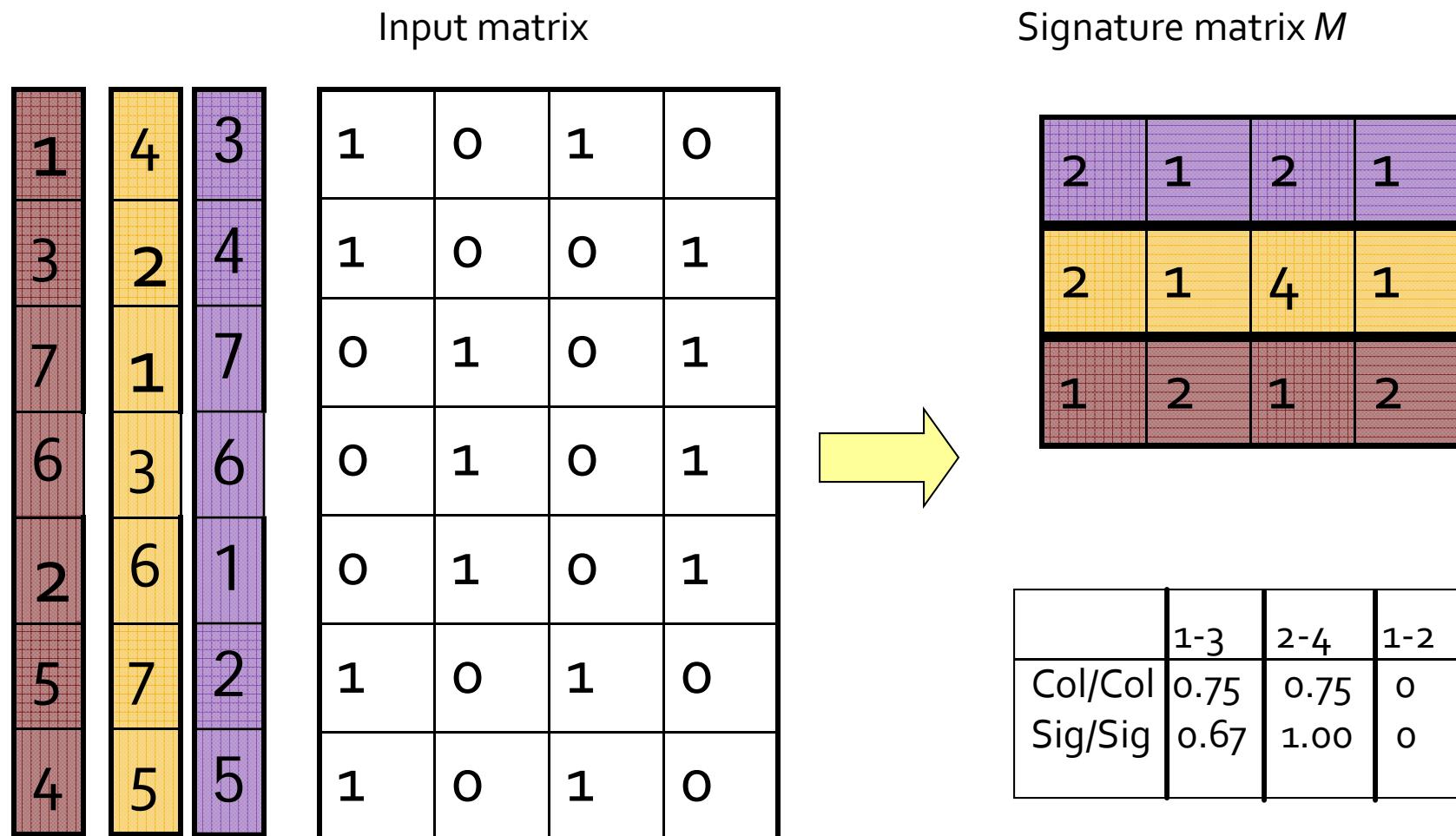
Surprising Property

- The probability (over all permutations of the rows) that $h(C_1) = h(C_2)$ is the same as $\text{Sim}(C_1, C_2)$.
- Both are $a / (a + b + c)!$
- Why?
 - Look down the permuted columns C_1 and C_2 until we see a 1.
 - If it's a type- a row, then $h(C_1) = h(C_2)$. If a type- b or type- c row, then not.

Similarity for Signatures

- The *similarity of signatures* is the fraction of the minhash functions in which they agree.
 - Thinking of signatures as columns of integers, the similarity of signatures is the fraction of rows in which they agree.
- Thus, the expected similarity of two signatures equals the Jaccard similarity of the columns or sets that the signatures represent.
 - And the longer the signatures, the smaller will be the expected error.

Min Hashing – Example



Implementation of Minhashing

- Suppose 1 billion rows.
- Hard to pick a random permutation of 1...billion.
- Representing a random permutation requires 1 billion entries.
- Accessing rows in permuted order leads to thrashing.

Implementation – (2)

- A good approximation to permuting rows: pick, say, 100 hash functions.
- For each column c and each hash function h_i , keep a “slot” $M(i, c)$.
- Intent: $M(i, c)$ will become the smallest value of $h_i(r)$ for which column c has 1 in row r .
 - I.e., $h_i(r)$ gives order of rows for i^{th} permutation.

Implementation – (3)

```
for each row  $r$  do begin
    for each hash function  $h_i$  do
        compute  $h_i(r)$ ;
    for each column  $c$ 
        if  $c$  has 1 in row  $r$ 
            for each hash function  $h_i$  do
                if  $h_i(r)$  is smaller than  $M(i, c)$  then
                     $M(i, c) := h_i(r)$ ;
end;
```

Example

Row	C_1	C_2	$h(1) = 1$	$g(1) = 3$	Sig_1	Sig_2
1	1	0	$h(2) = 2$	$g(2) = 0$	1	∞
2	0	1			3	∞
3	1	1				
4	1	0	$h(3) = 3$	$g(3) = 2$	1	2
5	0	1			2	0
			$h(4) = 4$	$g(4) = 4$	1	2
					2	0
			$h(5) = 0$	$g(5) = 1$	1	∞
					2	0

$h(x) = x \bmod 5$
 $g(x) = (2x+1) \bmod 5$

Implementation – (4)

- Often, data is given by column, not row.
 - Example: columns = documents, rows = shingles.
- If so, sort matrix once so it is by row.

Locality-Sensitive Hashing

Focusing on Similar Minhash Signatures
Other Applications Will Follow

Locality-Sensitive Hashing

- General idea: Generate from the collection of all elements (signatures in our example) a small list of *candidate pairs*: pairs of elements whose similarity must be evaluated.
- For signature matrices: Hash columns to many buckets, and make elements of the same bucket candidate pairs.

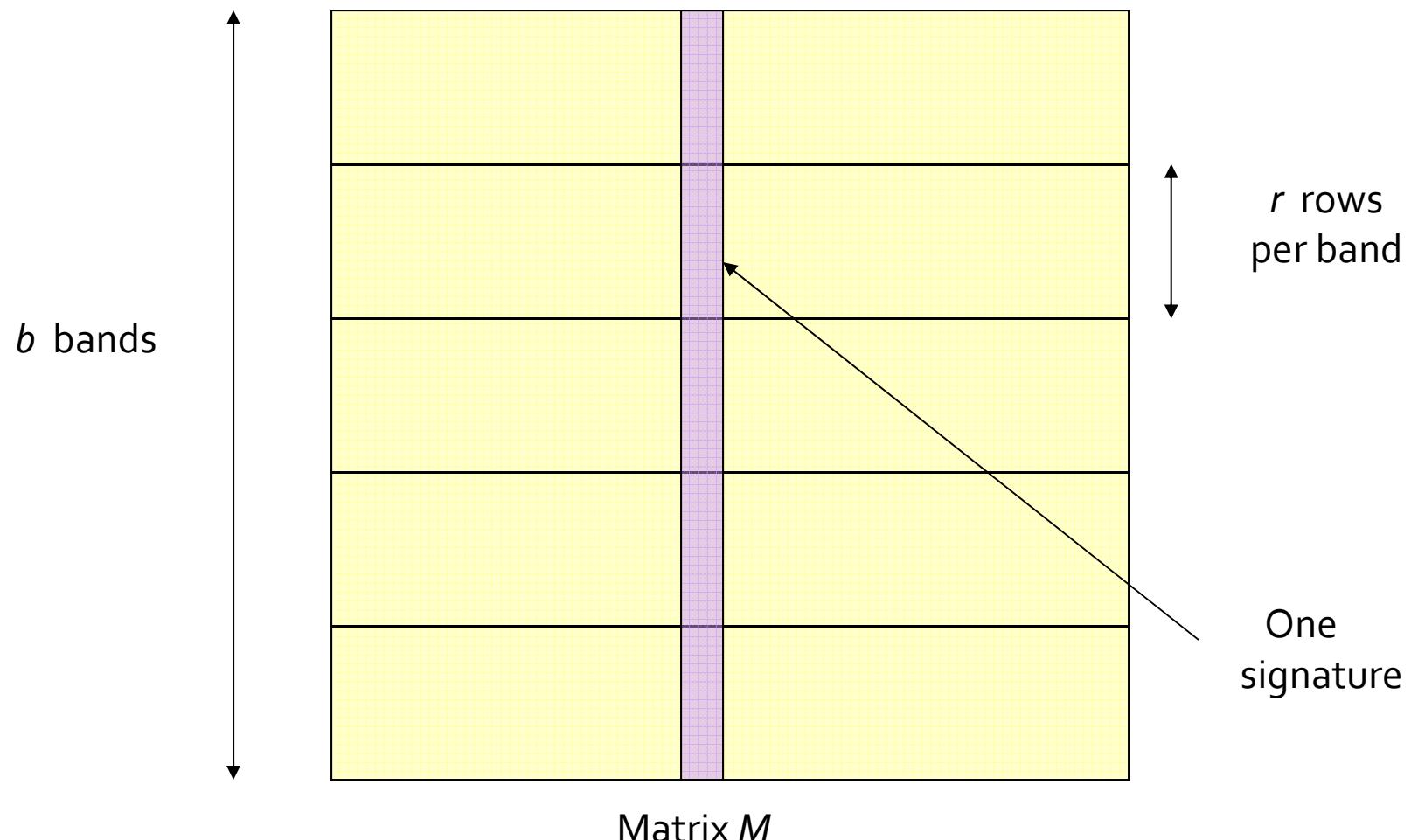
Candidate Generation From Minhash Signatures

- Pick a similarity threshold t , a fraction < 1 .
- We want a pair of columns c and d of the signature matrix M to be a *candidate pair* if and only if their signatures agree in at least fraction t of the rows.
 - I.e., $M(i, c) = M(i, d)$ for at least fraction t values of i .

LSH for Minhash Signatures

- Big idea: hash columns of signature matrix M several times.
- Arrange that (only) similar columns are likely to hash to the same bucket.
- Candidate pairs are those that hash at least once to the same bucket.

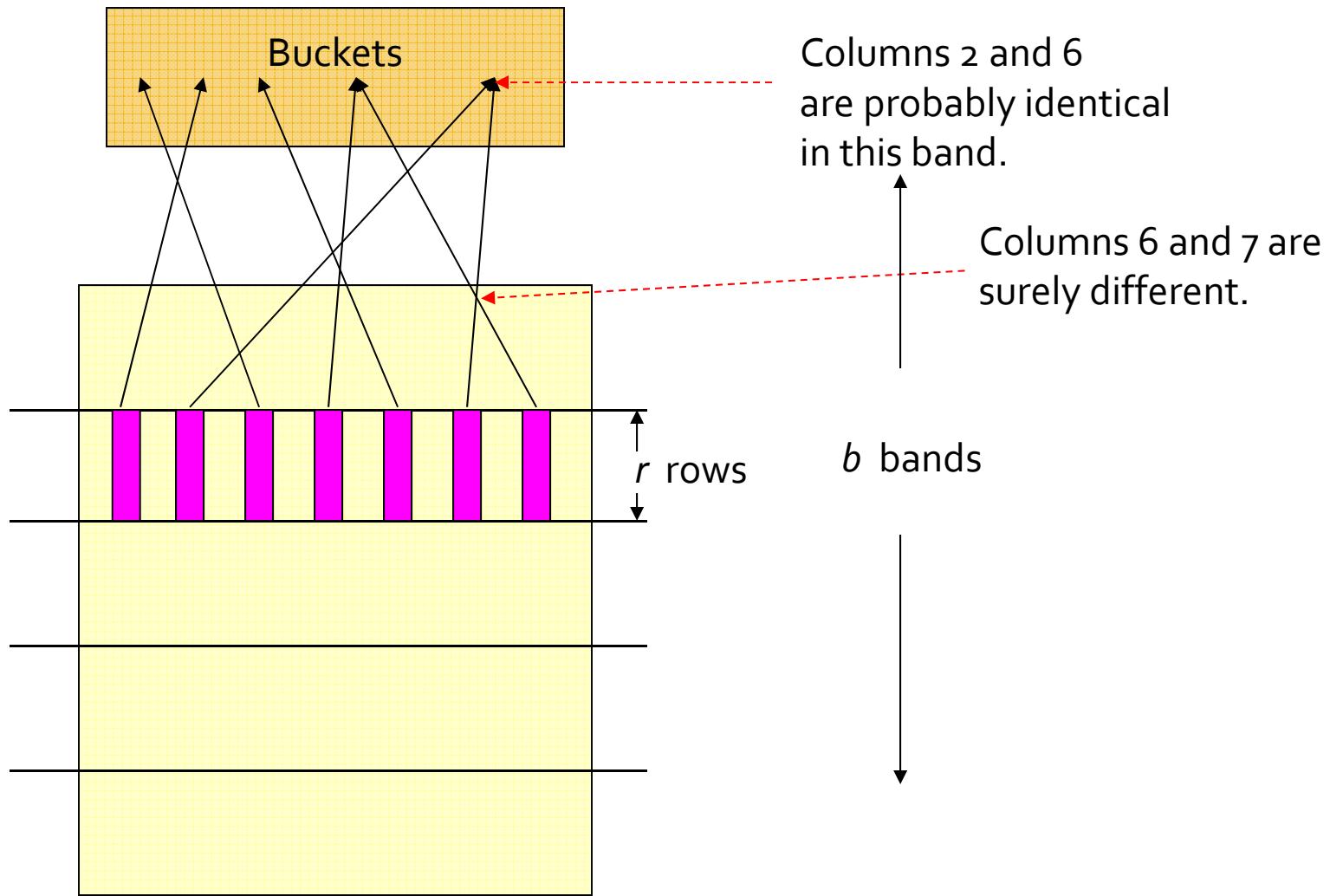
Partition Into Bands



Partition into Bands – (2)

- Divide matrix M into b bands of r rows.
- For each band, hash its portion of each column to a hash table with k buckets.
 - Make k as large as possible.
- *Candidate* column pairs are those that hash to the same bucket for ≥ 1 band.
- Tune b and r to catch most similar pairs, but few nonsimilar pairs.

Hash Function for One Bucket



Example – Bands

- Suppose 100,000 columns.
- Signatures of 100 integers.
- Therefore, signatures take 40Mb.
- Want all 80%-similar pairs of documents.
- 5,000,000,000 pairs of signatures can take a while to compare.
- Choose 20 bands of 5 integers/band.

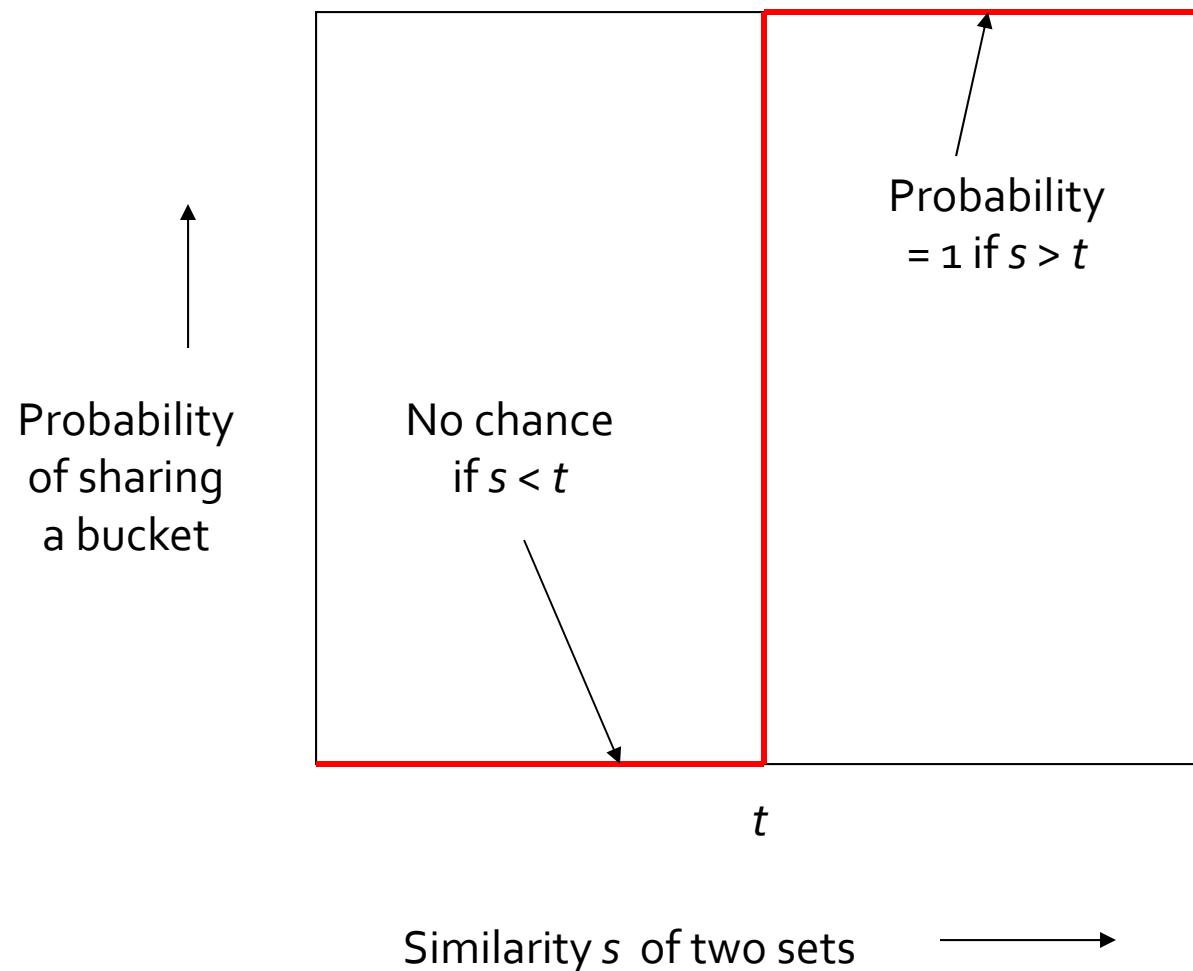
Suppose C_1 , C_2 are 80% Similar

- Probability C_1 , C_2 identical in one particular band: $(0.8)^5 = 0.328$.
- Probability C_1 , C_2 are *not* similar in any of the 20 bands: $(1-0.328)^{20} = .00035$.
 - i.e., about 1/3000th of the 80%-similar underlying sets are false negatives.

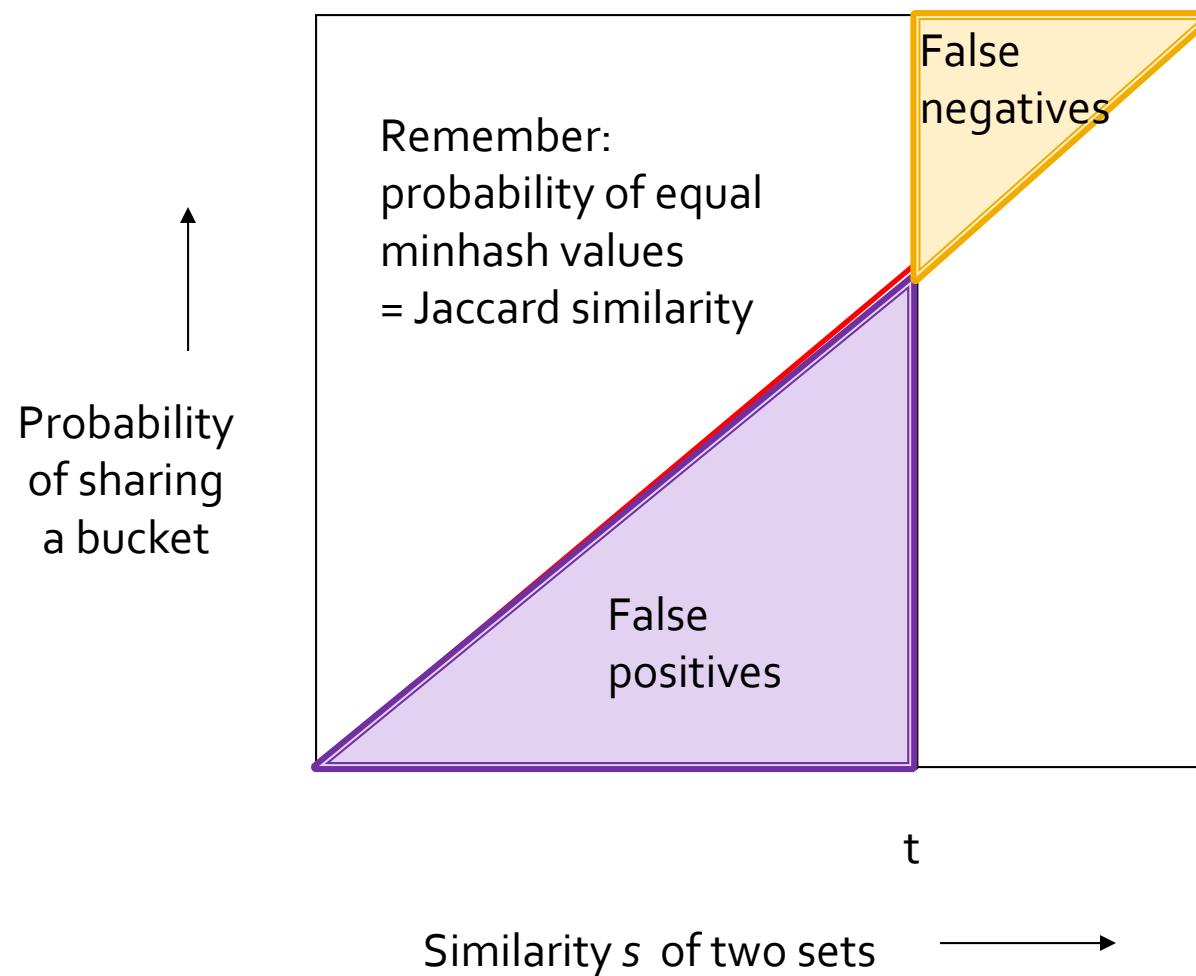
Suppose C_1, C_2 Only 40% Similar

- Probability C_1, C_2 identical in any one particular band: $(0.4)^5 = 0.01$.
- Probability C_1, C_2 identical in ≥ 1 of 20 bands:
 $\leq 20 * 0.01 = 0.2$.
- But false positives much lower for similarities
 $<< 40\%$.

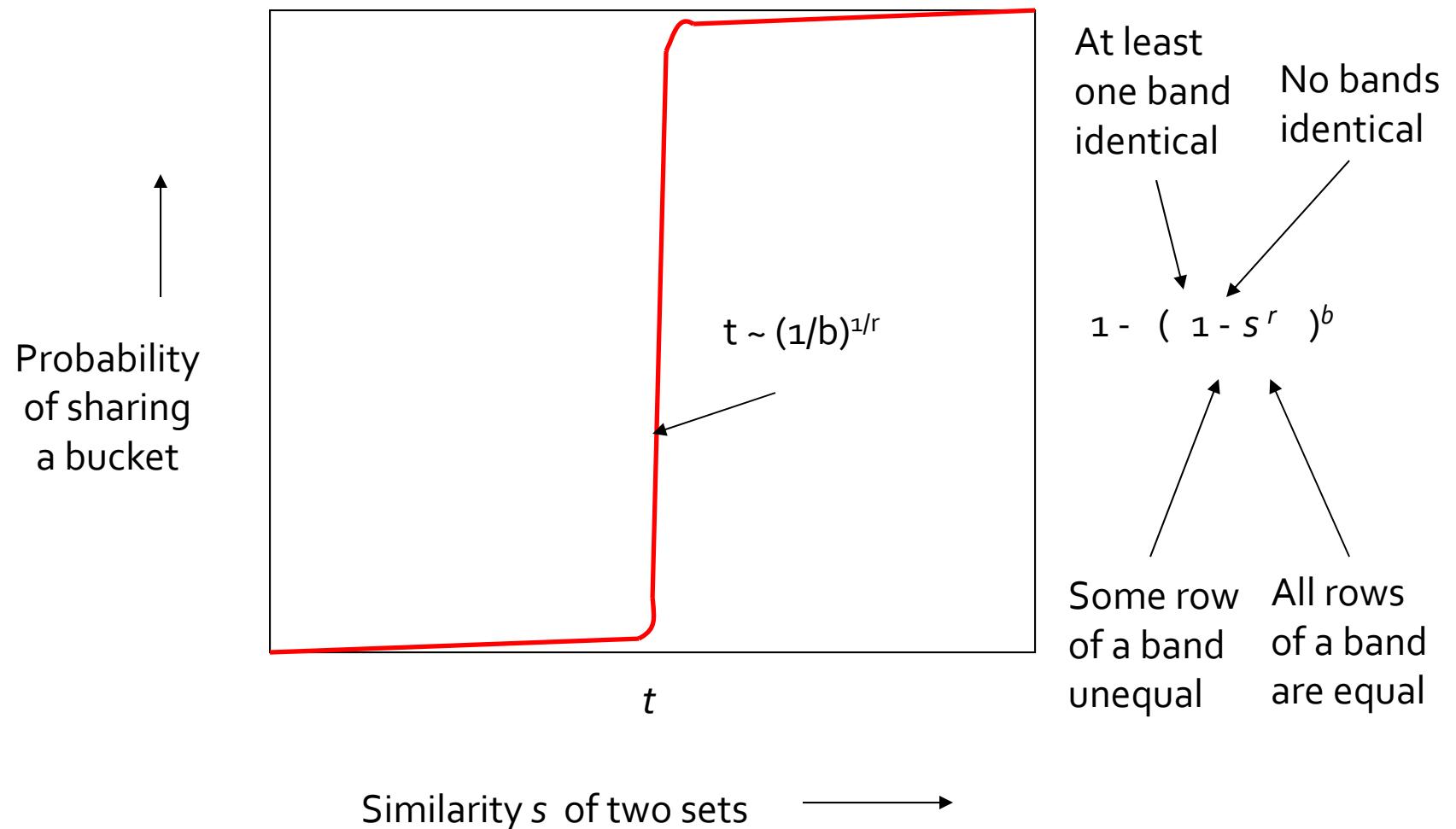
Analysis of LSH – What We Want



What One Band of One Row Gives You



What b Bands of r Rows Gives You



Example: $b = 20$; $r = 5$

s	$1 - (1 - s^r)^b$
.2	.006
.3	.047
.4	.186
.5	.470
.6	.802
.7	.975
.8	.9996

LSH Summary

- Tune to get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures.
- Check that candidate pairs really do have similar signatures.
- **Optional:** In another pass through data, check that the remaining candidate pairs really represent similar sets .

Applications of LSH

Entity Resolution
Fingerprints
Similar News Articles

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Entity Resolution

- The *entity-resolution* problem is to examine a collection of records and determine which refer to the same entity.
 - *Entities* could be people, events, etc.
- Typically, we want to merge records if their values in corresponding fields are similar.

Matching Customer Records

- I once took a consulting job solving the following problem:
 - Company A agreed to solicit customers for Company B, for a fee.
 - They then argued over how many customers.
 - Neither recorded exactly which customers were involved.

Customer Records – (2)

- Each company had about 1 million records describing customers that might have been sent from A to B.
- Records had name, address, and phone, but for various reasons, they could be different for the same person.

Customer Records – (3)

- Step 1: Design a measure (“*score*”) of how similar records are:
 - E.g., deduct points for small misspellings (“Jeffrey” vs. “Jeffery”) or same phone with different area code.
- Step 2: Score all pairs of records that the LSH scheme identified as candidates; report high scores as matches.

Customer Records – (4)

- **Problem:** $(1 \text{ million})^2$ is too many pairs of records to score.
- **Solution:** A simple LSH.
 - Three hash functions: exact values of name, address, phone.
 - Compare iff records are identical in at least one.
 - Misses similar records with a small differences in all three fields.

Aside: Hashing Names, Etc.

- How do we hash strings such as names so there is one bucket for each string?
- **Answer:** Sort the strings instead.
- Another option was to use a few million buckets, and deal with buckets that contain several different strings.

Aside: Validation of Results

- We were able to tell what values of the scoring function were reliable in an interesting way.
- Identical records had a creation date difference of 10 days.
- We only looked for records created within 90 days of each other, so bogus matches had a 45-day average.

Validation – (2)

- By looking at the pool of matches with a fixed score, we could compute the average time-difference, say x , and deduce that fraction $(45-x)/35$ of them were valid matches.
- Alas, the lawyers didn't think the jury would understand.

Validation – Generalized

- Any field not used in the LSH could have been used to validate, provided corresponding values were closer for true matches than false.
- Example: if records had a height field, we would expect true matches to be close, false matches to have the average difference for random people.

Fingerprint Matching

Minutiae
A New Way of Bucketing

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



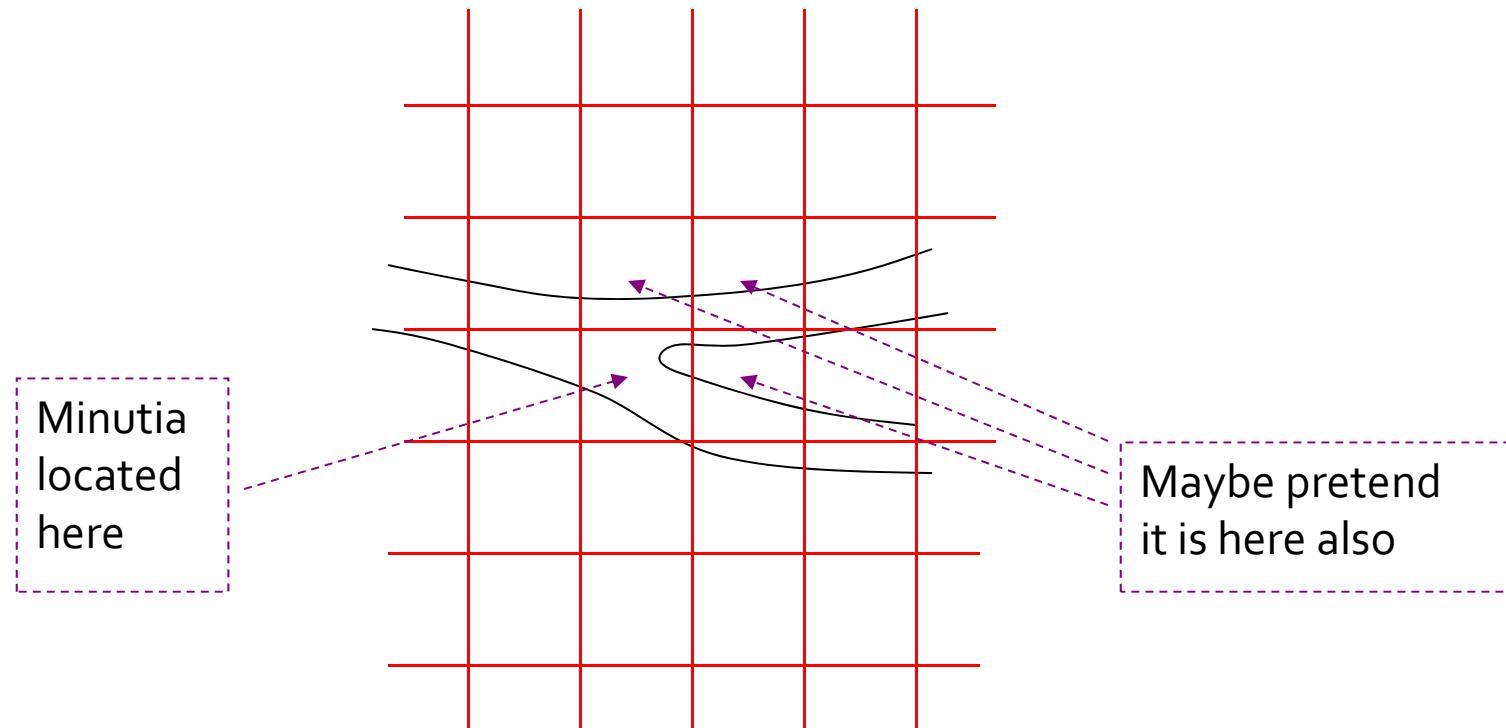
Fingerprint Comparison

- Represent a fingerprint by the set of positions of *minutiae*.
 - These are features of a fingerprint, e.g., points where two ridges come together or a ridge ends.

LSH for Fingerprints

- Place a grid on a fingerprint.
 - Normalize so identical prints will overlap.
- Set of grid squares where minutiae are located represents the fingerprint.
- Possibly, treat minutiae near a grid boundary as if also present in adjacent grid points.

Discretizing Minutiae



Applying LSH to Fingerprints

- Fingerprint = set of grid squares.
- No need to minhash, since the number of grid squares is not too large.
- Represent each fingerprint by a bit-vector with one position for each square.
 - 1 in only those positions whose squares have minutiae.

LSH/Fingerprints – (2)

- Pick 1024 (?) sets of 3 (?) grid squares (components of the bit vectors), randomly.
- For each set of three squares, two prints that each have 1 for all three squares are candidate pairs.
- Funny sort of ‘bucketization.’
 - Each set of three squares creates one bucket.
 - Prints can be in many buckets.

Example: LSH/Fingerprints

- Suppose typical fingerprints have minutiae in 20% of the grid squares.
- Suppose fingerprints from the same finger agree in at least 80% of their squares.
- Probability two random fingerprints each have minutiae in all three squares = $(0.2)^6 = .000064$.

Example: Continued

First print has
has minutia in
this square

Second print of the
same finger also has
minutia in that square

- Probability two fingerprints from the same finger each have 1's in three given squares =
 $((0.2)(0.8))^3 = .004096.$
- Prob. for at least one of 1024 sets of three points = $1-(1-.004096)^{1024} = .985.$
- But for random fingerprints:
 $1-(1-.000064)^{1024} = .063.$

1.5% false
negatives

6.3% false
positives

Finding Duplicate News Articles

A New Way of Shingling
Bucketing by Length

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Application: Same News Article

- The Political-Science Dept. at Stanford asked a team from CS to help them with the problem of identifying duplicate, on-line news articles.
- **Problem:** the same article, say from the Associated Press, appears on the Web site of many newspapers, but looks quite different.

News Articles – (2)

- Each newspaper surrounds the text of the article with:
 - Its own logo and text.
 - Ads.
 - Perhaps links to other articles.
- A newspaper may also “crop” the article (delete parts).

News Articles – (3)

- The team came up with its own solution, that included shingling, but not minhashing or LSH.
 - A special way of shingling that appears quite good for **this** application.
 - **LSH substitute**: candidates are articles of similar length.

Enter LSH

- I told them the story of minhashing + LSH.
- They implemented it and found it faster for similarities below 80%.
 - **Aside:** That's no surprise. When similarity is high, there are better methods, as we shall see.

Enter LSH – (2)

- Their first attempt at minhashing was very inefficient.
- They were unaware of the importance of doing the minhashing row-by-row.
- Since their data was column-by-column, they needed to sort once before minhashing.

Specialized Shingling Technique

- The team observed that news articles have a lot of *stop words*, while ads do not.
 - “Buy Sudzo” vs. “I recommend **that you** buy Sudzo **for your** laundry.”
- They defined a *shingle* to be a stop word and the next two following words.

Why it Works

- By requiring each shingle to have a stop word, they biased the mapping from documents to shingles so it picked more shingles from the article than from the ads.
- Pages with the same article, but different ads, have higher Jaccard similarity than those with the same ads, different articles.

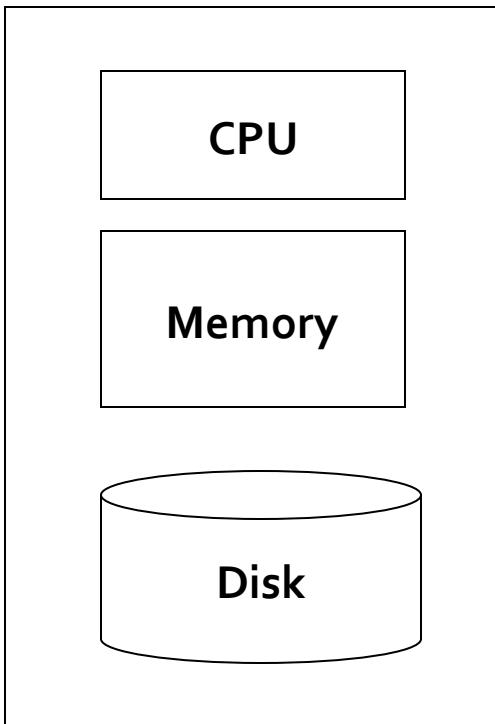
Map-Reduce

Distributed File System
Computational Model
Scheduling and Data Flow
Refinements

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Single Node Architecture



Machine Learning, Statistics

“Classical” Data Mining

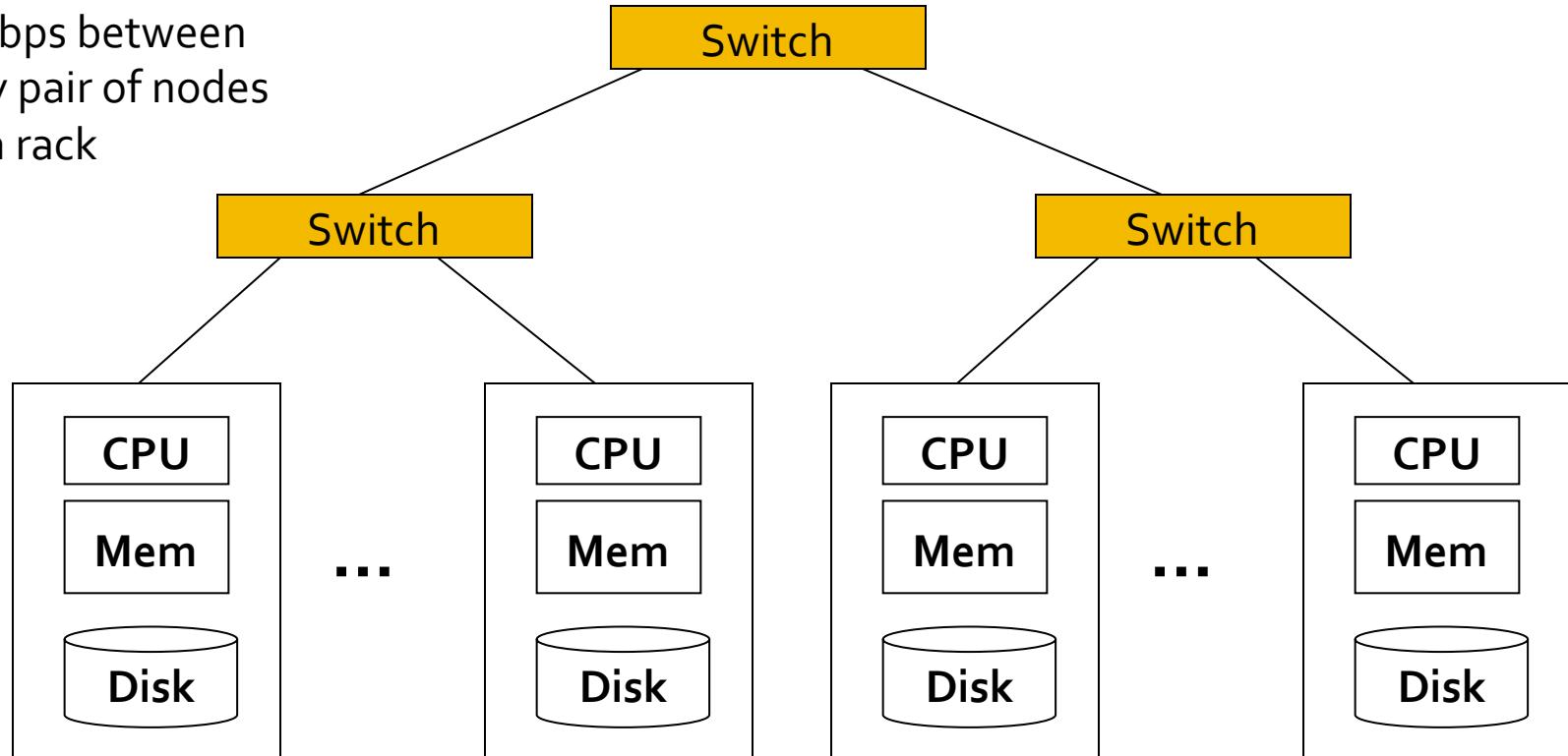
Motivation: Google Example

- 10 billion web pages
- Average size of webpage = 20KB
- $10 \text{ billion} * 20\text{KB} = 200 \text{ TB}$
- Disk read bandwidth = 50 MB/sec
- Time to read = 4 million seconds = 46+ days
- Even longer to do something useful with the data

Cluster Architecture

1 Gbps between
any pair of nodes
in a rack

2-10 Gbps backbone between racks



Each rack contains 16-64 commodity Linux nodes

In 2011 it was guestimated that Google had 1M machines, <http://bit.ly/Shh0RO>



Cluster Computing Challenges (1)

- Node failures
 - A single server can stay up for 3 years (1000 days)
 - 1000 servers in cluster => 1 failure/day
 - 1M servers in cluster => 1000 failures/day
- How to store data persistently and keep it available if nodes can fail?
- How to deal with node failures during a long-running computation?

Cluster Computing Challenges (2)

- Network bottleneck
 - Network bandwidth = 1 Gbps
 - Moving 10TB takes approximately 1 day
- Distributed programming is hard!
 - Need a simple model that hides most of the complexity

Map-Reduce

- Map-Reduce addresses the challenges of cluster computing
 - Store data redundantly on multiple nodes for persistence and availability
 - Move computation close to data to minimize data movement
 - Simple programming model to hide the complexity of all this magic

Redundant Storage Infrastructure

■ **Distributed File System**

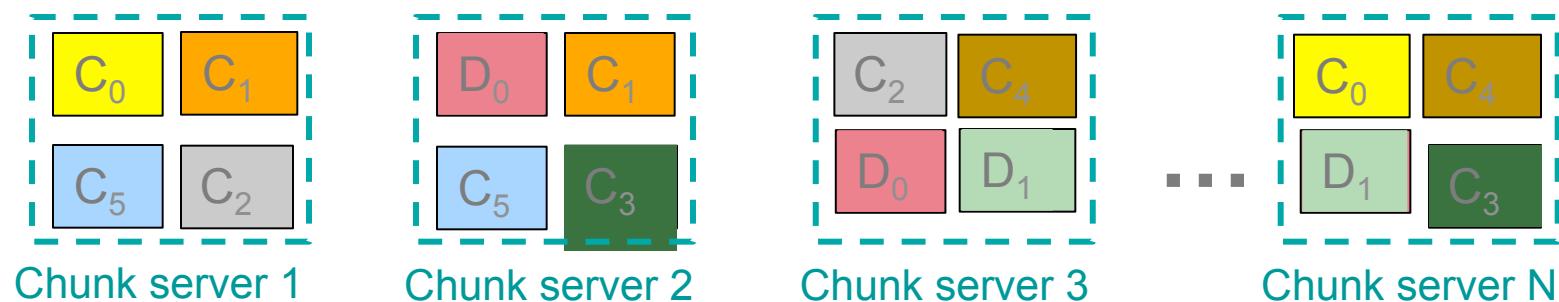
- Provides global file namespace, redundancy, and availability
- E.g., Google GFS; Hadoop HDFS

■ **Typical usage pattern**

- Huge files (100s of GB to TB)
- Data is rarely updated in place
- Reads and appends are common

Distributed File System

- Data kept in “chunks” spread across machines
- Each chunk **replicated** on different machines
 - Ensures persistence and availability



Chunk servers also serve as compute servers

Bring computation to data!

Distributed File System

- **Chunk servers**

- File is split into contiguous chunks (16-64MB)
- Each chunk replicated (usually 2x or 3x)
- Try to keep replicas in different racks

- **Master node**

- a.k.a. Name Node in Hadoop's HDFS
- Stores metadata about where files are stored
- Might be replicated

- **Client library for file access**

- Talks to master to find chunk servers
- Connects directly to chunk servers to access data

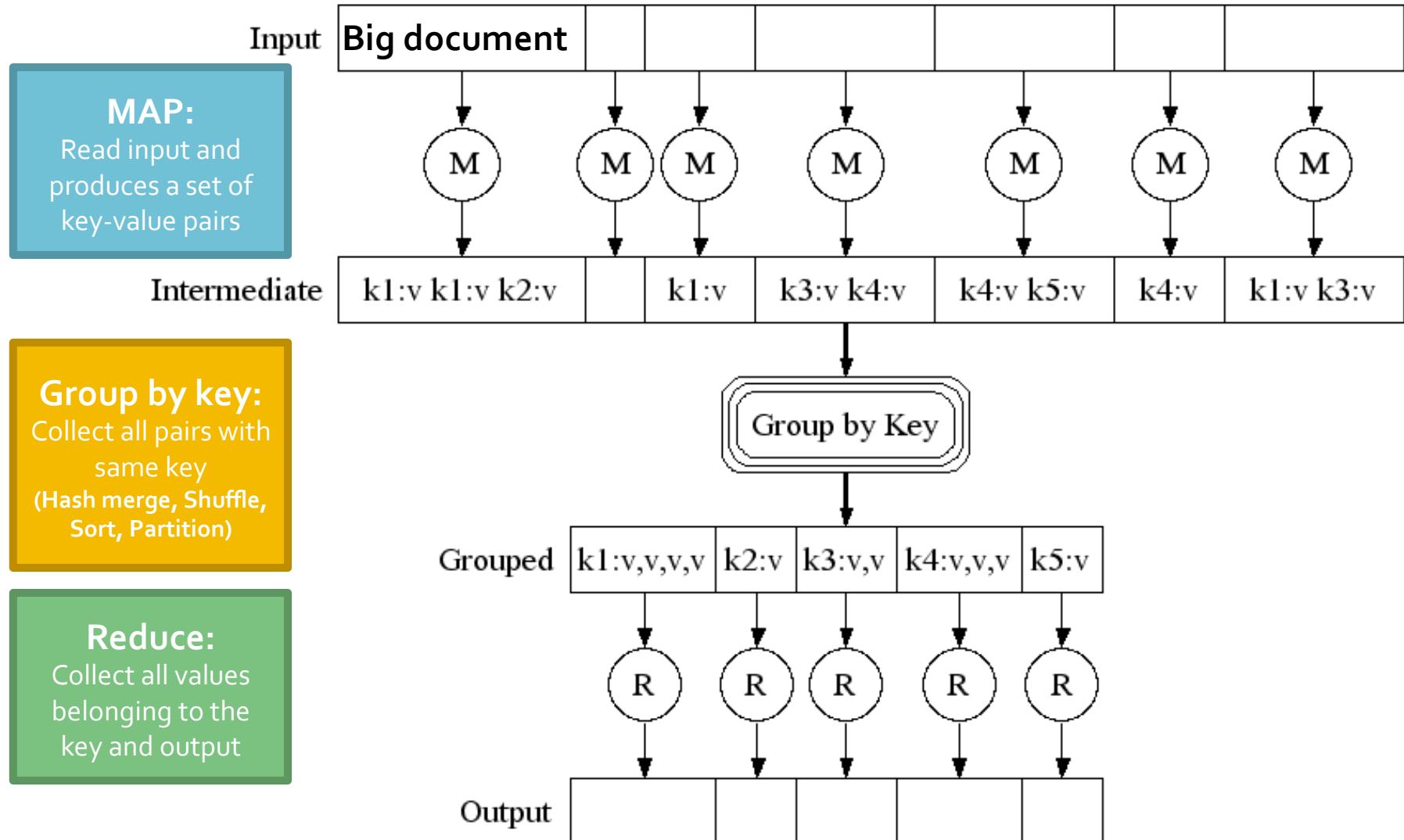
Map-Reduce

Scheduling and Data Flow

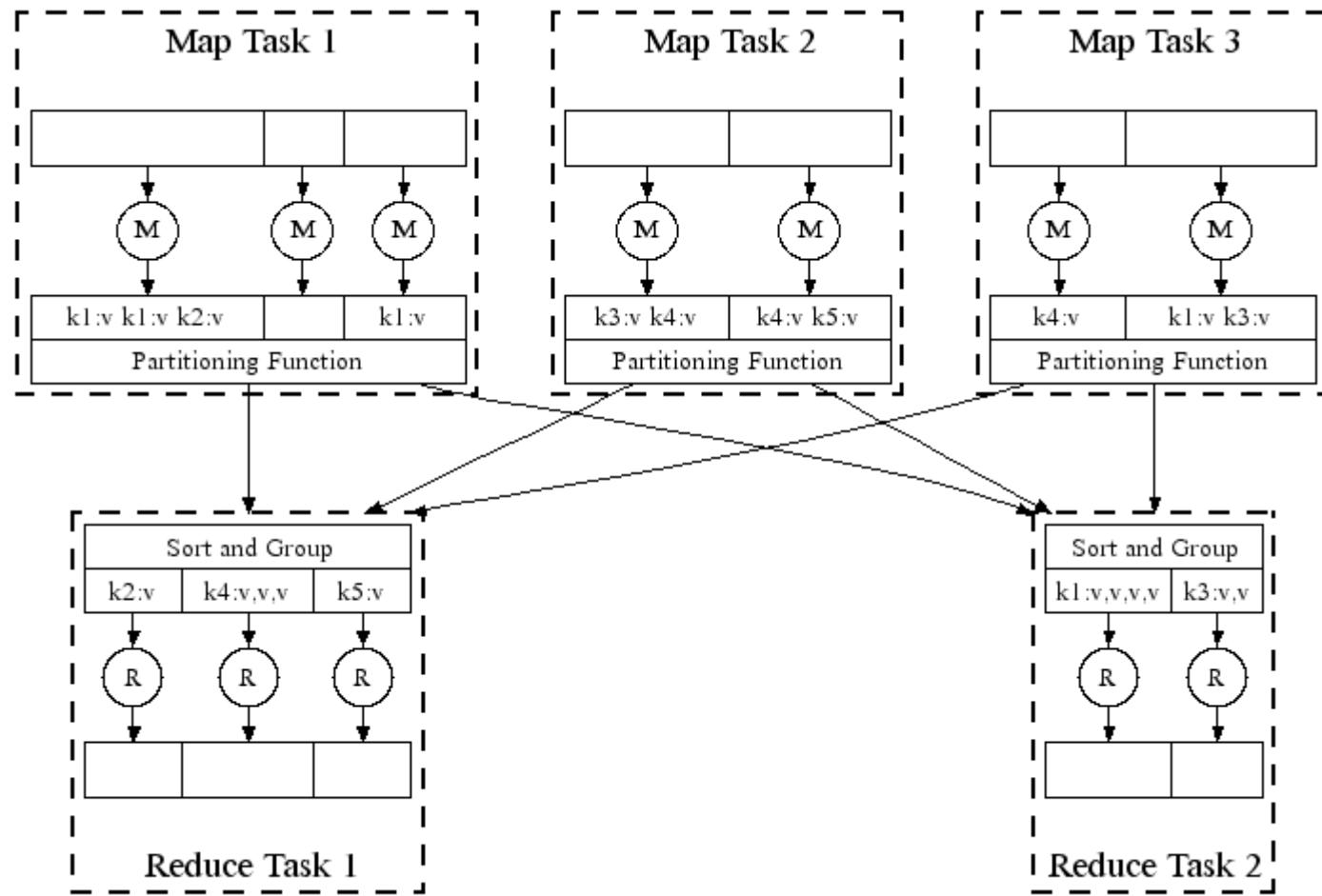
Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Map-Reduce: A diagram



Map-Reduce: In Parallel



All phases are distributed with many tasks doing the work

Map-Reduce: Environment

Map-Reduce environment takes care of:

- Partitioning the input data
- Scheduling the program's execution across a set of machines
- Performing the group by key step
- Handling node failures
- Managing required inter-machine communication

Data Flow

- Input and final output are stored on the distributed file system (DFS):
 - Scheduler tries to schedule map tasks “close” to physical storage location of input data
- Intermediate results are stored on local FS of Map and Reduce workers
- Output is often input to another MapReduce task

Coordination: Master

- **Master node takes care of coordination:**
 - **Task status:** (idle, in-progress, completed)
 - **Idle tasks** get scheduled as workers become available
 - When a map task completes, it sends the master the location and sizes of its R intermediate files, one for each reducer
 - Master pushes this info to reducers
- Master pings workers periodically to detect failures

Dealing with Failures

■ Map worker failure

- Map tasks completed or in-progress at worker are reset to idle
- Idle tasks eventually rescheduled on other worker(s)

■ Reduce worker failure

- Only in-progress tasks are reset to idle
- Idle Reduce tasks restarted on other worker(s)

■ Master failure

- MapReduce task is aborted and client is notified

How many Map and Reduce jobs?

- M map tasks, R reduce tasks
- **Rule of thumb:**
 - Make M much larger than the number of nodes in the cluster
 - One DFS chunk per map is common
 - Improves dynamic load balancing and speeds up recovery from worker failures
- **Usually R is smaller than M**
 - Because output is spread across R files

Map-Reduce

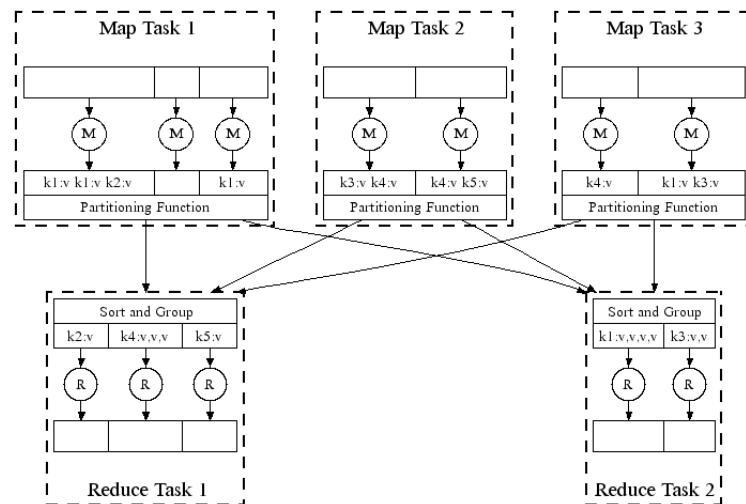
Refinements
Implementations

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



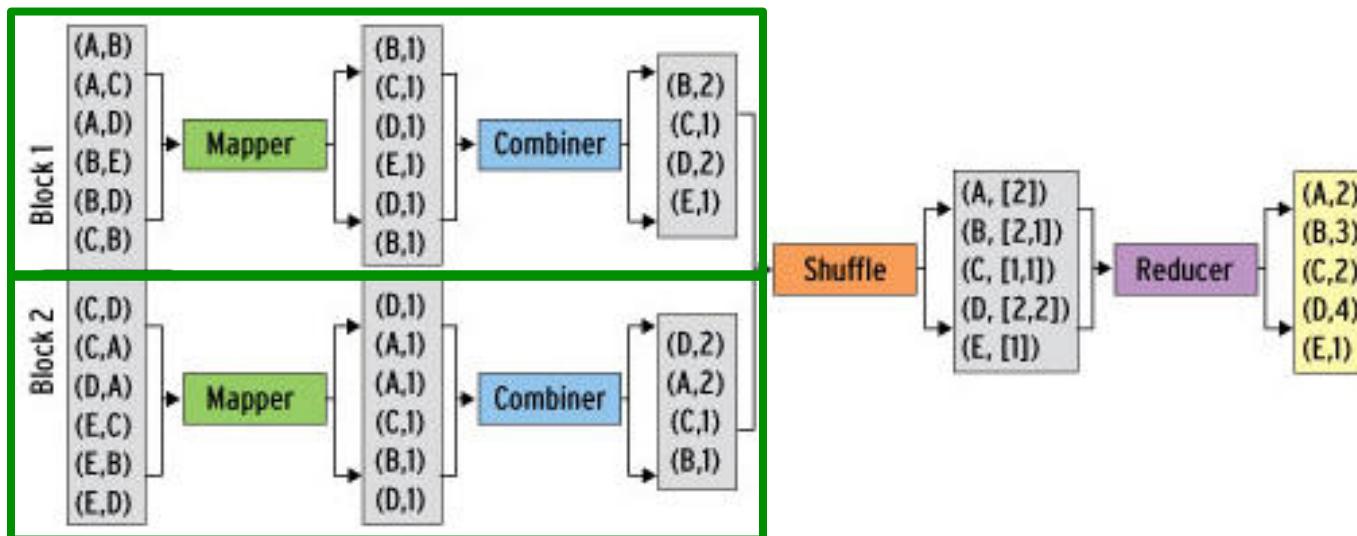
Refinement: Combiners (1)

- Often a Map task will produce many pairs of the form $(k, v_1), (k, v_2), \dots$ for the same key k
 - E.g., popular words in the word count example
- Can save network time by pre-aggregating values in the mapper:**
 - $\text{combine}(k, \text{list}(v_1)) \rightarrow v_2$
 - Combiner is **usually** same as the reduce function



Refinement: Combiners (2)

- Back to our word counting example:
 - Combiner combines the values of all keys of a single mapper (single node):



- Much less data needs to be copied and shuffled!

Refinement: Combiners (3)

- Combiner trick works only if reduce function is commutative and associative
- Sum
- Average
- Median

Refinement: Partition Function

- Want to control how keys get partitioned
 - The set of keys that go to a single reduce worker
- System uses a default partition function:
 - $\text{hash}(\text{key}) \bmod R$
- Sometimes useful to override the hash function:
 - E.g., $\text{hash}(\text{hostname(URL)}) \bmod R$ ensures URLs from a host end up in the same output file

Implementations

- **Google MapReduce**
 - Uses Google File System (GFS) for stable storage
 - Not available outside Google
- **Hadoop**
 - Open-source implementation in Java
 - Uses HDFS for stable storage
 - Download: <http://lucene.apache.org/hadoop/>
- **Hive, Pig**
 - Provide SQL-like abstractions on top of Hadoop Map-Reduce layer

Cloud Computing

- Ability to rent computing by the hour
 - Additional services e.g., persistent storage
- E.g., Amazon's “Elastic Compute Cloud” (**EC2**)
 - S3 (stable storage)
 - Elastic Map Reduce (**EMR**)

Pointers and Further Reading

Reading

- Jeffrey Dean and Sanjay Ghemawat:
MapReduce: Simplified Data Processing on
Large Clusters
 - <http://labs.google.com/papers/mapreduce.html>
- Sanjay Ghemawat, Howard Gobioff, and
Shun-Tak Leung: The Google File System
 - <http://labs.google.com/papers/gfs.html>

Resources

- Hadoop Wiki
 - Introduction
 - <http://wiki.apache.org/lucene-hadoop/>
 - Getting Started
 - <http://wiki.apache.org/lucene-hadoop/GettingStartedWithHadoop>
 - Map/Reduce Overview
 - <http://wiki.apache.org/lucene-hadoop/HadoopMapReduce>
 - <http://wiki.apache.org/lucene-hadoop/HadoopMapRedClasses>
 - Eclipse Environment
 - <http://wiki.apache.org/lucene-hadoop/EclipseEnvironment>
- Javadoc
 - <http://lucene.apache.org/hadoop/docs/api/>

Resources

- Releases from Apache download mirrors
 - <http://www.apache.org/dyn/closer.cgi/lucene/hadoop/>
- Nightly builds of source
 - <http://people.apache.org/dist/lucene/hadoop/nightly/>
- Source code from subversion
 - [http://lucene.apache.org/hadoop/version control.html](http://lucene.apache.org/hadoop/version_control.html)

Stanford

Mining Massive Datasets

This class teaches algorithms for extracting models and other information from very large amounts of data. The emphasis is on techniques that are efficient and that scale well.



About the Course

We introduce the student to modern distributed file systems and MapReduce, including what distinguishes good MapReduce algorithms from good algorithms in general. The rest of the course is devoted to algorithms for extracting models and information from large datasets. Students will learn how Google's PageRank algorithm models importance of Web pages and some of the many extensions that have been used for a variety of purposes. We'll cover locality-sensitive hashing, a bit of magic that allows you to find similar items in a set of items so large you cannot possibly compare each pair. When data is stored as a very large, sparse matrix, dimensionality reduction is often a good way to model the data, but standard approaches do not scale well; we'll talk about efficient approaches. Many other large-scale algorithms are covered as well, as outlined in the course syllabus.

Course Syllabus

Week 1:
MapReduce
Link Analysis -- PageRank

Week 2:
Locality-Sensitive Hashing -- Basics + Applications
Distance Measures
Nearest Neighbors
Frequent Itemsets

Week 3:
Data Stream Mining
Analysis of Large Graphs

Week 4:
Recommender Systems
Dimensionality Reduction

Week 5:
Clustering
Computational Advertising

Week 6:
Support-Vector Machines
Decision Trees
MapReduce Algorithms

Week 7:
More About Link Analysis -- Topic-specific PageRank, Link Spam.
More About Locality-Sensitive Hashing

Recommended Background

A course in database systems programming (e.g., SQL) is recommended, as is a basic course on algorithms and data structures.

Sessions

Sep 29th 2014 - Dec 1st 2014 ▾

[Go to class](#)

Eligible for

Statement of Accomplishment

Course at a Glance

- 📅 7 weeks of study
- ⌚ 8-10 hours of work / week
- 🌐 English
- 🎞 English subtitles

Instructors



Jure Leskovec
Stanford University



Anand Rajaraman
Stanford University



Jeff Ullman
Stanford University

Categories

Computer Science: Systems & Security
Computer Science: Artificial Intelligence

Share

1.3k Share 278 g+1 581 Tweet

Suggested Readings

There is a free book "Mining of Massive Datasets, by Leskovec, Rajaraman, and Ullman (who by coincidence are the instructors for this course :-). You can download it at <http://www.mmds.org/> Hardcopies can be purchased from Cambridge Univ. Press.

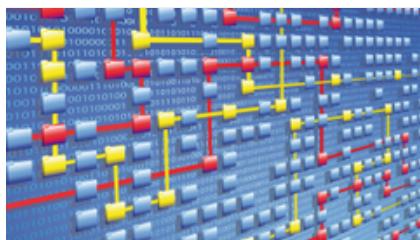
Course Format

There will be about 2 hours of video to watch each week, broken into small segments. There will be automated homeworks to do for each week, and a final exam.

FAQ

- Will I get a Statement of Accomplishment after completing this class?**
Yes. Students who successfully complete the class will receive a Statement of Accomplishment signed by the instructors. A level designated "distinction" will also be offered.

Related Courses



[Process Mining: Data science in Action](#)



[Cryptography](#)



[人工智慧\(Artificial Intelligence\)](#)

[Browse more courses](#)

Coursera empowers people to improve their lives, the lives of their families, and the communities they live in with education.

© 2014 Coursera Inc. All rights reserved.

COMPANY

About
People
Leadership
Careers

FRIENDS

Partners
Community
Programs
Developers
Translate

CONNECT

Google+
Twitter
Facebook
Blog
Tech Blog

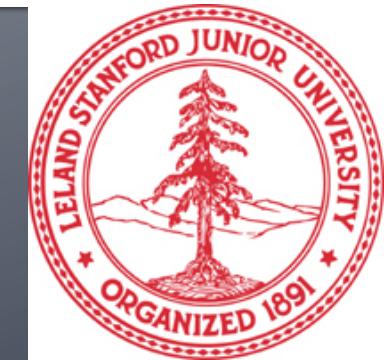
MORE

Terms
Privacy
Help
Press
Contact

Online Algorithms

The BALANCE Algorithm

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Adwords Problem

- **Given:**
 - A set of bids by advertisers for search queries
 - A click-through rate for each advertiser-query pair
 - A budget for each advertiser (say for 1 day, month...)
 - A limit on the number of ads to be displayed with each search query
- **Respond to each search query with a set of advertisers such that:**
 - The size of the set is no larger than the limit on the number of ads per query
 - Each advertiser has bid on the search query
 - Each advertiser has enough budget left to pay for the ad if it is clicked upon

Dealing with Limited Budgets

- Our setting: Simplified environment

- There is 1 ad shown for each query
- All advertisers have the same budget B
- All ads are equally likely to be clicked
- Value of each ad is the same (=1)

- Simplest algorithm is greedy:

- For a query pick any advertiser who has bid 1 for that query
- Competitive ratio of greedy is 1/2

Bad Scenario for Greedy

BALANCE Algorithm [MSVV]

- **BALANCE** Algorithm by Mehta, Saberi, Vazirani, and Vazirani
 - For each query, pick the advertiser with the largest unspent budget
 - Break ties arbitrarily (but in a deterministic way)

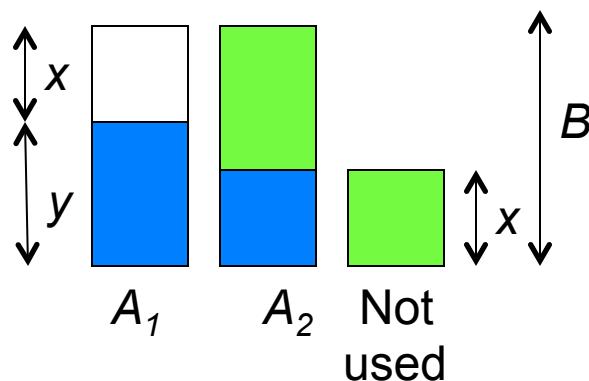
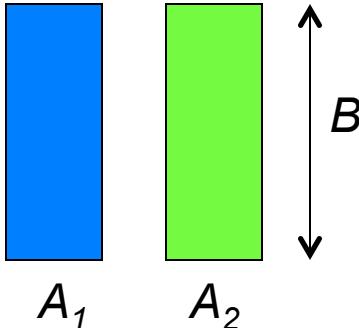
Example: BALANCE

- **Two advertisers A and B**
 - A bids on query x , B bids on x and y
 - Both have budgets of \$4
- **Query stream:** $x \ x \ x \ x \ y \ y \ y \ y$
- **BALANCE choice:** A B A B B B _ _
 - Optimal: A A A A B B B B
- **Competitive ratio = $\frac{3}{4}$**
 - For BALANCE with 2 advertisers

Analyzing 2-advertiser BALANCE

- Consider simple case
 - 2 advertisers, A_1 and A_2 , each with budget B (≥ 1)
 - Optimal solution exhausts both advertisers' budgets
- **BALANCE must exhaust at least one advertiser's budget:**
 - If not, we can allocate more queries
 - Assume BALANCE exhausts A_2 's budget

Analyzing Balance



- Queries allocated to A_1 in the optimal solution
- Queries allocated to A_2 in the optimal solution

Optimal revenue = $2B$

Balance revenue = $B+y$

Goal: Show that $y \geq B/2$

Case 1: BALANCE assigns at least $B/2$ blue queries to A_1 . So $y \geq B/2$.

Case 2: BALANCE assigns more than $B/2$ blue queries to A_2 .

Consider the last blue query assigned to A_2 .

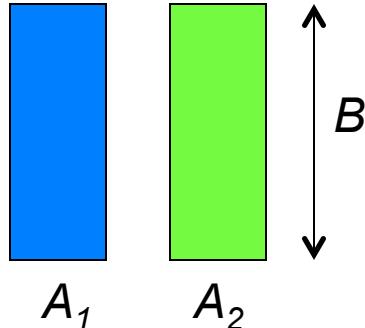
At that time, A_2 's unspent budget must have been at least as big as A_1 's.

That means at least as many queries have been assigned to A_1 as to A_2 .

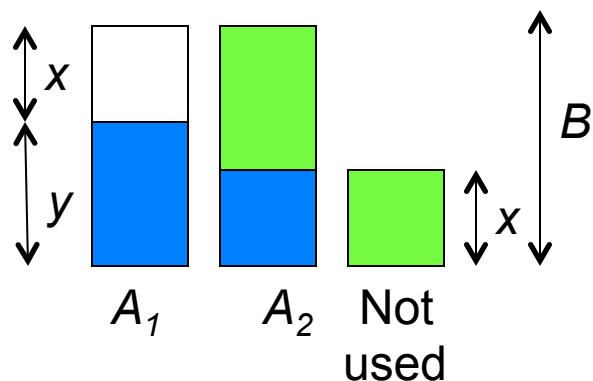
At this point, we have already assigned at least $B/2$ queries to A_2 .

So $y \geq B/2$.

Analyzing BALANCE



- Queries allocated to A_1 in the optimal solution
- Queries allocated to A_2 in the optimal solution



Optimal revenue $\text{OPT} = 2B$
Balance revenue $\text{BAL} = B+y$

We have shown that $y \geq B/2$
 $\text{BAL} \geq B+B/2 = 3B/2$
 $\text{BAL}/\text{OPT} \geq 3/4$

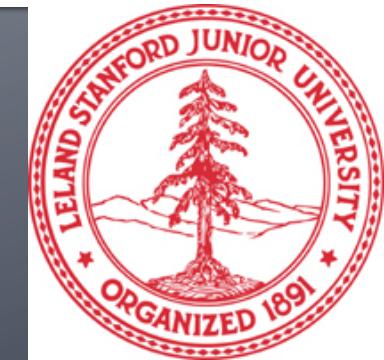
BALANCE: General Result

- In the general case, worst competitive ratio of BALANCE is $1 - 1/e = \text{approx. } 0.63$
 - Interestingly, no online algorithm has a better competitive ratio!
- Let's see the worst case example that gives this ratio

Online Algorithms

Generalized BALANCE

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



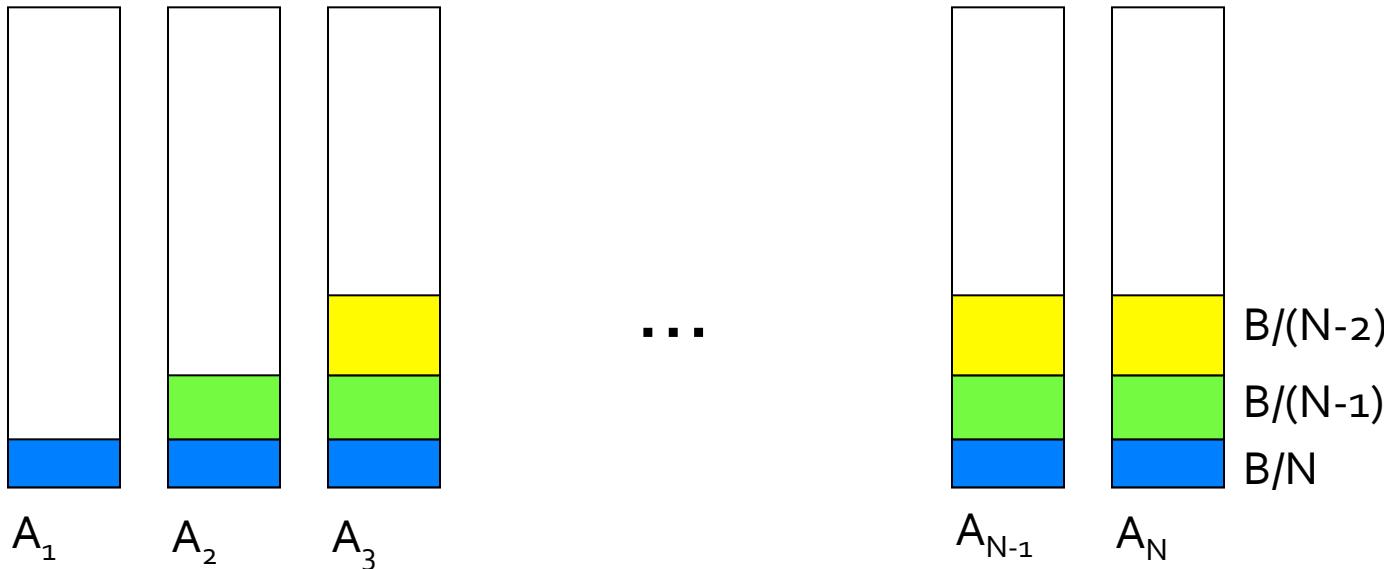
Worst case for BALANCE

- **N advertisers:** A_1, A_2, \dots, A_N
 - Each with budget $B > N$
- **Queries:**
 - $N \cdot B$ queries appear in N rounds of B queries each
- **Bidding:**
 - Round 1 queries: bidders A_1, A_2, \dots, A_N
 - Round 2 queries: bidders A_2, A_3, \dots, A_N
 - Round i queries: bidders A_i, \dots, A_N
- **Optimum allocation:**

Allocate round i queries to A_i

 - Optimum revenue $N \cdot B$

BALANCE Allocation



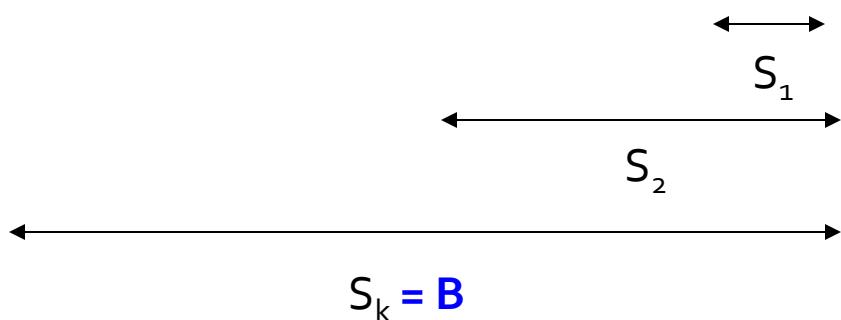
After k rounds, the allocation to advertiser k is:

$$S_k = \sum_{1 \leq i \leq k} B/(N-i+1)$$

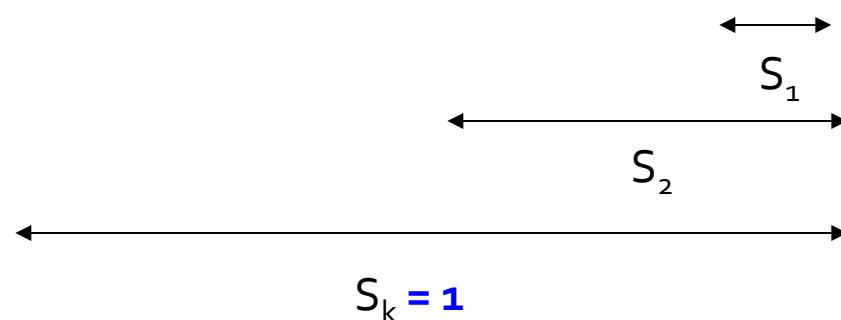
If we find the smallest k such that $S_k \geq B$, then after k rounds we cannot allocate any queries to any advertiser

BALANCE: Analysis

$B/1 \quad B/2 \quad B/3 \quad \dots \quad B/(N-(k-1)) \quad \dots \quad B/(N-1) \quad B/N$



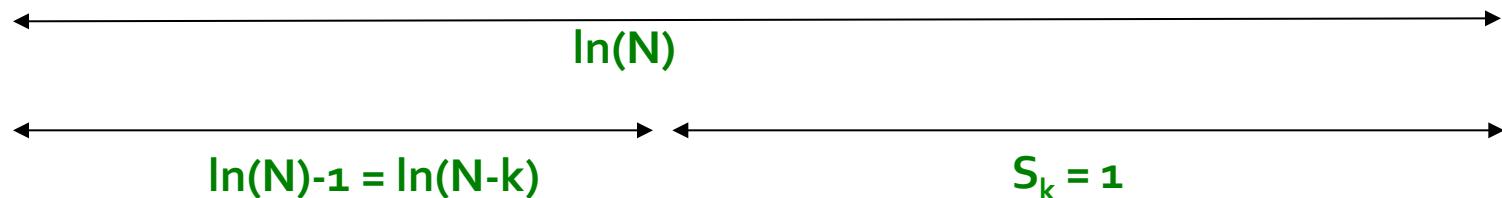
$1/1 \quad 1/2 \quad 1/3 \quad \dots \quad 1/(N-(k-1)) \quad \dots \quad 1/(N-1) \quad 1/N$



BALANCE: Analysis

- **Fact:** for large n
 - Result due to Euler

$$1/1 \quad 1/2 \quad 1/3 \quad \dots \quad 1/(N-(k-1)) \quad \dots \quad 1/(N-1) \quad 1/N$$



$$\ln(N-k) = \ln(N) - 1$$

$$\ln(N/(N-k)) = 1$$

$$N/(N-k) = e$$

$$k = N(1 - 1/e)$$

BALANCE: Analysis

- So after the first $k=N(1-1/e)$ rounds, we cannot allocate a query to any advertiser
- $\text{Revenue} = B \cdot N (1-1/e)$
- $\text{Competitive ratio} = 1-1/e$

General Version of the Problem

- So far: all bids = 1, all budgets equal (=B)
- **In a general setting BALANCE can be terrible**
 - Consider query \mathbf{q} , two advertisers A_1 and A_2
 - A_1 : *bid* = 1, *budget* = 110
 - A_2 : *bid* = 10, *budget* = 100
 - Suppose we see 10 instances of \mathbf{q}
 - BALANCE always selects A_1 and earns 10
 - Optimal earns 100

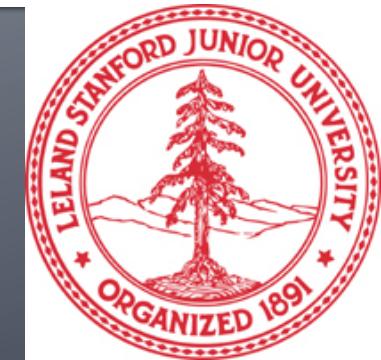
Generalized BALANCE

- Consider query q , bidder i
 - Bid = x_i ,
 - Budget = b_i ,
 - Amount spent so far = m_i ,
 - Fraction of budget left over $f_i = 1-m_i/b_i$,
 - Define $\psi_i(q) = x_i(1-e^{-f_i})$
- Allocate query q to bidder i with largest value of $\psi_i(q)$
- Same competitive ratio ($1-1/e$)

Online Algorithms

Performance-based Advertising

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Online Algorithms

- **Classic model of algorithms**
 - You get to see the entire input, then compute some function of it
 - In this context, “offline algorithm”
- **Online Algorithms**
 - You get to see the input one piece at a time, and need to make irrevocable decisions along the way
 - **Similar to the data stream model**

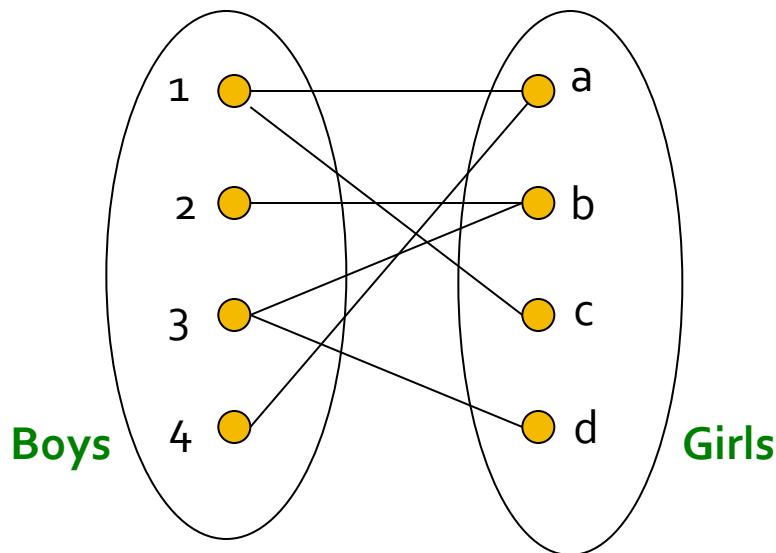
Online Algorithms

Bipartite Matching

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



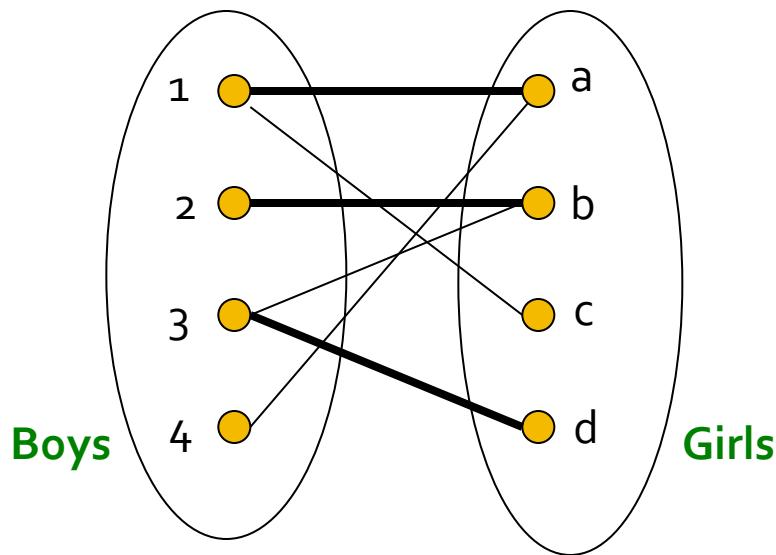
Example: Bipartite Matching



Nodes: Boys and Girls; Edges: Compatible Pairs

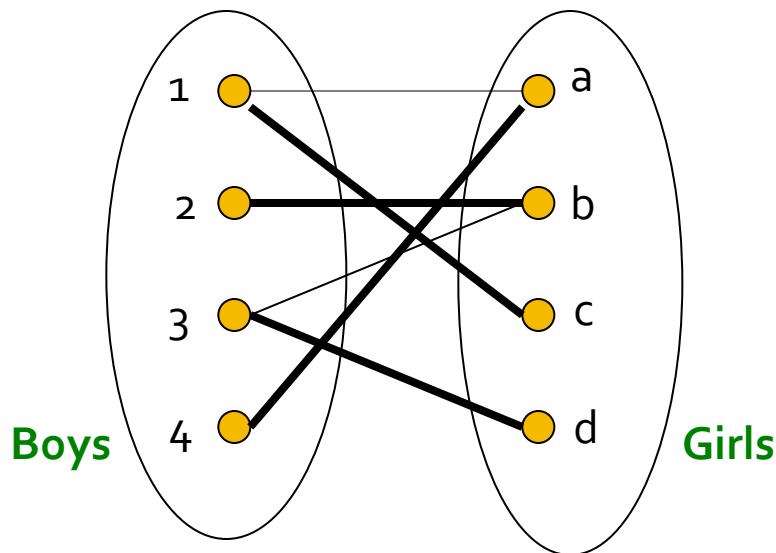
Goal: Match as many compatible pairs as possible

Example: Bipartite Matching



$M = \{(1,a), (2,b), (3,d)\}$ is a **matching**
Cardinality of matching = $|M| = 3$

Example: Bipartite Matching



$M = \{(1,c), (2,b), (3,d), (4,a)\}$ is a
perfect matching

Perfect matching ... all vertices of the graph are matched

Maximum matching ... a matching that contains the largest possible number of matches

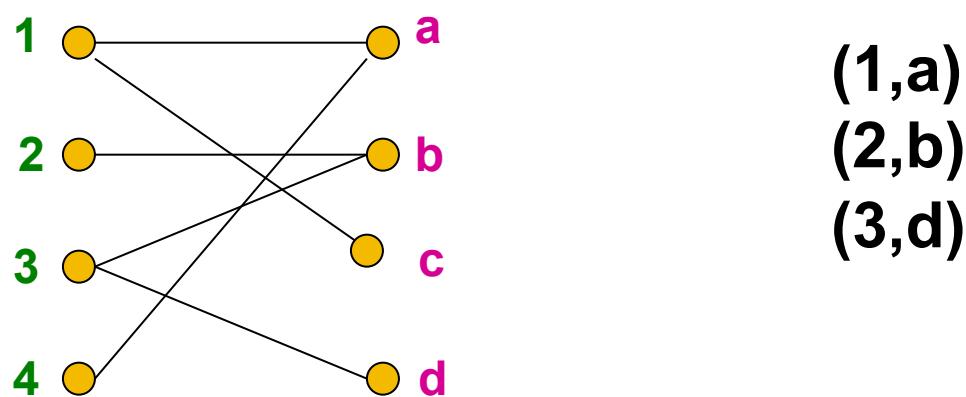
Matching Algorithm

- **Problem:** Find a maximum matching for a given bipartite graph
 - A perfect one if it exists
- There is a polynomial-time offline algorithm based on augmenting paths (Hopcroft & Karp 1973, see http://en.wikipedia.org/wiki/Hopcroft-Karp_algorithm)
- **But what if we do not know the entire graph upfront?**

Online Graph Matching Problem

- Initially, we are given the set **boys**
- In each **round**, **one girl's choices are revealed**
 - That is, girl's **edges** are revealed
- **At that time, we have to decide to either:**
 - Pair the **girl** with a **boy**
 - Do not pair the **girl** with any **boy**
- **Example of application:**
Assigning tasks to servers

Online Graph Matching: Example



Greedy Algorithm

- **Greedy algorithm for the online graph matching problem:**
 - Pair the new girl with **any** eligible boy
 - If there is none, do not pair girl
- **How good is the algorithm?**

Competitive Ratio

- For input I , suppose greedy produces matching M_{greedy} while an optimal matching is M_{opt}

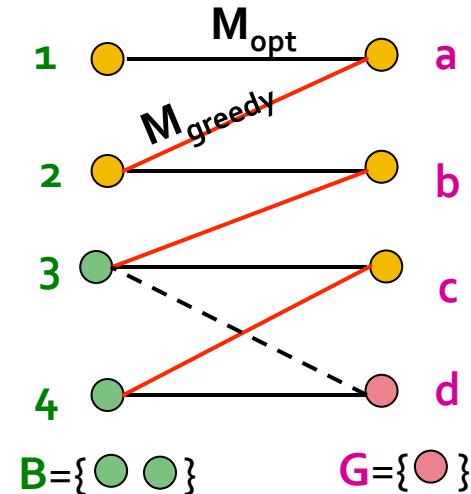
Competitive ratio =

$$\min_{\text{all possible inputs } I} (|M_{greedy}| / |M_{opt}|)$$

(what is greedy's worst performance over all possible inputs I)

Analyzing the Greedy Algorithm

- Suppose $M_{greedy} \neq M_{opt}$
- Consider the set G of girls matched in M_{opt} but not in M_{greedy}
- (1) $|M_{opt}| \leq |M_{greedy}| + |G|$

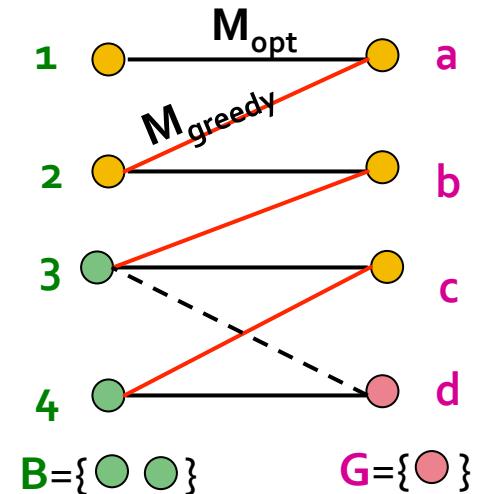


- Every boy B adjacent to girls in G is already matched in M_{greedy}
- (2) $|M_{greedy}| \geq |B|$

Analyzing the Greedy Algorithm

So far:

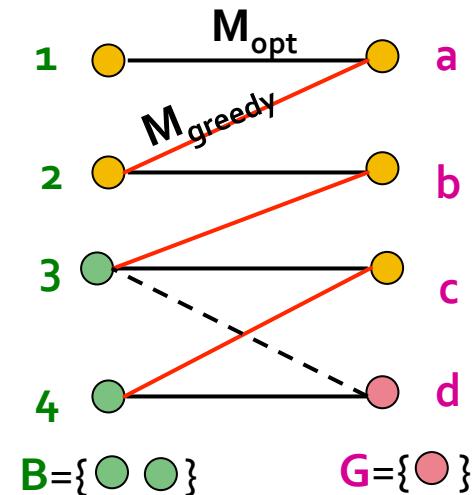
- G matched in M_{opt} but not in M_{greedy}
- Boys B adjacent to girls G
- (1) $|M_{opt}| \leq |M_{greedy}| + |G|$
- (2) $|M_{greedy}| \geq |B|$



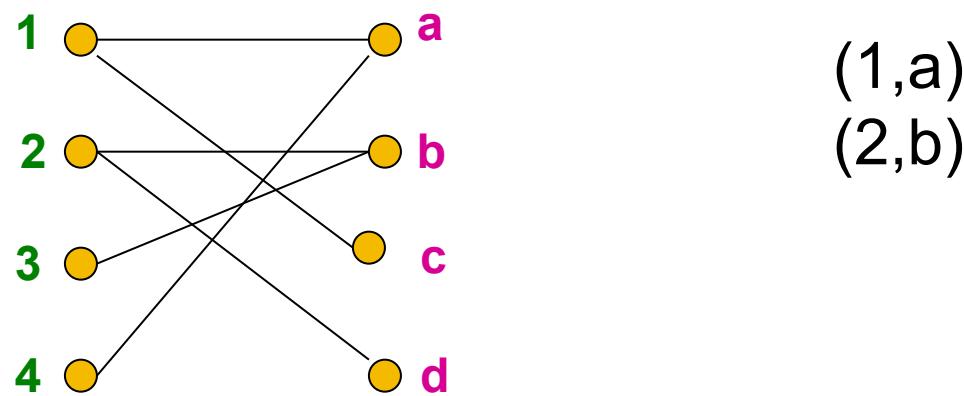
- Optimal matches all the girls in G to boys in B
 - (3) $|G| \leq |B|$
- Combining (2) and (3):
 - (4) $|G| \leq |B| \leq |M_{greedy}|$

Analyzing the Greedy Algorithm

- So we have:
 - (1) $|M_{opt}| \leq |M_{greedy}| + |G|$
 - (4) $|G| \leq |B| \leq |M_{greedy}|$
- Combining (1) and (4):
 - $|M_{opt}| \leq |M_{greedy}| + |M_{greedy}|$
 - $|M_{opt}| \leq 2|M_{greedy}|$
 - $|M_{greedy}| / |M_{opt}| \geq 1/2$



Worst-case Scenario



(1,a)
(2,b)

Online Algorithms

Performance-based Advertising The AdWords Problem

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



History of Web Advertising

■ Banner ads (1995-2001)

- Initial form of web advertising
- Popular websites charged X\$ for every 1,000 “impressions” of the ad
 - Called “**CPM**” rate
(Cost per thousand impressions)
 - Modeled similar to TV, magazine ads
- From **untargeted** to **demographically targeted**
- **Low click-through rates**
 - Low ROI for advertisers



Performance-based Advertising

- Introduced by Overture around 2000
 - Advertisers **bid on search keywords**
 - When someone searches for that keyword, the **highest bidder's ad is shown**
 - Advertiser is charged only if the ad is clicked on
- Similar model adopted by Google with some changes around 2002
 - Called **Adwords**

Algorithmic Challenges

- **Performance-based advertising works!**
 - Multi-billion-dollar industry
- **What ads to show for a given query?**
 - (Today's lecture)
- **If I am an advertiser, which search terms should I bid on and how much should I bid?**
 - (Not focus of today's lecture)

AdWords Problem

- A stream of queries arrives at the search engine: q_1, q_2, \dots
- Several advertisers bid on each query
- When query q_i arrives, search engine must pick a subset of advertisers whose ads are shown
- **Goal: Maximize search engine's revenues**
- **Clearly we need an online algorithm!**

Expected Revenue

Advertiser	Bid	CTR	Bid * CTR
A	\$1.00	1%	1 cent
B	\$0.75	2%	1.5 cents
C	\$0.50	2.5%	1.125 cents

Click through
rate Expected
revenue

The Adwords Innovation

Instead of sorting advertisers by bid, sort by expected revenue!

Advertiser	Bid	CTR	Bid * CTR
B	\$0.75	2%	1.5 cents
C	\$0.50	2.5%	1.125 cents
A	\$1.00	1%	1 cent

Adwords Problem

- **Given:**
 - A set of bids by advertisers for search queries
 - A click-through rate for each advertiser-query pair
 - A budget for each advertiser (say for 1 day, month...)
 - A limit on the number of ads to be displayed with each search query
- **Respond to each search query with a set of advertisers such that:**
 - The size of the set is no larger than the limit on the number of ads per query
 - Each advertiser has bid on the search query
 - Each advertiser has enough budget left to pay for the ad if it is clicked upon

Limitations of Simple Algorithm

Instead of sorting advertisers by bid, sort by expected revenue!

Advertiser	Bid	CTR	Bid * CTR
B	\$0.75	2%	1.5 cents
C	\$0.50	2.5%	1.125 cents
A	\$1.00	1%	1 cent

- CTR of an ad is unknown
- Advertisers have limited budgets and bid on multiple ads (BALANCE algorithm)

Estimating CTR

- Clickthrough rate (CTR) for a query-ad pair is measured historically
 - Averaged over a time period
- Some complications we won't cover in this lecture
 - CTR is position dependent
 - Ad #1 is clicked more than Ad #2
 - Explore v Exploit: Keep showing ads we already know the CTR of, or show new ads to estimate their CTR?

The A-Priori Algorithm

Monotonicity of “Frequent”
Candidate Pairs
Extension to Larger Itemsets

A-Priori Algorithm

- A two-pass approach called *a-priori* limits the need for main memory.
- Key idea: *monotonicity*: if a set of items appears at least s times, so does every subset of s .
- *Contrapositive for pairs*: if item i does not appear in s baskets, then no pair including i can appear in s baskets.

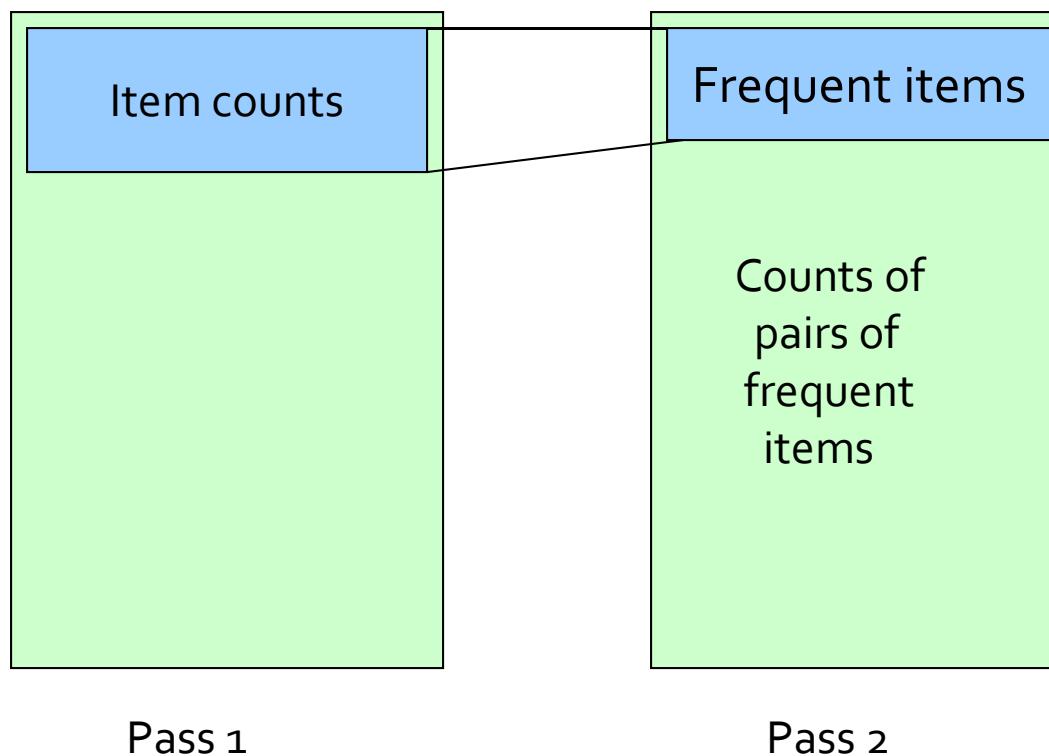
A-Priori Algorithm – (2)

- Pass 1: Read baskets and count in main memory the occurrences of each item.
 - Requires only memory proportional to #items.
- Items that appear at least s times are the *frequent items*.

A-Priori Algorithm – (3)

- Pass 2: Read baskets again and count in main memory only those pairs both of which were found in Pass 1 to be frequent.
- Requires memory proportional to square of *frequent* items only (for counts), plus a list of the frequent items (so you know what must be counted).

Picture of A-Priori



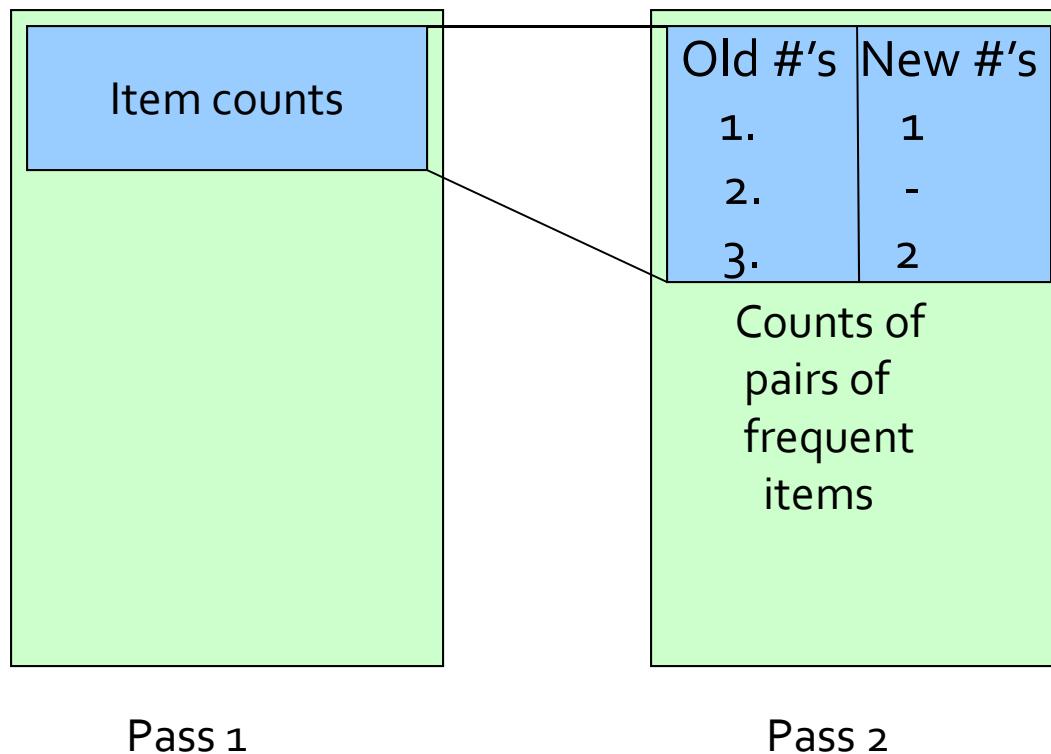
Pass 1

Pass 2

Detail for A-Priori

- You can use the triangular matrix method with $n = \text{number of frequent items}$.
 - May save space compared with storing triples.
- Trick: number frequent items 1,2,... and keep a table relating new numbers to original item numbers.

A-Priori Using Triangular Matrix

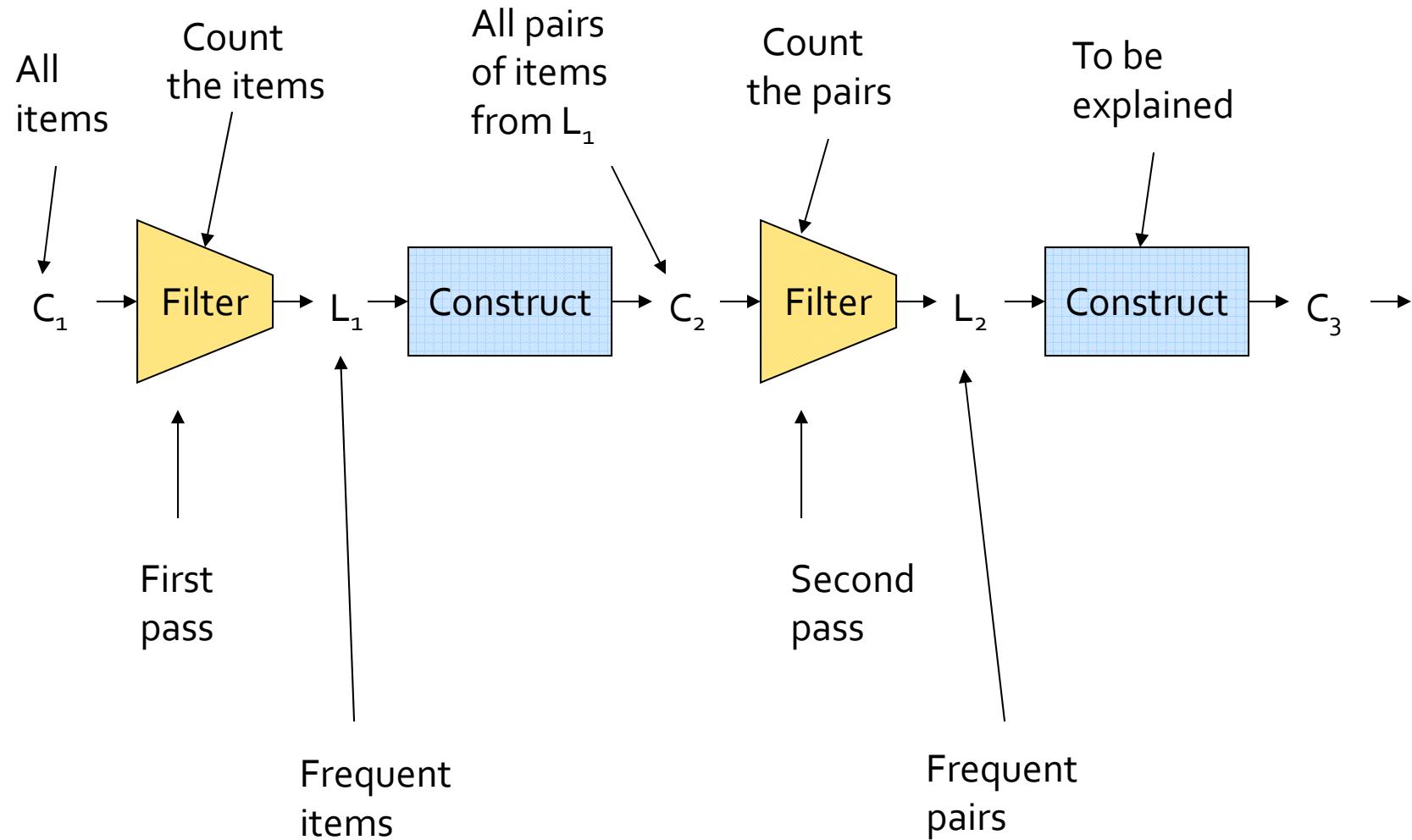


Pass 1

Pass 2

Frequent Triples, Etc.

- For each k , we construct two sets of *k-sets* (sets of size k):
 - C_k = *candidate k-sets* = those that might be frequent sets ($\text{support} \geq s$) based on information from the pass for $k - 1$.
 - L_k = the set of truly frequent k -sets.



Passes Beyond Two

- C_1 = all items
- In general, L_k = members of C_k with support $\geq s$.
 - Requires one pass.
- C_{k+1} = $(k+1)$ -sets, each k of which is in L_k .

Memory Requirements

- At the k^{th} pass, you need space to count each member of C_k .
- In realistic cases, because you need fairly high support, the number of candidates of each size drops, once you get beyond pairs.

All (Or Most) Frequent Itemsets In ≤ 2 Passes

Simple Algorithm

Savasere-Omiecinski- Navathe (SON)

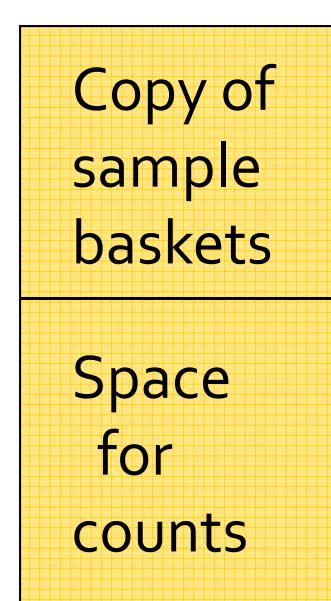
Algorithm

Toivonen's Algorithm

Simple Algorithm

- Take a random sample of the market baskets.
- Run a-priori or one of its improvements (for sets of all sizes, not just pairs) in main memory, so you don't pay for disk I/O each time you increase the size of itemsets.
- Use as your support threshold a suitable, scaled-back number.
 - **Example:** if your sample is $1/100$ of the baskets, use $s/100$ as your support threshold instead of s .

Main-Memory Picture



Simple Algorithm – Option

- Optionally, verify that your guesses are truly frequent in the entire data set by a second pass.
- But you don't catch sets frequent in the whole but not in the sample.
 - Smaller threshold, e.g., $s/125$ instead of $s/100$, helps catch more truly frequent itemsets.
 - But requires more space.

SON Algorithm

- Repeatedly read small subsets of the baskets into main memory and perform the first pass of the simple algorithm on each subset.
- An itemset becomes a candidate if it is found to be frequent in *any* one or more subsets of the baskets.

SON Algorithm – Pass 2

- On a second pass, count all the candidate itemsets and determine which are frequent in the entire set.
- Key “monotonicity” idea: an itemset cannot be frequent in the entire set of baskets unless it is frequent in at least one subset.

SON Algorithm – Distributed Version

- This idea lends itself to distributed data mining.
- If baskets are distributed among many nodes, compute *local* frequent itemsets at each node, then distribute the candidates from each node.
- Each node counts all the candidate itemsets.
- Finally, accumulate the counts of all candidates.

Toivonen's Algorithm

- Start as in the simple algorithm, but lower the threshold slightly for the sample.
 - **Example:** if the sample is 1% of the baskets, use $s/125$ as the support threshold rather than $s/100$.
 - Goal is to avoid missing any itemset that is frequent in the full set of baskets.

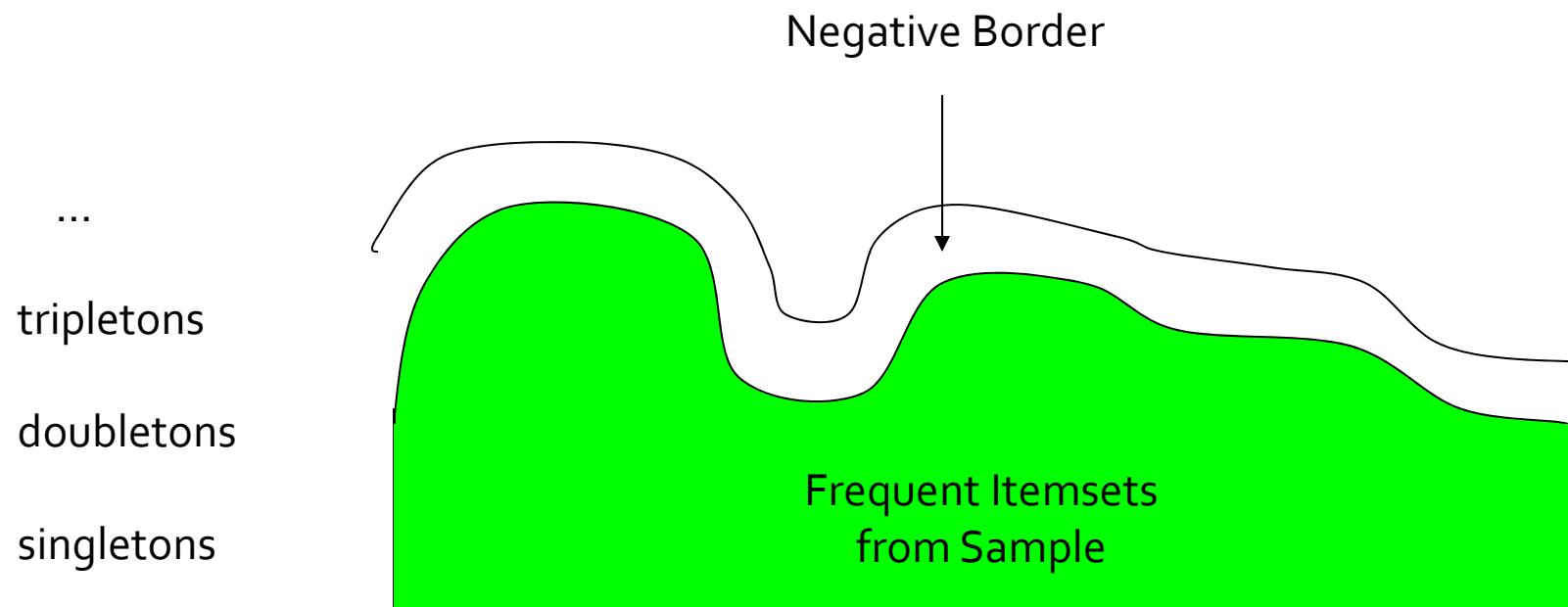
Toivonen's Algorithm – (2)

- Add to the itemsets that are frequent in the sample the *negative border* of these itemsets.
- An itemset is in the negative border if it is not deemed frequent in the sample, but *all* its immediate subsets are.

Example: Negative Border

- $\{A,B,C,D\}$ is in the negative border if and only if:
 1. It is not frequent in the sample, but
 2. All of $\{A,B,C\}$, $\{B,C,D\}$, $\{A,C,D\}$, and $\{A,B,D\}$ are.
- $\{A\}$ is in the negative border if and only if it is not frequent in the sample.
 - Because the empty set is always frequent.
 - Unless there are fewer baskets than the support threshold (silly case).

Picture of Negative Border



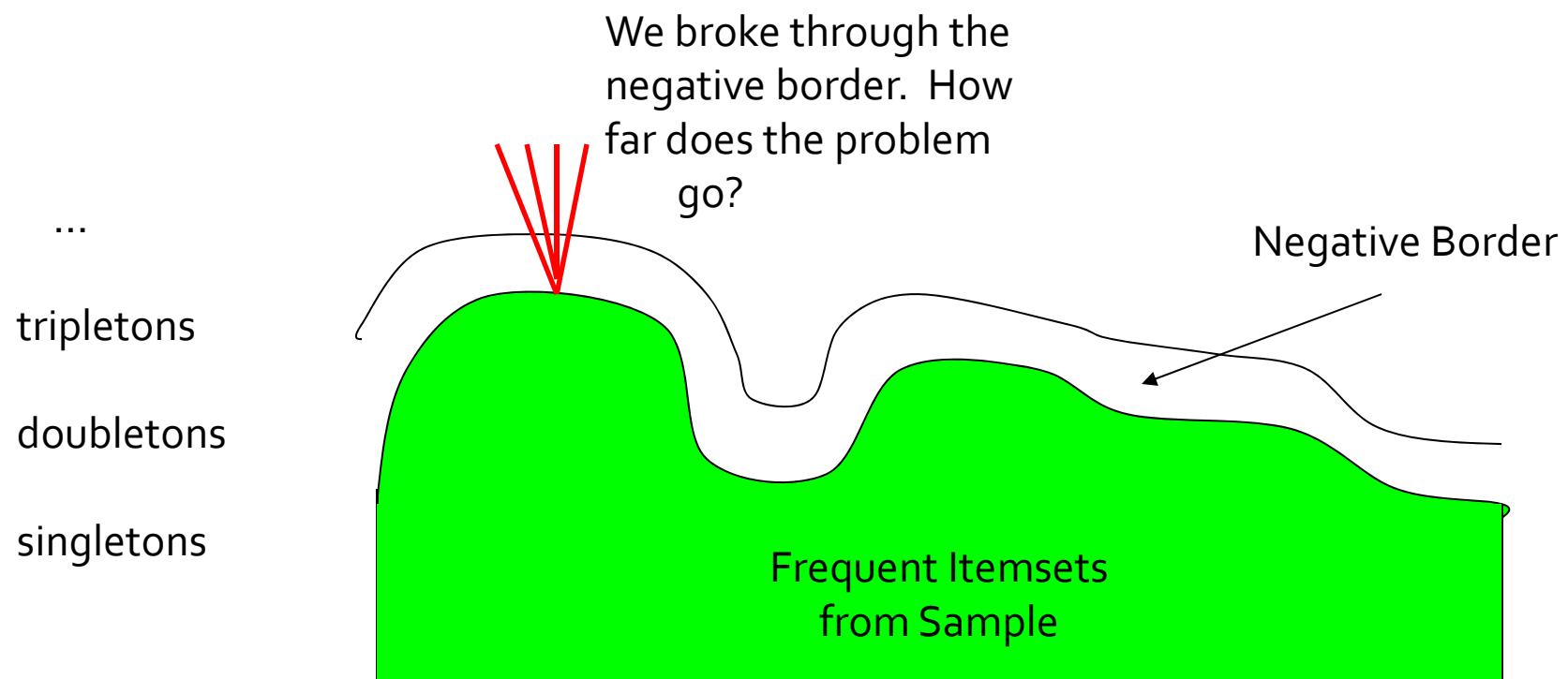
Toivonen's Algorithm – (3)

- In a second pass, count all candidate frequent itemsets from the first pass, and also count their negative border.
- If no itemset from the negative border turns out to be frequent, then the candidates found to be frequent in the whole data are *exactly* the frequent itemsets.

Toivonen's Algorithm – (4)

- What if we find that something in the negative border is actually frequent?
- We must start over again with another sample!
- Try to choose the support threshold so the probability of failure is low, while the number of itemsets checked on the second pass fits in main-memory.

If Something in the Negative Border Is Frequent . . .



Theorem:

- If there is an itemset that is frequent in the whole, but not frequent in the sample, then there is a member of the negative border for the sample that is frequent in the whole.

Proof:

- Suppose not; i.e.;
 1. There is an itemset S frequent in the whole but not frequent in the sample, and
 2. Nothing in the negative border is frequent in the whole.
- Let T be a **smallest** subset of S that is not frequent in the sample.
- T is frequent in the whole (S is frequent + monotonicity).
- T is in the negative border (else not “smallest”).

Frequent Itemsets

The Market-Basket Model
Association Rules
A-Priori Algorithm

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



The Market-Basket Model

- A large set of *items*, e.g., things sold in a supermarket.
- A large set of *baskets*, each of which is a small set of the items, e.g., the things one customer buys on one day.

Support

- Simplest question: find sets of items that appear “frequently” in the baskets.
- *Support* for itemset I = the number of baskets containing all items in I .
 - Sometimes given as a percentage.
- Given a *support threshold* s , sets of items that appear in at least s baskets are called *frequent itemsets*.

Example: Frequent Itemsets

- Items={milk, coke, pepsi, beer, juice}.
- Support = 3 baskets.

$$B_1 = \{m, c, b\}$$

$$B_3 = \{m, b\}$$

$$B_5 = \{m, p, b\}$$

$$B_7 = \{c, b, j\}$$

$$B_2 = \{m, p, j\}$$

$$B_4 = \{c, j\}$$

$$B_6 = \{m, c, b, j\}$$

$$B_8 = \{b, c\}$$

- Frequent itemsets: $\{m\}, \{c\}, \{b\}, \{j\}, \{m, b\}, \{b, c\}, \{c, j\}$.

Applications

- Items = products; baskets = sets of products someone bought in one trip to the store.
- Example application: given that many people buy beer and diapers together:
 - Run a sale on diapers; raise price of beer.
 - Only useful if many buy diapers & beer.
 - Essential for brick-and-mortar stores, not on-line stores.

Applications – (2)

- Baskets = sentences; items = documents containing those sentences.
- Items that appear together too often could represent plagiarism.
- Notice items do not have to be “in” baskets.
 - But it is better if baskets have small numbers of items, while items can be in large numbers of baskets.

Applications – (3)

- Baskets = documents; items = words.
- Unusual words appearing together in a large number of documents, e.g., “Brad” and “Angelina,” may indicate an interesting relationship.

Scale of the Problem

- WalMart sells 100,000 items and can store billions of baskets.
- The Web has billions of words and many billions of pages.

Association Rules

- If-then rules about the contents of baskets.
- $\{i_1, i_2, \dots, i_k\} \rightarrow j$ means: “if a basket contains all of i_1, \dots, i_k then it is *likely* to contain j .”
- *Confidence* of this association rule is the probability of j given i_1, \dots, i_k .
 - That is, the fraction of the baskets with i_1, \dots, i_k that also contain j .

Example: Confidence

$$\begin{array}{ll} + & B_1 = \{m, c, b\} \qquad B_2 = \{m, p, j\} \\ - & B_3 = \{m, b\} \qquad B_4 = \{c, j\} \\ - & B_5 = \{m, p, b\} \qquad + B_6 = \{m, c, b, j\} \\ & B_7 = \{c, b, j\} \qquad B_8 = \{b, c\} \end{array}$$

- An association rule: $\{m, b\} \rightarrow c$.
 - Confidence = $2/4 = 50\%$.

Finding Association Rules

- Question: “find all association rules with support $\geq s$ and confidence $\geq c$.”
 - Note: “support” of an association rule is the support of the set of items on the left.
- Hard part: finding the frequent itemsets.
 - Note: if $\{i_1, i_2, \dots, i_k\} \rightarrow j$ has high support and confidence, then both $\{i_1, i_2, \dots, i_k\}$ and $\{i_1, i_2, \dots, i_k, j\}$ will be “frequent.”

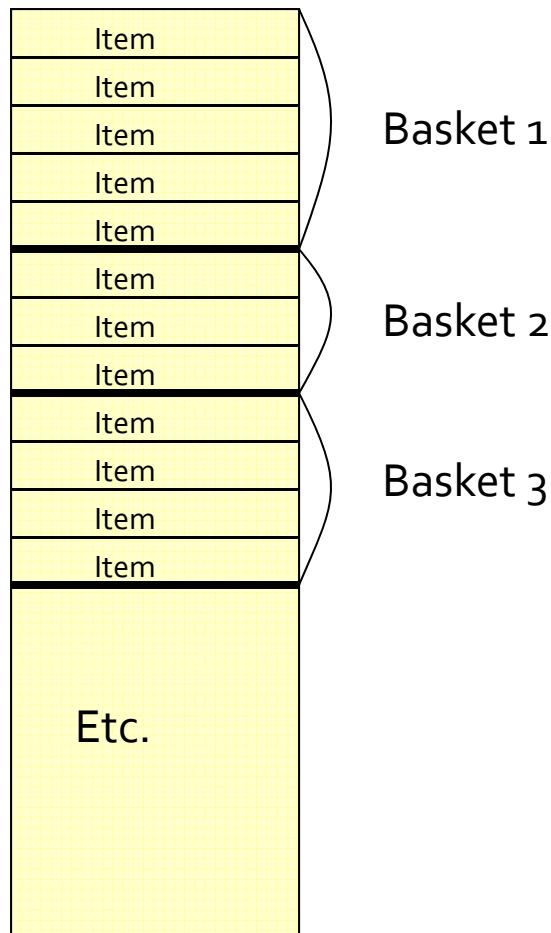
Finding Association Rules – (2)

1. Find all sets with support at least cs .
2. Find all sets with support at least s .
3. If $\{i_1, i_2, \dots, i_k, j\}$ has support at least cs , see which subsets missing one element have support at least s .
 - Take j to be the missing element.
4. $\{i_1, i_2, \dots, i_k\} \rightarrow j$ is an acceptable association rule if $\{i_1, i_2, \dots, i_k\}$ has support $s_1 \geq s$, $\{i_1, i_2, \dots, i_k, j\}$ has support $s_2 \geq cs$, and s_2/s_1 , the confidence of the rule, is at least c .

Computation Model

- Typically, data is kept in flat files.
- Stored on disk.
- Stored basket-by-basket.
- Expand baskets into pairs, triples, etc. as you read baskets.
 - Use k nested loops to generate all sets of size k .

File Organization



Example: items are positive integers, and boundaries between baskets are -1 .

Computation Model – (2)

- The true cost of mining disk-resident data is usually the **number of disk I/O's**.
- In practice, algorithms for finding frequent itemsets read the data in *passes* – all baskets read in turn.
- Thus, we measure the cost by the **number of passes** an algorithm takes.

Main-Memory Bottleneck

- For many frequent-itemset algorithms, main memory is the critical resource.
- As we read baskets, we need to count something, e.g., occurrences of pairs.
- The number of different things we can count is limited by main memory.
- Swapping counts in/out is a disaster.

Finding Frequent Pairs

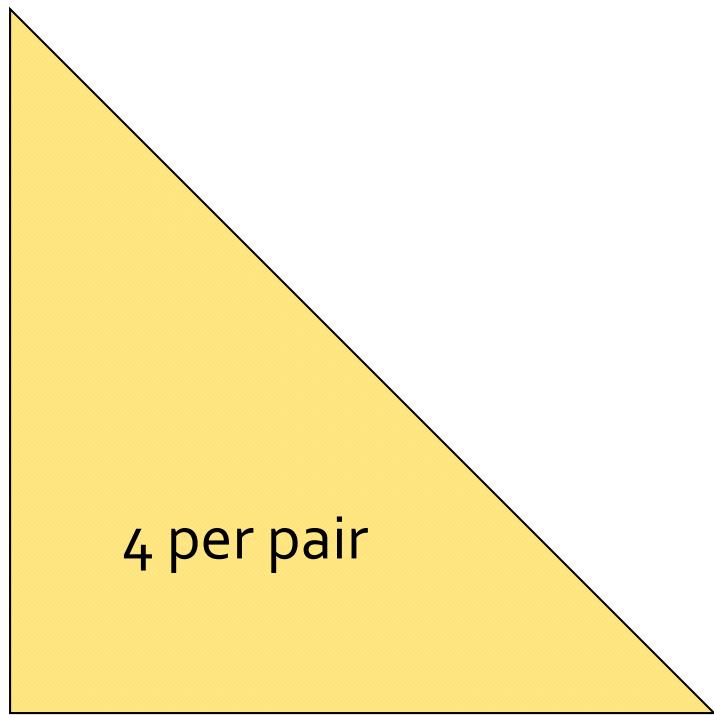
- The hardest problem often turns out to be finding the **frequent pairs**.
 - **Why?** Often frequent pairs are common, frequent triples are rare.
 - **Why?** Support threshold is usually set high enough that you don't get too many frequent itemsets.
- We'll concentrate on pairs, then extend to larger sets.

Naïve Algorithm

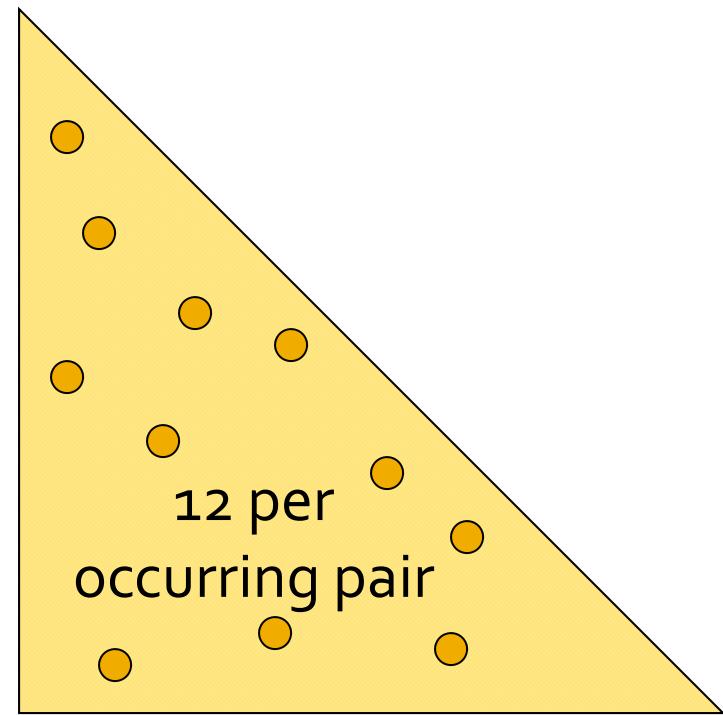
- Read file once, counting in main memory the occurrences of each pair.
 - From each basket of n items, generate its $n(n-1)/2$ pairs by two nested loops.
- Fails if $(\#items)^2$ exceeds main memory.
 - Remember: #items can be 100K (Wal-Mart) or 100B (Web pages).

Details of Main-Memory Counting

- Two approaches:
 1. Count all pairs, using a triangular matrix.
 2. Keep a table of triples $[i, j, c] = \text{"the count of the pair of items } \{i, j\} \text{ is } c\text{"}$
- (1) requires only 4 bytes/pair.
 - Note: always assume integers are 4 bytes.
 - (2) requires 12 bytes, but only for those pairs with count > 0.



Triangular matrix



Tabular method

Triangular-Matrix Approach

- Number items 1, 2, ...
 - Requires table of size $O(n)$ to convert item names to consecutive integers.
- Count $\{i, j\}$ only if $i < j$.
- Keep pairs in the order $\{1,2\}, \{1,3\}, \dots, \{1,n\}$, $\{2,3\}, \{2,4\}, \dots, \{2,n\}$, $\{3,4\}, \dots, \{3,n\}, \dots, \{n-1,n\}$.

Triangular-Matrix Approach – (2)

- Find pair $\{i, j\}$, where $i < j$, at the position:
$$(i - 1)(n - i/2) + j - i$$
- Total number of pairs $n(n - 1)/2$; total bytes about $2n^2$.

Details of Tabular Approach

- Total bytes used is about $12p$, where p is the number of pairs that actually occur.
 - Beats triangular matrix if at most 1/3 of possible pairs actually occur.
- May require extra space for retrieval structure, e.g., a hash table.

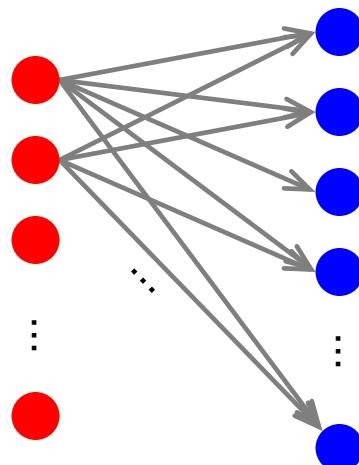
Analysis of Large Graphs: Trawling

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Trawling

- Searching for small communities in the Web graph
- What is the signature of a community / discussion in a Web graph?



Dense 2-layer graph

Use this to define “topics”:
What the same people on the left talk about on the right
Remember HITS!

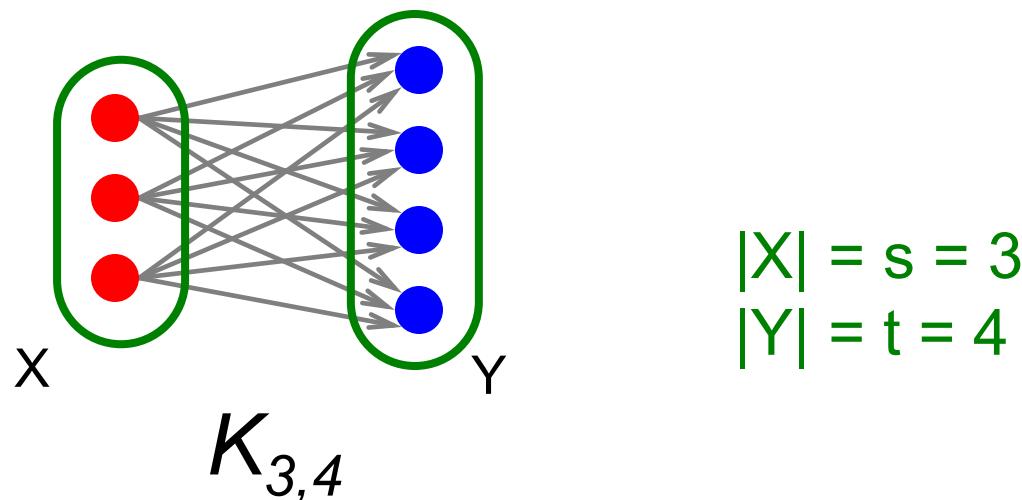
Intuition: Many people all talking about the same things

Searching for Small Communities

- **A more well-defined problem:**

Enumerate complete bipartite subgraphs $K_{s,t}$

- Where $K_{s,t} : s$ nodes on the “left” where each links to the same t other nodes on the “right”



Fully connected

Frequent Itemset Enumeration

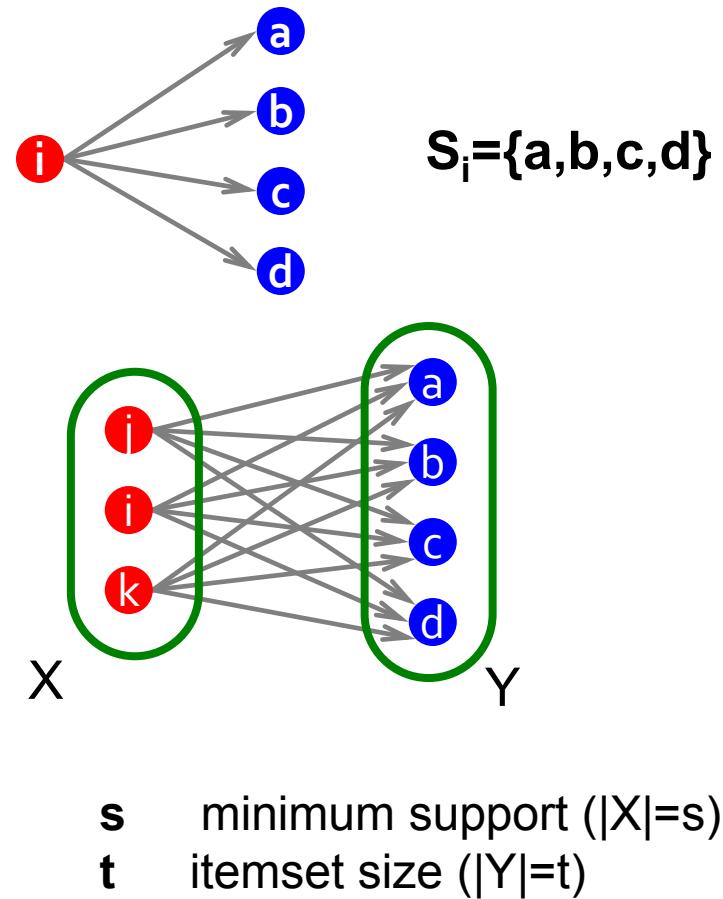
- **Market basket analysis.** Setting:
 - **Market:** Universe U of n items
 - **Baskets:** m subsets of U : $S_1, S_2, \dots, S_m \subseteq U$
(S_i is a set of items one person bought)
 - **Support:** Frequency threshold f
- **Goal:**
 - Find all subsets T s.t. $T \subseteq S_i$ of at least f sets S_i
(items in T were bought together at least f times)
- **What's the connection between the itemsets and complete bipartite graphs?**

From Itemsets to Bipartite $K_{s,t}$

Frequent itemsets = complete bipartite graphs!

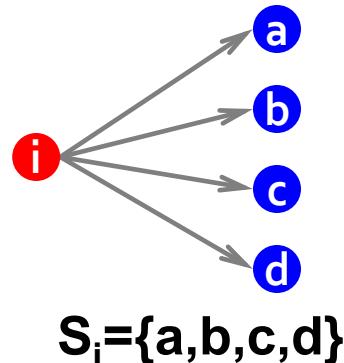
■ How?

- View each node i as a set S_i of nodes i points to
- $K_{s,t} =$ a set Y of size t that occurs in s sets S_i
- Looking for $K_{s,t} \rightarrow$ set of frequency threshold to s and look at layer t – all frequent sets of size t



From Itemsets to Bipartite $K_{s,t}$

View each node i as a set S_i of nodes i points to



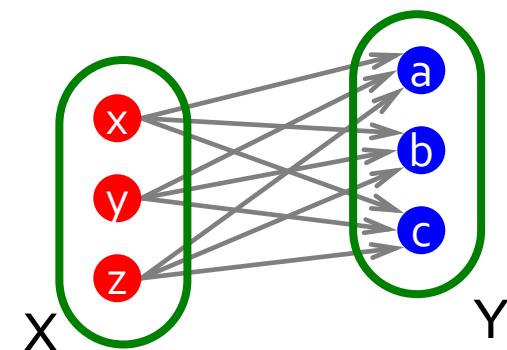
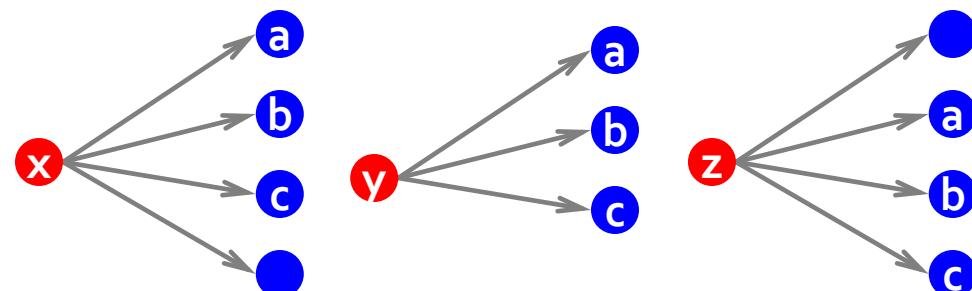
Find frequent itemsets:

s minimum support
t itemset size

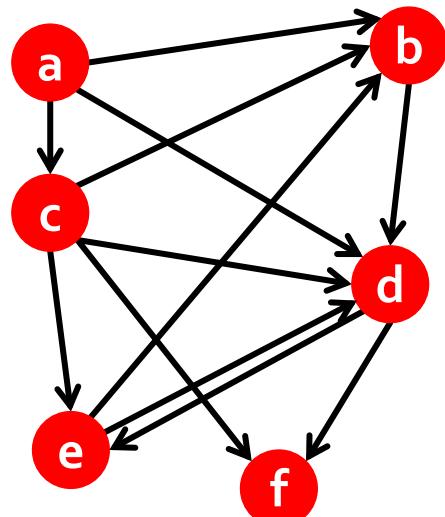
We found $K_{s,t}$!

$K_{s,t}$ = a set Y of size t that occurs in s sets S_i

Say we find a **frequent itemset** $Y=\{a,b,c\}$ of supp s
So, there are s nodes that link to all of $\{a,b,c\}$:



Example (1)



Itemsets:

$$a = \{b, c, d\}$$

$$b = \{d\}$$

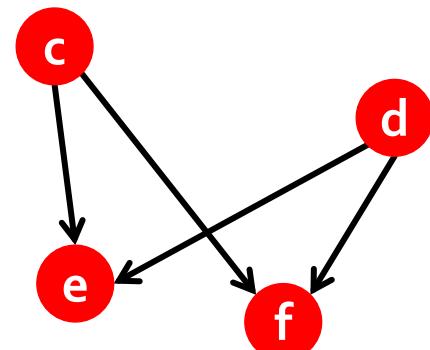
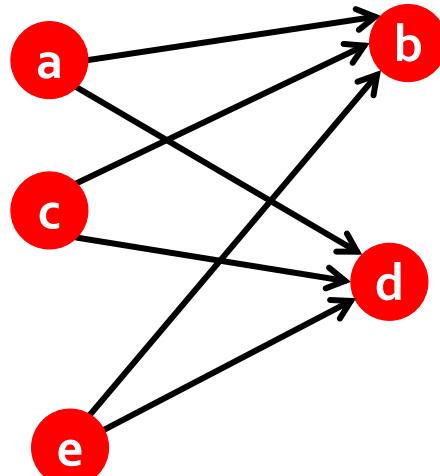
$$c = \{b, d, e, f\}$$

$$d = \{e, f\}$$

$$e = \{b, d\}$$

$$f = \{\}$$

- **Support threshold $s=2$**
 - $\{b, d\}$: support 3
 - $\{e, f\}$: support 2
- **And we just found 2 bipartite subgraphs:**



Example (2)

■ Example of a community from a web graph

A community of Australian fire brigades

Nodes on the right

NSW Rural Fire Service Internet Site
NSW Fire Brigades
Sutherland Rural Fire Service
CFA: County Fire Authority
“The National Cente...ted Children’s Ho...
CRAFTI Internet Connexions-INFO
Welcome to Blackwoo... Fire Safety Serv...
The World Famous Guestbook Server
Wilberforce County Fire Brigade
NEW SOUTH WALES FIR...ES 377 STATION
Woronora Bushfire Brigade
Mongarlowe Bush Fire – Home Page
Golden Square Fire Brigade
FIREBREAK Home Page
Guises Creek Volunt...fficial Home Page...

Nodes on the left

New South Wales Fir...ial Australian Links
Feuerwehrlinks Australien
FireNet Information Network
The Cherrybrook Rur...re Brigade Home Page
New South Wales Fir...ial Australian Links
Fire Departments, F... Information Network
The Australian Firefighter Page
Kristiansand brannv...dens brannvesener...
Australian Fire Services Links
The 911 F,P,M., Fir...mp; Canada A Section
Feuerwehrlinks Australien
Sanctuary Point Rural Fire Brigade
Fire Trails “l...ghters around the...
FireSafe – Fire and Safety Directory
Kristiansand Firede...departments of th...

More Stream Mining

Bloom Filters

Sampling Streams

Counting Distinct Items

Computing Moments

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Filtering Stream Content

- To motivate the Bloom-filter idea, consider a web crawler.
- It keeps, centrally, a list of all the URL's it has found so far.
- It assigns these URL's to any of a number of parallel tasks; these tasks stream back the URL's they find in the links they discover on a page.
- It needs to filter out those URL's it has seen before.

Role of the Bloom Filter

- A Bloom filter placed on the stream of URL's will declare that certain URL's have been seen before.
- Others will be declared new, and will be added to the list of URL's that need to be crawled.
- Unfortunately, the Bloom filter can have false positives.
 - It can declare a URL has been seen before when it hasn't.
 - But if it says "never seen," then it is truly new.

How a Bloom Filter Works

- A *Bloom filter* is an array of bits, together with a number of hash functions.
- The argument of each hash function is a stream element, and it returns a position in the array.
- Initially, all bits are 0.
- When input x arrives, we set to 1 the bits $h(x)$, for each hash function h .

Example: Bloom Filter

- Use $N = 11$ bits for our filter.
- Stream elements = integers.
- Use two hash functions:
 - $h_1(x) =$
 - Take odd-numbered bits from the right in the binary representation of x .
 - Treat it as an integer i .
 - Result is i modulo 11.
 - $h_2(x) = \text{same, but take even-numbered bits.}$

Example – Continued

Stream element	h_1	h_2	Filter contents
			000000000000
$25 = 1\textcolor{magenta}{1}001$	5	2	00\textcolor{blue}{1}00100000
$159 = \textcolor{magenta}{1}001\textcolor{magenta}{1}111$	7	0	\textcolor{blue}{1}010010\textcolor{blue}{1}000
$585 = \textcolor{magenta}{1}001001001$	9	7	101001010\textcolor{blue}{1}0

Bloom Filter Lookup

- Suppose element y appears in the stream, and we want to know if we have seen y before.
- Compute $h(y)$ for each hash function y .
- If all the resulting bit positions are 1, say we have seen y before.
- If at least one of these positions is 0, say we have not seen y before.

Example: Lookup

- Suppose we have the same Bloom filter as before, and we have set the filter to 10100101010.
- Lookup element $y = 118 = 1110110$ (binary).
- $h_1(y) = 14$ modulo 11 = 3.
- $h_2(y) = 5$ modulo 11 = 5.
- Bit 5 is 1, but bit 3 is 0, so we are sure y is not in the set.

Performance of Bloom Filters

- Probability of a false positive depends on the density of 1's in the array and the number of hash functions.
 - $= (\text{fraction of 1's})^{\# \text{ of hash functions}}$.
- The number of 1's is approximately the number of elements inserted times the number of hash functions.
 - But collisions lower that number slightly.

Throwing Darts

- Turning random bits from 0 to 1 is like throwing d darts at t targets, at random.
- How many targets are hit by at least one dart?
- Probability a given target is hit by a given dart = $1/t$.
- Probability none of d darts hit a given target is $(1-1/t)^d$.
- Rewrite as $(1-1/t)^{t(d/t)} \sim= e^{-d/t}$.

Example: Throwing Darts

- Suppose we use an array of 1 billion bits, 5 hash functions, and we insert 100 million elements.
- That is, $t = 10^9$, and $d = 5 \cdot 10^8$.
- The fraction of 0's that remain will be $e^{-1/2} = 0.607$.
- Density of 1's = 0.393.
- Probability of a false positive = $(0.393)^5 = 0.00937$.

CS 188: Artificial Intelligence Spring 2008

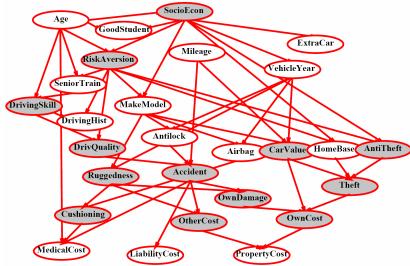
Bayes Nets
2/5/08, 2/7/08

Dan Klein – UC Berkeley

Bayes' Nets

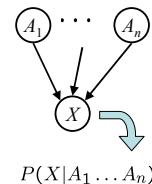
- A Bayes' net is an efficient encoding of a probabilistic model of a domain
- Questions we can ask:
 - Inference: given a fixed BN, what is $P(X | e)$?
 - Representation: given a fixed BN, what kinds of distributions can it encode?
 - Modeling: what BN is most appropriate for a given domain?

Example Bayes' Net



Bayes' Net Semantics

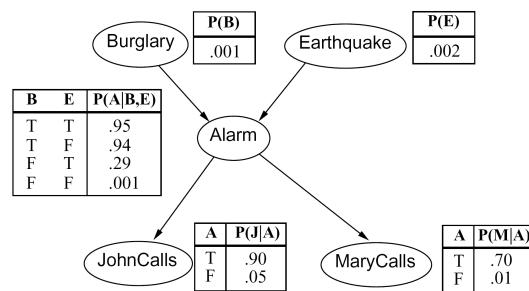
- A Bayes' net:
 - A set of nodes, one per variable X
 - A directed, acyclic graph
 - A conditional distribution of each variable conditioned on its parents (the *parameters* θ)
- Semantics:
 - A BN defines a joint probability distribution over its variables:
$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{parents}(X_i))$$



Building the (Entire) Joint

- We can take a Bayes' net and build any entry from the full joint distribution it encodes
- Typically, there's no reason to build ALL of it
 - We build what we need on the fly
- To emphasize: every BN over a domain **implicitly represents some joint distribution** over that domain, but is specified by local probabilities

Example: Alarm Network



$$P(b, e, \neg a, j, m) =$$

Size of a Bayes' Net

- How big is a joint distribution over N Boolean variables?
- How big is an N-node net if nodes have k parents?
- Both give you the power to calculate $P(X_1, X_2, \dots, X_n)$
- BNs: Huge space savings!
- Also easier to elicit local CPTs
- Also turns out to be faster to answer queries (next class)

Bayes' Nets

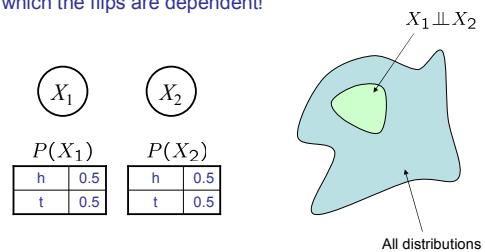
- So far: how a Bayes' net encodes a joint distribution
- Next: how to answer queries about that distribution
 - Key idea: conditional independence
 - Last class: assembled BNs using an intuitive notion of conditional independence as causality
 - Today: formalize these ideas
 - Main goal: answer queries about conditional independence and influence
- After that: how to answer numerical queries (inference)

Conditional Independence

- Reminder: independence
 - X and Y are **independent** if
 $\forall x, y \ P(x, y) = P(x)P(y) \dashrightarrow X \perp\!\!\!\perp Y$
 - X and Y are **conditionally independent** given Z
 $\forall x, y, z \ P(x, y|z) = P(x|z)P(y|z) \dashrightarrow X \perp\!\!\!\perp Y | Z$
 - (Conditional) independence is a property of a distribution

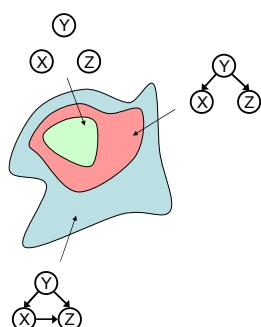
Example: Independence

- For this graph, you can fiddle with θ (the CPTs) all you want, but you won't be able to represent any distribution in which the flips are dependent!



Topology Limits Distributions

- Given some graph topology G, only certain joint distributions can be encoded
- The graph structure guarantees certain (conditional) independences
- (There might be more independence)
- Adding arcs increases the set of distributions, but has several costs

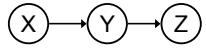


Independence in a BN

- Important question about a BN:
 - Are two nodes independent given certain evidence?
 - If yes, can calculate using algebra (really tedious)
 - If no, can prove with a counter example
 - Example:
 
- Question: are X and Z independent?
 - Answer: not *necessarily*, we've seen examples otherwise: low pressure causes rain which causes traffic.
 - X can influence Z, Z can influence X (via Y)
 - Addendum: they *could* be independent: how?

Causal Chains

- This configuration is a “causal chain”



X: Low pressure
Y: Rain
Z: Traffic

- Is X independent of Z given Y?

$$P(z|x,y) = \frac{P(x,y,z)}{P(x,y)} = \frac{P(x)P(y|x)P(z|y)}{P(x)P(y|x)} = P(z|y) \quad \text{Yes!}$$

- Evidence along the chain “blocks” the influence

Common Cause

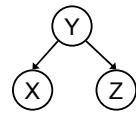
- Another basic configuration: two effects of the same cause

- Are X and Z independent?

- Are X and Z independent given Y?

$$P(z|x,y) = \frac{P(x,y,z)}{P(x,y)} = \frac{P(y)P(x|y)P(z|y)}{P(y)P(x|y)} = P(z|y)$$

Yes!



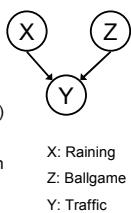
Y: Project due
X: Newsgroup busy
Z: Lab full

- Observing the cause blocks influence between effects.

Common Effect

- Last configuration: two causes of one effect (v-structures)

- Are X and Z independent?
 - Yes: remember the ballgame and the rain causing traffic, no correlation?
 - Still need to prove they must be (homework)
- Are X and Z independent given Y?
 - No: remember that seeing traffic put the rain and the ballgame in competition?
- This is backwards from the other cases**
 - Observing the effect **enables** influence between effects.



X: Raining
Z: Ballgame
Y: Traffic

The General Case

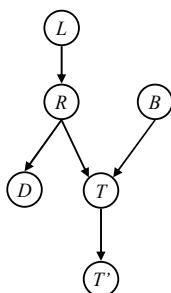
- Any complex example can be analyzed using these three canonical cases

- General question: in a given BN, are two variables independent (given evidence)?

- Solution: graph search!

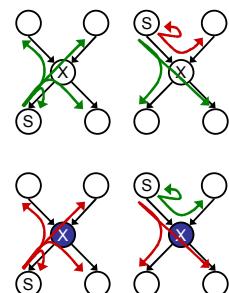
Reachability

- Recipe: shade evidence nodes
- Attempt 1: if two nodes are connected by an undirected path not blocked by a shaded node, they are conditionally independent
- Almost works, but not quite
 - Where does it break?
 - Answer: the v-structure at T doesn't count as a link in a path unless shaded



Reachability (the Bayes' Ball)

- Correct algorithm:
 - Shade in evidence
 - Start at source node
 - Try to reach target by search
- States: pair of (node X, previous state S)
- Successor function:
 - X unobserved:
 - To any child
 - To any parent if coming from a child
 - X observed:
 - From parent to parent
- If you can't reach a node, it's conditionally independent of the start node given evidence

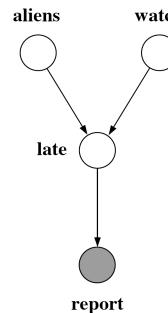


Example

$A \perp\!\!\!\perp W$

Yes

$A \perp\!\!\!\perp W|R$



Example

$L \perp\!\!\!\perp T'|T$

Yes

$L \perp\!\!\!\perp B$

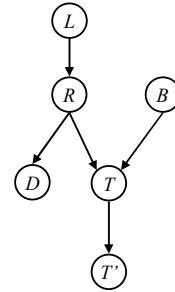
Yes

$L \perp\!\!\!\perp B|T$

$L \perp\!\!\!\perp B|T'$

$L \perp\!\!\!\perp B|T, R$

Yes



Example

Variables:

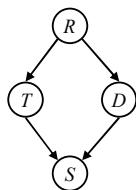
- R: Raining
- T: Traffic
- D: Roof drips
- S: I'm sad

Questions:

$T \perp\!\!\!\perp D$

$T \perp\!\!\!\perp D|R$ Yes

$T \perp\!\!\!\perp D|R, S$



Causality?

When Bayes' nets reflect the true causal patterns:

- Often simpler (nodes have fewer parents)
- Often easier to think about
- Often easier to elicit from experts

BNs need not actually be causal

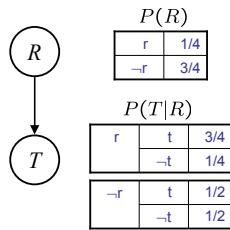
- Sometimes no causal net exists over the domain
- E.g. consider the variables *Traffic* and *Drips*
- End up with arrows that reflect correlation, not causation

What do the arrows really mean?

- Topology may happen to encode causal structure
- Topology only guaranteed to encode conditional independencies

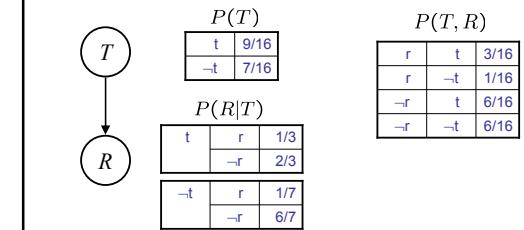
Example: Traffic

- Basic traffic net
- Let's multiply out the joint



Example: Reverse Traffic

- Reverse causality?



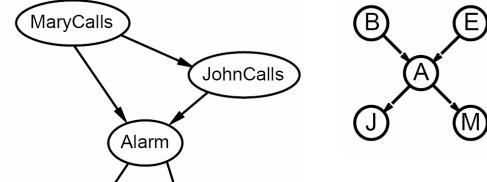
Example: Coins

- Extra arcs don't prevent representing independence, just allow non-independence

X_1	X_2
$P(X_1)$	$P(X_2)$
$\begin{array}{ c c }\hline h & 0.5 \\ \hline t & 0.5 \\ \hline\end{array}$	$\begin{array}{ c c }\hline h & 0.5 \\ \hline t & 0.5 \\ \hline\end{array}$

$P(X_1)$	$P(X_2 X_1)$
$\begin{array}{ c c }\hline h & 0.5 \\ \hline t & 0.5 \\ \hline\end{array}$	$\begin{array}{ c c c }\hline h & h & 0.5 \\ \hline h & t & 0.5 \\ \hline t & h & 0.5 \\ \hline t & t & 0.5 \\ \hline\end{array}$
$\begin{array}{ c c }\hline h & 0.5 \\ \hline t & 0.5 \\ \hline\end{array}$	$\begin{array}{ c c }\hline h & 0.5 \\ \hline t & 0.5 \\ \hline\end{array}$

Alternate BNs

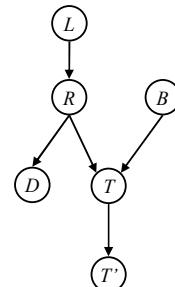


Summary

- Bayes nets compactly encode joint distributions
- Guaranteed independencies of distributions can be deduced from BN graph structure
- A Bayes' net may have other independencies that are not detectable until you inspect its specific distribution
- The Bayes' ball algorithm (aka d-separation) tells us when an observation of one variable can change belief about another variable

Inference

- Inference: calculating some statistic from a joint probability distribution
- Examples:
 - Posterior probability:
 $P(Q|E_1 = e_1, \dots, E_k = e_k)$
 - Most likely explanation:
 $\text{argmax}_q P(Q = q|E_1 = e_1, \dots)$

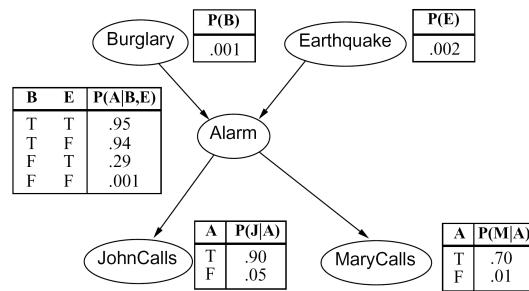


Inference by Enumeration

- $P(\text{sun})?$
- $P(\text{sun} | \text{winter})?$
- $P(\text{sun} | \text{winter, warm})?$

S	T	R	P
summer	warm	sun	0.30
summer	warm	rain	0.05
summer	cold	sun	0.10
summer	cold	rain	0.05
winter	warm	sun	0.10
winter	warm	rain	0.05
winter	cold	sun	0.15
winter	cold	rain	0.20

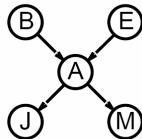
Reminder: Alarm Network



Inference by Enumeration

- Given unlimited time, inference in BNs is easy
- Recipe:**
 - State the marginal probabilities you need
 - Figure out ALL the atomic probabilities you need
 - Calculate and combine them
- Example:**

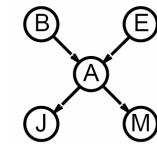
$$P(b|j, m) = \frac{P(b, j, m)}{P(j, m)}$$



Example

$$P(b|j, m) = \frac{P(b, j, m)}{P(j, m)}$$

$$\begin{aligned} P(b, j, m) &= P(b, e, a, j, m) + \\ &\quad P(b, \bar{e}, a, j, m) + \\ &\quad P(b, e, \bar{a}, j, m) + \\ &\quad P(b, \bar{e}, \bar{a}, j, m) \\ &= \sum_{e,a} P(b, e, a, j, m) \end{aligned}$$



Where did we use the BN structure?
We didn't!

Example

- In this simple method, we only need the BN to synthesize the joint entries

$$\begin{aligned} P(b, j, m) &= \\ &P(b)P(e)P(a|b, e)P(j|a)P(m|a) + \\ &P(b)P(e)P(\bar{a}|b, e)P(j|\bar{a})P(m|\bar{a}) + \\ &P(b)P(\bar{e})P(a|b, \bar{e})P(j|a)P(m|a) + \\ &P(b)P(\bar{e})P(\bar{a}|b, \bar{e})P(j|\bar{a})P(m|\bar{a}) \end{aligned}$$

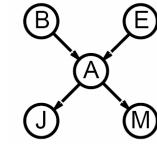
Normalization Trick

$$P(B|j, m) = \frac{P(B, j, m)}{P(j, m)}$$

$$P(b, j, m) = \sum_{e,a} P(b, e, a, j, m)$$

$$P(\bar{b}, j, m) = \sum_{e,a} P(\bar{b}, e, a, j, m)$$

$$\left[\begin{array}{c} P(b, j, m) \\ P(\bar{b}, j, m) \end{array} \right] \xrightarrow{\text{Normalize}} \left[\begin{array}{c} P(b|j, m) \\ P(\bar{b}|j, m) \end{array} \right]$$



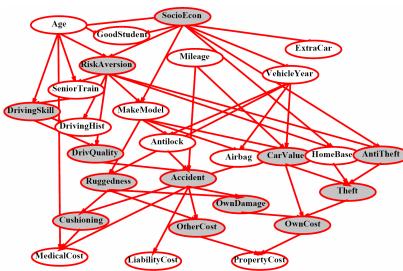
Inference by Enumeration

- General case:**
 - Evidence variables: $(E_1 \dots E_k) = (e_1 \dots e_k)$
 - Query variables: $Y_1 \dots Y_m$
 - Hidden variables: $H_1 \dots H_r$
- We want: $P(Y_1 \dots Y_m | e_1 \dots e_k)$
- First, select the entries consistent with the evidence
- Second, sum out H:

$$P(Y_1 \dots Y_m, e_1 \dots e_k) = \sum_{h_1 \dots h_r} P(Y_1 \dots Y_m, h_1 \dots h_r, e_1 \dots e_k) \underbrace{X_1, X_2, \dots, X_n}_{All variables}$$

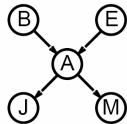
- Finally, normalize the remaining entries to conditionalize
- Obvious problems:
 - Worst-case time complexity $O(d^n)$
 - Space complexity $O(d^n)$ to store the joint distribution

Inference by Enumeration?



Nesting Sums

- Atomic inference is extremely slow!
- Slightly clever way to save work:
 - Move the sums as far right as possible
 - Example:



$$\begin{aligned}
 P(b, j, m) &= \sum_{e,a} P(b, e, a, j, m) \\
 &= \sum_{e,a} P(b)P(e)P(a|b,e)P(j|a)P(m|a) \\
 &= P(b) \sum_e P(e) \sum_a P(a|b,e)P(j|a)P(m|a)
 \end{aligned}$$

Variable Elimination: Idea

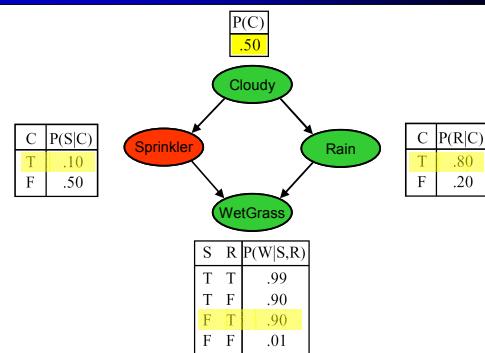
- Lots of redundant work in the computation tree
- We can save time if we cache all partial results
- This is the basic idea behind variable elimination

Sampling

- Basic idea:
 - Draw N samples from a sampling distribution S
 - Compute an approximate posterior probability
 - Show this converges to the true probability P
- Outline:
 - Sampling from an empty network
 - Rejection sampling: reject samples disagreeing with evidence
 - Likelihood weighting: use evidence to weight samples



Prior Sampling



Prior Sampling

- This process generates samples with probability

$$S_{PS}(x_1 \dots x_n) = \prod_{i=1}^n P(x_i | \text{Parents}(X_i)) = P(x_1 \dots x_n)$$

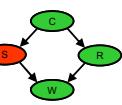
...i.e. the BN's joint probability
- Let the number of samples of an event be $N_{PS}(x_1 \dots x_n)$
- Then $\lim_{N \rightarrow \infty} \bar{P}(x_1, \dots, x_n) = \lim_{N \rightarrow \infty} N_{PS}(x_1, \dots, x_n)/N = S_{PS}(x_1, \dots, x_n) = P(x_1 \dots x_n)$
- I.e., the sampling procedure is consistent

Example

- We'll get a bunch of samples from the BN:
 - C, $\neg S$, r, w
 - C, s, r, w
 - $\neg C$, s, r, $\neg w$
 - C, $\neg S$, r, w
 - $\neg C$, s, $\neg r$, w
- If we want to know $P(W)$
 - We have counts <w:4, $\neg w:1$ >
 - Normalize to get $P(W) = <w:0.8, \neg w:0.2>$
 - This will get closer to the true distribution with more samples
 - Can estimate anything else, too
 - What about $P(C|\neg r)$? $P(C|r, \neg w)$?

Rejection Sampling

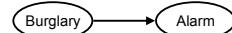
- Let's say we want $P(C)$
 - No point keeping all samples around
 - Just tally counts of C outcomes
- Let's say we want $P(C|s)$
 - Same thing: tally C outcomes, but ignore (reject) samples which don't have $S=s$
 - This is rejection sampling
 - It is also consistent (correct in the limit)



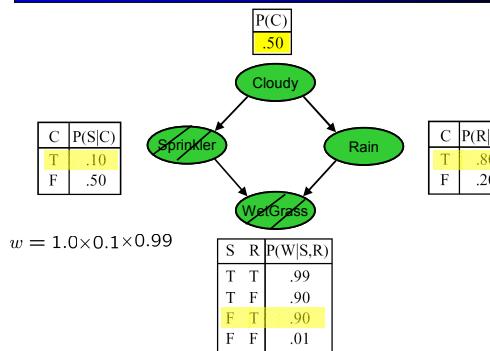
$C, \neg S, r, w$
 C, S, r, w
 $\neg C, S, r, \neg w$
 $C, \neg S, r, w$
 $\neg C, S, \neg r, w$

Likelihood Weighting

- Problem with rejection sampling:
 - If evidence is unlikely, you reject a lot of samples
 - You don't exploit your evidence as you sample
 - Consider $P(B|a)$
- Idea: fix evidence variables and sample the rest
- Problem: sample distribution not consistent!
- Solution: weight by probability of evidence given parents



Likelihood Sampling



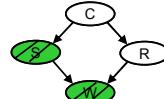
Likelihood Weighting

- Sampling distribution if z sampled and e fixed evidence

$$S_{WS}(z, e) = \prod_{i=1}^t P(z_i | \text{Parents}(Z_i))$$
- Now, samples have weights

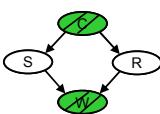
$$w(z, e) = \prod_{i=1}^m P(e_i | \text{Parents}(E_i))$$
- Together, weighted sampling distribution is consistent

$$S_{WS}(z, e)w(z, e) = \prod_{i=1}^m P(e_i | \text{Parents}(E_i)) \prod_{i=1}^m P(e_i | \text{Parents}(E_i)) \\ = P(z, e)$$



Likelihood Weighting

- Note that likelihood weighting doesn't solve all our problems
- Rare evidence is taken into account for downstream variables, but not upstream ones
- A better solution is Markov-chain Monte Carlo (MCMC), more advanced
- We'll return to sampling for robot localization and tracking in dynamic BNs



The A-Priori Algorithm

Monotonicity of “Frequent”
Candidate Pairs
Extension to Larger Itemsets

A-Priori Algorithm

- A two-pass approach called *a-priori* limits the need for main memory.
- Key idea: *monotonicity*: if a set of items appears at least s times, so does every subset of s .
- *Contrapositive for pairs*: if item i does not appear in s baskets, then no pair including i can appear in s baskets.

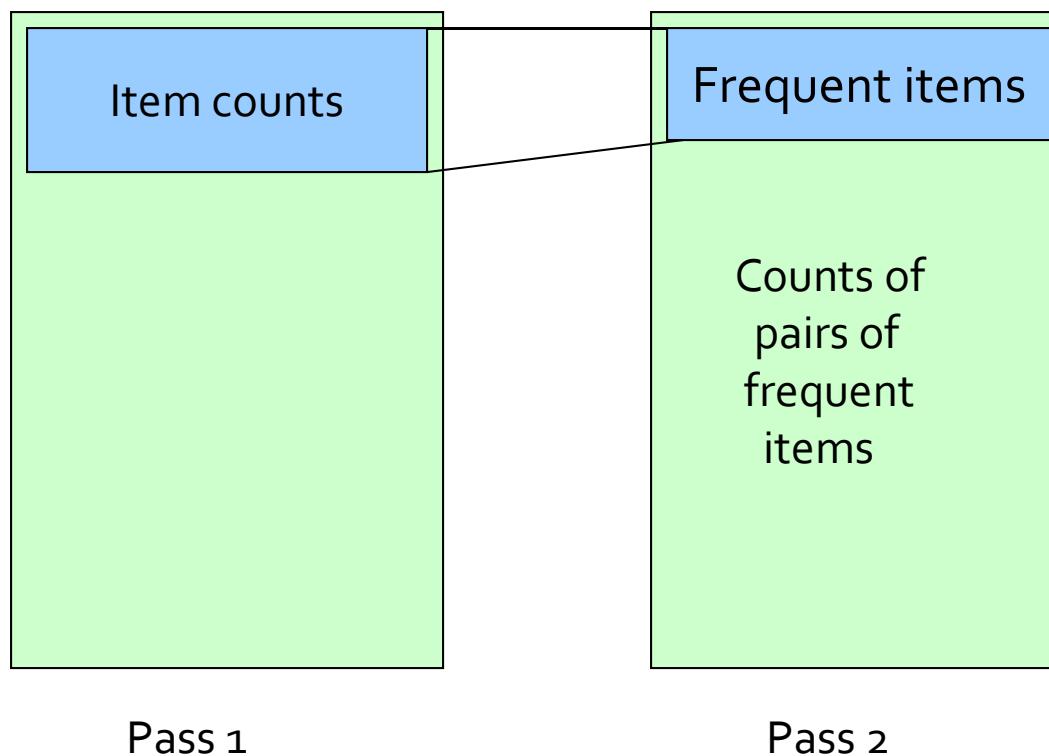
A-Priori Algorithm – (2)

- Pass 1: Read baskets and count in main memory the occurrences of each item.
 - Requires only memory proportional to #items.
- Items that appear at least s times are the *frequent items*.

A-Priori Algorithm – (3)

- Pass 2: Read baskets again and count in main memory only those pairs both of which were found in Pass 1 to be frequent.
- Requires memory proportional to square of *frequent* items only (for counts), plus a list of the frequent items (so you know what must be counted).

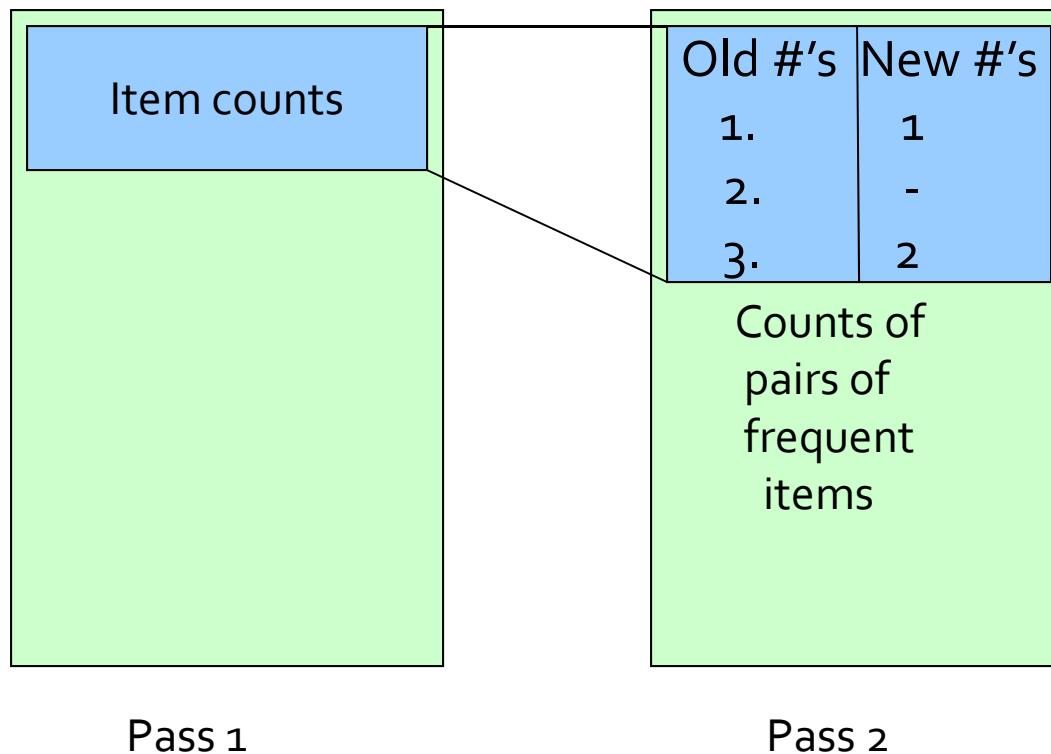
Picture of A-Priori



Detail for A-Priori

- You can use the triangular matrix method with $n = \text{number of frequent items}$.
 - May save space compared with storing triples.
- Trick: number frequent items 1,2,... and keep a table relating new numbers to original item numbers.

A-Priori Using Triangular Matrix

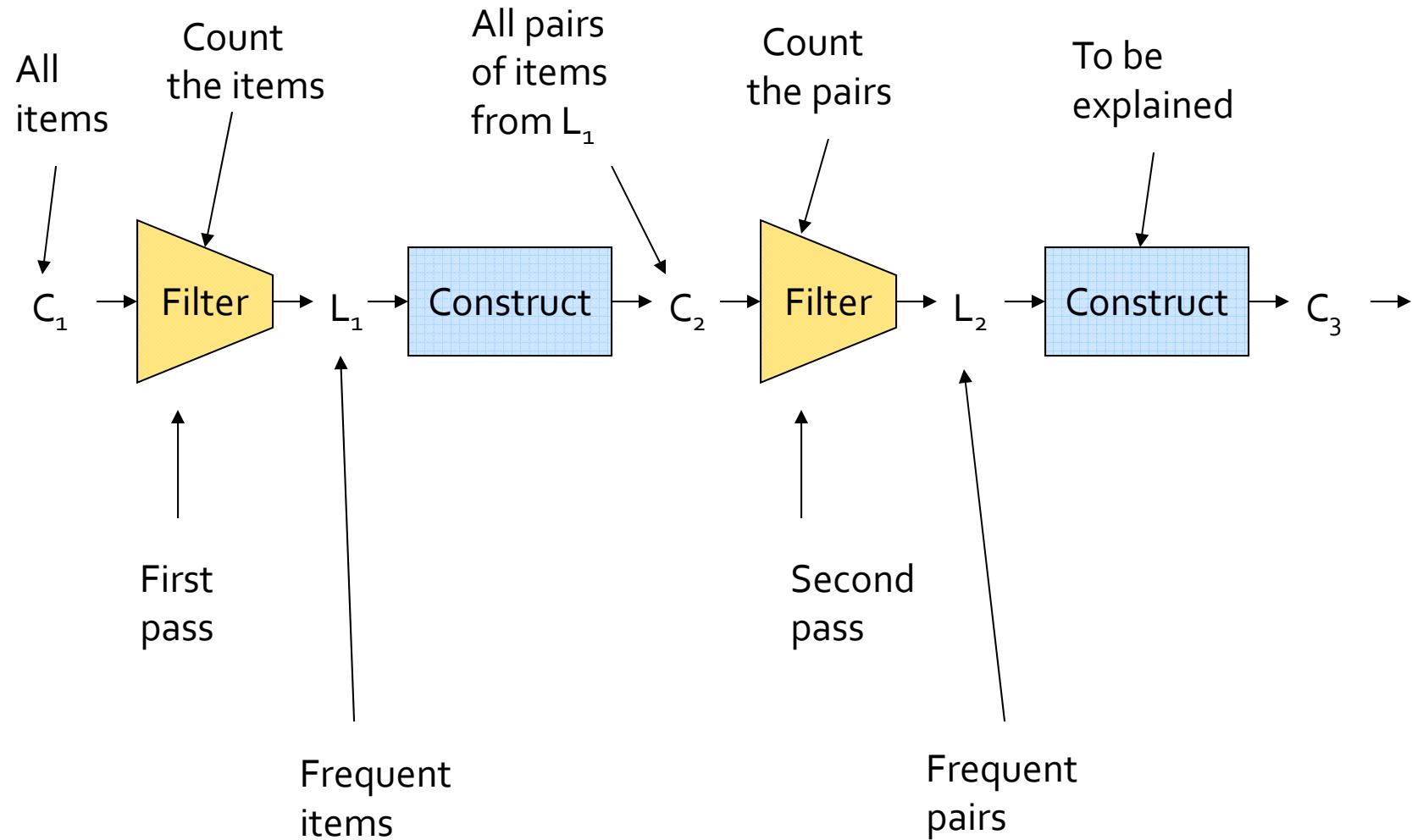


Pass 1

Pass 2

Frequent Triples, Etc.

- For each k , we construct two sets of *k-sets* (sets of size k):
 - C_k = *candidate k-sets* = those that might be frequent sets ($\text{support} \geq s$) based on information from the pass for $k - 1$.
 - L_k = the set of truly frequent k -sets.



Passes Beyond Two

- C_1 = all items
- In general, L_k = members of C_k with support $\geq s$.
 - Requires one pass.
- C_{k+1} = $(k+1)$ -sets, each k of which is in L_k .

Memory Requirements

- At the k^{th} pass, you need space to count each member of C_k .
- In realistic cases, because you need fairly high support, the number of candidates of each size drops, once you get beyond pairs.

All (Or Most) Frequent Itemsets In ≤ 2 Passes

Simple Algorithm

Savasere-Omiecinski- Navathe (SON)

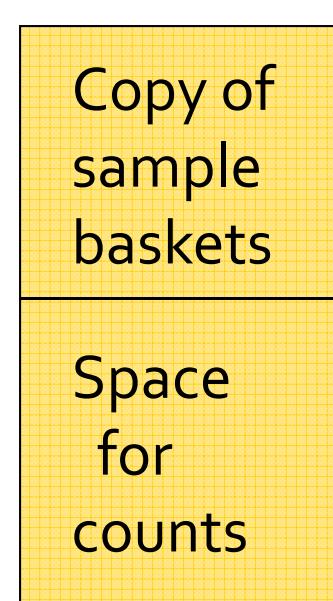
Algorithm

Toivonen's Algorithm

Simple Algorithm

- Take a random sample of the market baskets.
- Run a-priori or one of its improvements (for sets of all sizes, not just pairs) in main memory, so you don't pay for disk I/O each time you increase the size of itemsets.
- Use as your support threshold a suitable, scaled-back number.
 - **Example:** if your sample is $1/100$ of the baskets, use $s/100$ as your support threshold instead of s .

Main-Memory Picture



Simple Algorithm – Option

- Optionally, verify that your guesses are truly frequent in the entire data set by a second pass.
- But you don't catch sets frequent in the whole but not in the sample.
 - Smaller threshold, e.g., $s/125$ instead of $s/100$, helps catch more truly frequent itemsets.
 - But requires more space.

SON Algorithm

- Repeatedly read small subsets of the baskets into main memory and perform the first pass of the simple algorithm on each subset.
- An itemset becomes a candidate if it is found to be frequent in *any* one or more subsets of the baskets.

SON Algorithm – Pass 2

- On a second pass, count all the candidate itemsets and determine which are frequent in the entire set.
- Key “monotonicity” idea: an itemset cannot be frequent in the entire set of baskets unless it is frequent in at least one subset.

SON Algorithm – Distributed Version

- This idea lends itself to distributed data mining.
- If baskets are distributed among many nodes, compute *local* frequent itemsets at each node, then distribute the candidates from each node.
- Each node counts all the candidate itemsets.
- Finally, accumulate the counts of all candidates.

Toivonen's Algorithm

- Start as in the simple algorithm, but lower the threshold slightly for the sample.
 - **Example:** if the sample is 1% of the baskets, use $s/125$ as the support threshold rather than $s/100$.
 - Goal is to avoid missing any itemset that is frequent in the full set of baskets.

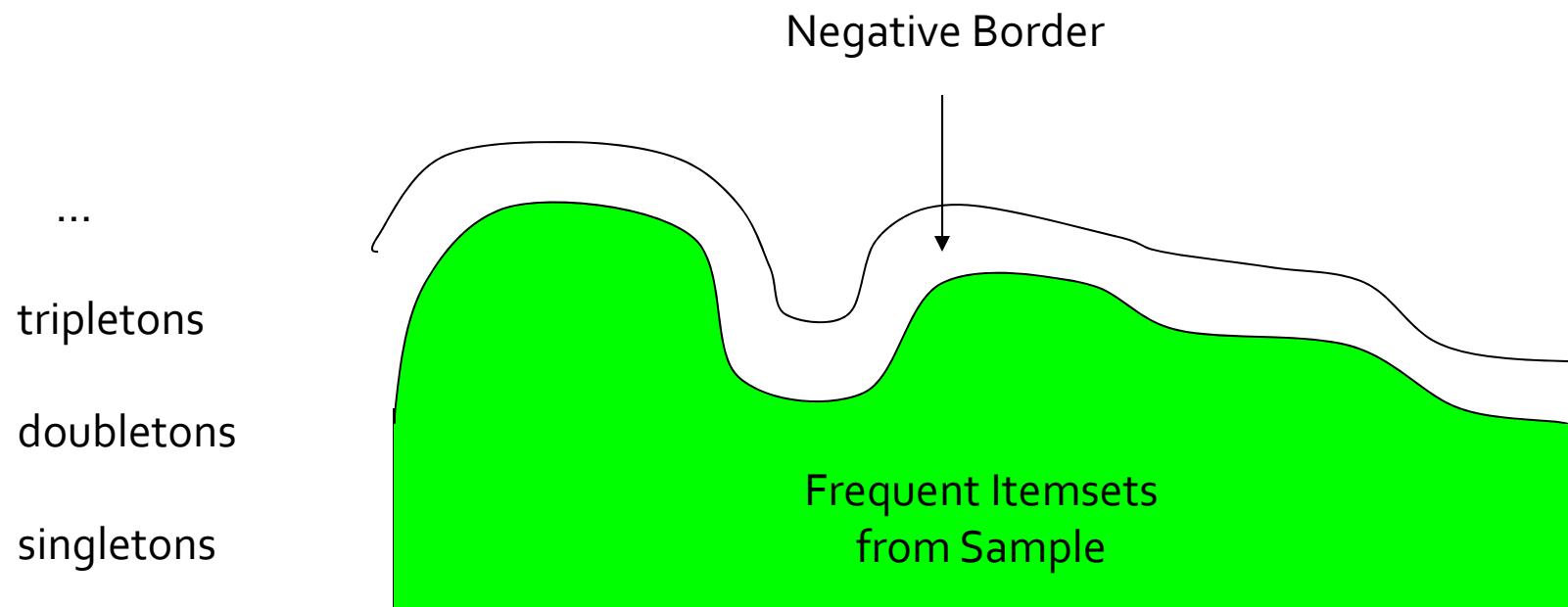
Toivonen's Algorithm – (2)

- Add to the itemsets that are frequent in the sample the *negative border* of these itemsets.
- An itemset is in the negative border if it is not deemed frequent in the sample, but *all* its immediate subsets are.

Example: Negative Border

- $\{A,B,C,D\}$ is in the negative border if and only if:
 1. It is not frequent in the sample, but
 2. All of $\{A,B,C\}$, $\{B,C,D\}$, $\{A,C,D\}$, and $\{A,B,D\}$ are.
- $\{A\}$ is in the negative border if and only if it is not frequent in the sample.
 - Because the empty set is always frequent.
 - Unless there are fewer baskets than the support threshold (silly case).

Picture of Negative Border



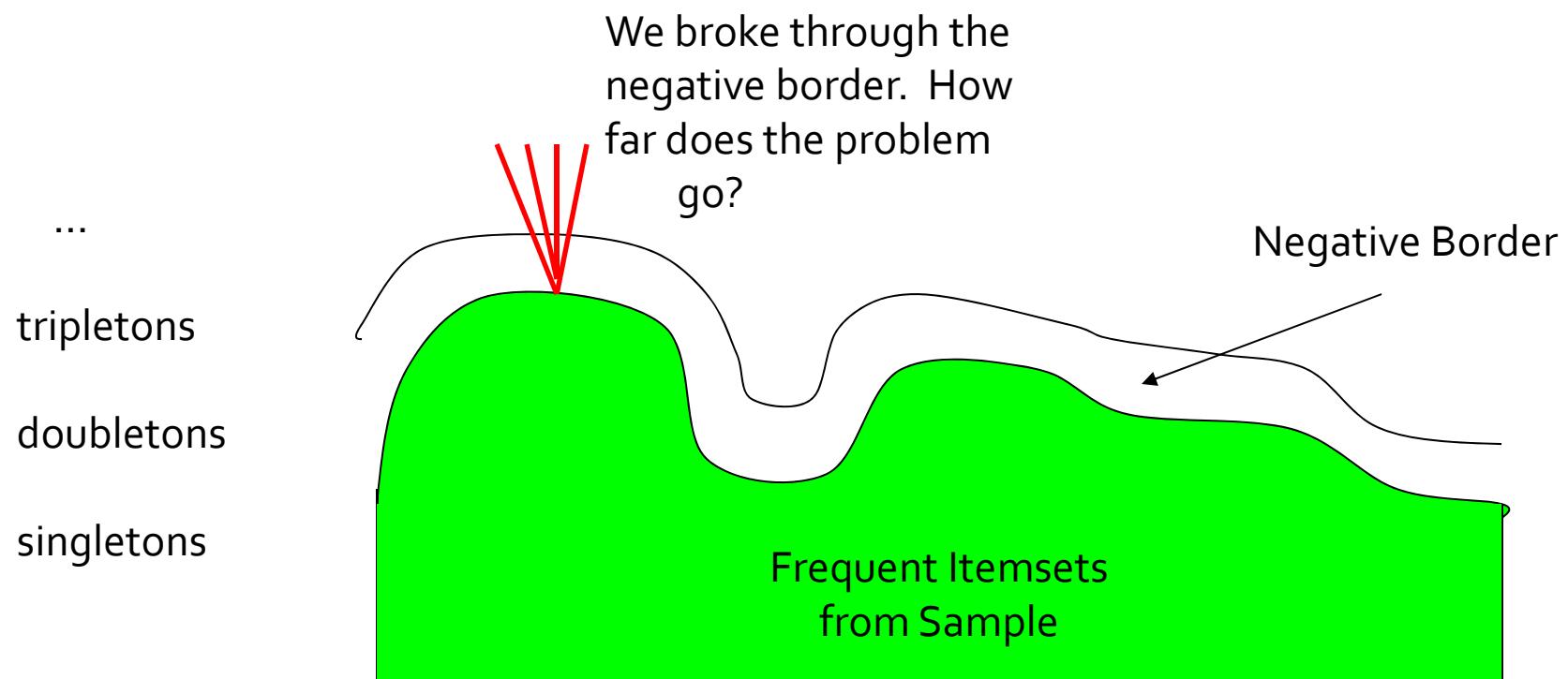
Toivonen's Algorithm – (3)

- In a second pass, count all candidate frequent itemsets from the first pass, and also count their negative border.
- If no itemset from the negative border turns out to be frequent, then the candidates found to be frequent in the whole data are *exactly* the frequent itemsets.

Toivonen's Algorithm – (4)

- What if we find that something in the negative border is actually frequent?
- We must start over again with another sample!
- Try to choose the support threshold so the probability of failure is low, while the number of itemsets checked on the second pass fits in main-memory.

If Something in the Negative Border Is Frequent . . .



Theorem:

- If there is an itemset that is frequent in the whole, but not frequent in the sample, then there is a member of the negative border for the sample that is frequent in the whole.

Proof:

- Suppose not; i.e.;
 1. There is an itemset S frequent in the whole but not frequent in the sample, and
 2. Nothing in the negative border is frequent in the whole.
- Let T be a **smallest** subset of S that is not frequent in the sample.
- T is frequent in the whole (S is frequent + monotonicity).
- T is in the negative border (else not “smallest”).

Frequent Itemsets

The Market-Basket Model
Association Rules
A-Priori Algorithm

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



The Market-Basket Model

- A large set of *items*, e.g., things sold in a supermarket.
- A large set of *baskets*, each of which is a small set of the items, e.g., the things one customer buys on one day.

Support

- Simplest question: find sets of items that appear “frequently” in the baskets.
- *Support* for itemset I = the number of baskets containing all items in I .
 - Sometimes given as a percentage.
- Given a *support threshold* s , sets of items that appear in at least s baskets are called *frequent itemsets*.

Example: Frequent Itemsets

- Items={milk, coke, pepsi, beer, juice}.
- Support = 3 baskets.

$$B_1 = \{m, c, b\}$$

$$B_3 = \{m, b\}$$

$$B_5 = \{m, p, b\}$$

$$B_7 = \{c, b, j\}$$

$$B_2 = \{m, p, j\}$$

$$B_4 = \{c, j\}$$

$$B_6 = \{m, c, b, j\}$$

$$B_8 = \{b, c\}$$

- Frequent itemsets: $\{m\}, \{c\}, \{b\}, \{j\}, \{m, b\}, \{b, c\}, \{c, j\}$.

Applications

- Items = products; baskets = sets of products someone bought in one trip to the store.
- Example application: given that many people buy beer and diapers together:
 - Run a sale on diapers; raise price of beer.
 - Only useful if many buy diapers & beer.
 - Essential for brick-and-mortar stores, not on-line stores.

Applications – (2)

- Baskets = sentences; items = documents containing those sentences.
- Items that appear together too often could represent plagiarism.
- Notice items do not have to be “in” baskets.
 - But it is better if baskets have small numbers of items, while items can be in large numbers of baskets.

Applications – (3)

- Baskets = documents; items = words.
- Unusual words appearing together in a large number of documents, e.g., “Brad” and “Angelina,” may indicate an interesting relationship.

Scale of the Problem

- WalMart sells 100,000 items and can store billions of baskets.
- The Web has billions of words and many billions of pages.

Association Rules

- If-then rules about the contents of baskets.
- $\{i_1, i_2, \dots, i_k\} \rightarrow j$ means: “if a basket contains all of i_1, \dots, i_k then it is *likely* to contain j .”
- *Confidence* of this association rule is the probability of j given i_1, \dots, i_k .
 - That is, the fraction of the baskets with i_1, \dots, i_k that also contain j .

Example: Confidence

$$\begin{array}{ll} + & B_1 = \{m, c, b\} \qquad B_2 = \{m, p, j\} \\ - & B_3 = \{m, b\} \qquad B_4 = \{c, j\} \\ - & B_5 = \{m, p, b\} \qquad + B_6 = \{m, c, b, j\} \\ & B_7 = \{c, b, j\} \qquad B_8 = \{b, c\} \end{array}$$

- An association rule: $\{m, b\} \rightarrow c$.
 - Confidence = $2/4 = 50\%$.

Finding Association Rules

- Question: “find all association rules with support $\geq s$ and confidence $\geq c$.”
 - Note: “support” of an association rule is the support of the set of items on the left.
- Hard part: finding the frequent itemsets.
 - Note: if $\{i_1, i_2, \dots, i_k\} \rightarrow j$ has high support and confidence, then both $\{i_1, i_2, \dots, i_k\}$ and $\{i_1, i_2, \dots, i_k, j\}$ will be “frequent.”

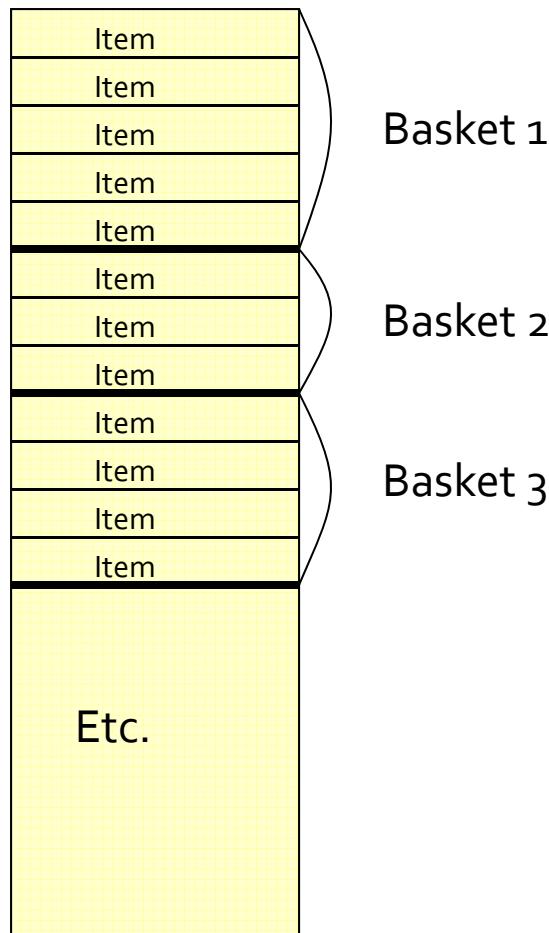
Finding Association Rules – (2)

1. Find all sets with support at least cs .
2. Find all sets with support at least s .
3. If $\{i_1, i_2, \dots, i_k, j\}$ has support at least cs , see which subsets missing one element have support at least s .
 - Take j to be the missing element.
4. $\{i_1, i_2, \dots, i_k\} \rightarrow j$ is an acceptable association rule if $\{i_1, i_2, \dots, i_k\}$ has support $s_1 \geq s$, $\{i_1, i_2, \dots, i_k, j\}$ has support $s_2 \geq cs$, and s_2/s_1 , the confidence of the rule, is at least c .

Computation Model

- Typically, data is kept in flat files.
- Stored on disk.
- Stored basket-by-basket.
- Expand baskets into pairs, triples, etc. as you read baskets.
 - Use k nested loops to generate all sets of size k .

File Organization



Example: items are positive integers, and boundaries between baskets are -1 .

Computation Model – (2)

- The true cost of mining disk-resident data is usually the **number of disk I/O's**.
- In practice, algorithms for finding frequent itemsets read the data in *passes* – all baskets read in turn.
- Thus, we measure the cost by the **number of passes** an algorithm takes.

Main-Memory Bottleneck

- For many frequent-itemset algorithms, main memory is the critical resource.
- As we read baskets, we need to count something, e.g., occurrences of pairs.
- The number of different things we can count is limited by main memory.
- Swapping counts in/out is a disaster.

Finding Frequent Pairs

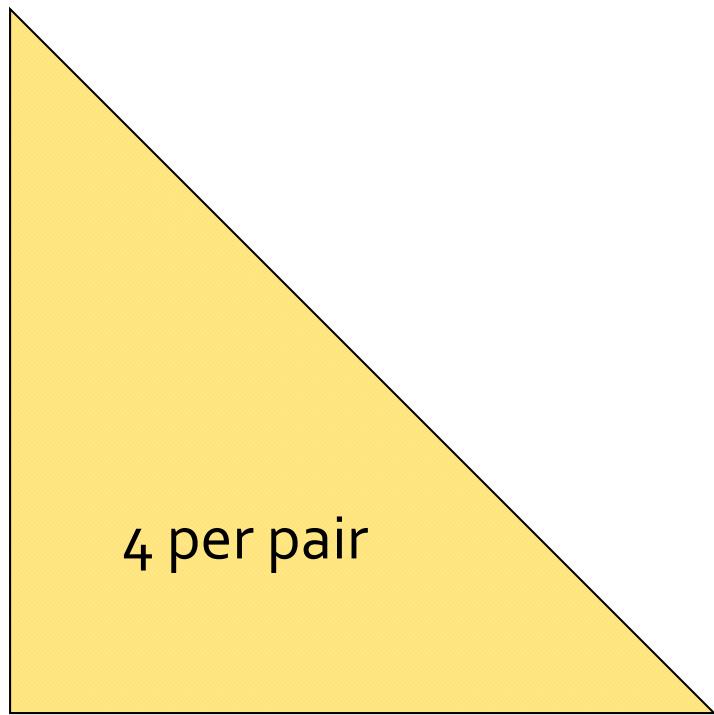
- The hardest problem often turns out to be finding the **frequent pairs**.
 - **Why?** Often frequent pairs are common, frequent triples are rare.
 - **Why?** Support threshold is usually set high enough that you don't get too many frequent itemsets.
- We'll concentrate on pairs, then extend to larger sets.

Naïve Algorithm

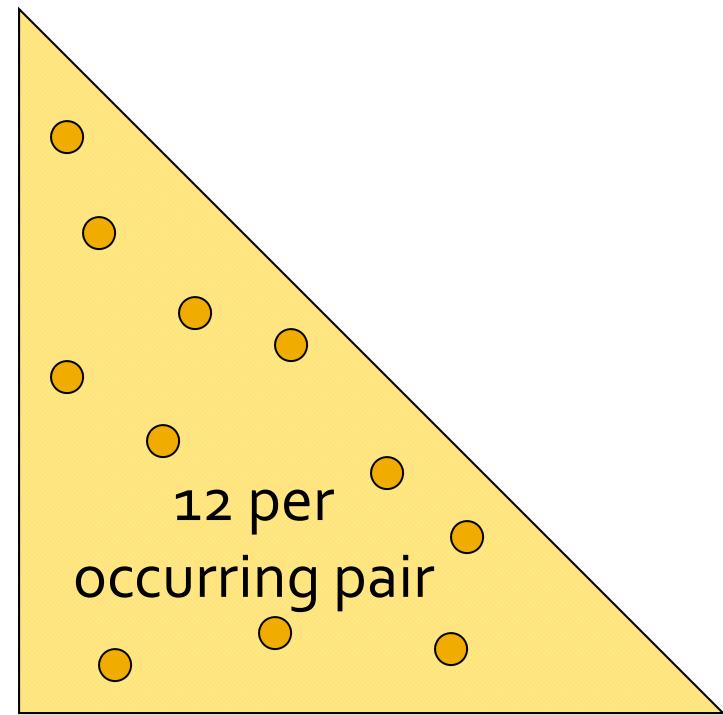
- Read file once, counting in main memory the occurrences of each pair.
 - From each basket of n items, generate its $n(n-1)/2$ pairs by two nested loops.
- Fails if $(\#items)^2$ exceeds main memory.
 - Remember: #items can be 100K (Wal-Mart) or 100B (Web pages).

Details of Main-Memory Counting

- Two approaches:
 1. Count all pairs, using a triangular matrix.
 2. Keep a table of triples $[i, j, c] = \text{"the count of the pair of items } \{i, j\} \text{ is } c\text{"}$
- (1) requires only 4 bytes/pair.
 - Note: always assume integers are 4 bytes.
 - (2) requires 12 bytes, but only for those pairs with count > 0.



Triangular matrix



Tabular method

Triangular-Matrix Approach

- Number items 1, 2, ...
 - Requires table of size $O(n)$ to convert item names to consecutive integers.
- Count $\{i, j\}$ only if $i < j$.
- Keep pairs in the order $\{1,2\}, \{1,3\}, \dots, \{1,n\}$, $\{2,3\}, \{2,4\}, \dots, \{2,n\}$, $\{3,4\}, \dots, \{3,n\}, \dots, \{n-1,n\}$.

Triangular-Matrix Approach – (2)

- Find pair $\{i, j\}$, where $i < j$, at the position:
$$(i - 1)(n - i/2) + j - i$$
- Total number of pairs $n(n - 1)/2$; total bytes about $2n^2$.

Details of Tabular Approach

- Total bytes used is about $12p$, where p is the number of pairs that actually occur.
 - Beats triangular matrix if at most $1/3$ of possible pairs actually occur.
- May require extra space for retrieval structure, e.g., a hash table.

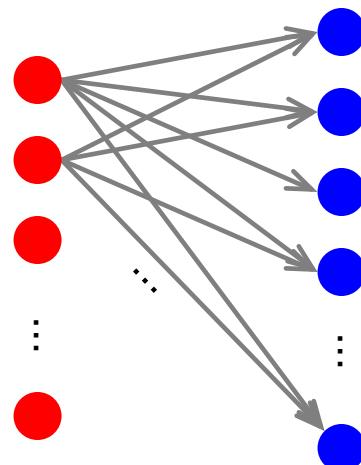
Analysis of Large Graphs: Trawling

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Trawling

- Searching for small communities in the Web graph
- What is the signature of a community / discussion in a Web graph?



Dense 2-layer graph

Use this to define “topics”:
What the same people on the left talk about on the right
Remember HITS!

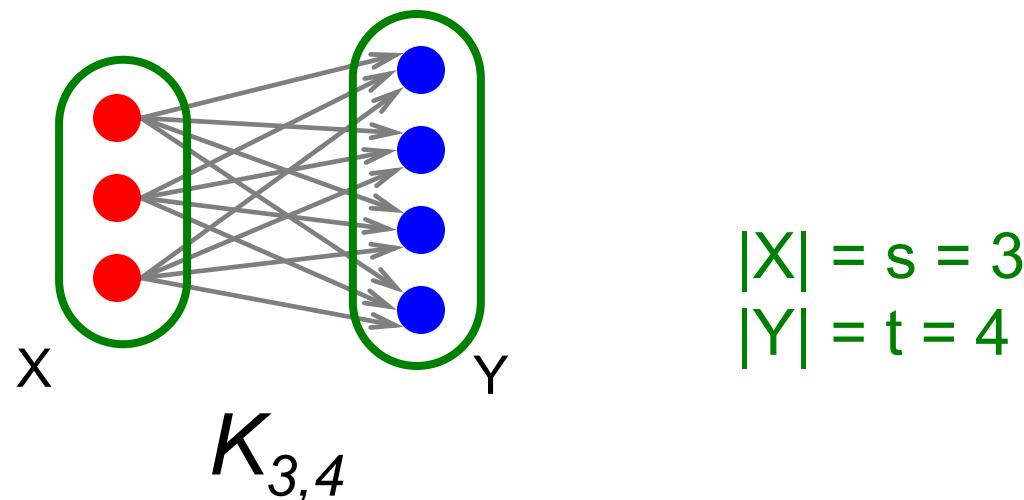
Intuition: Many people all talking about the same things

Searching for Small Communities

- **A more well-defined problem:**

- Enumerate complete bipartite subgraphs $K_{s,t}$

- Where $K_{s,t} : s$ nodes on the “left” where each links to the same t other nodes on the “right”



Fully connected

Frequent Itemset Enumeration

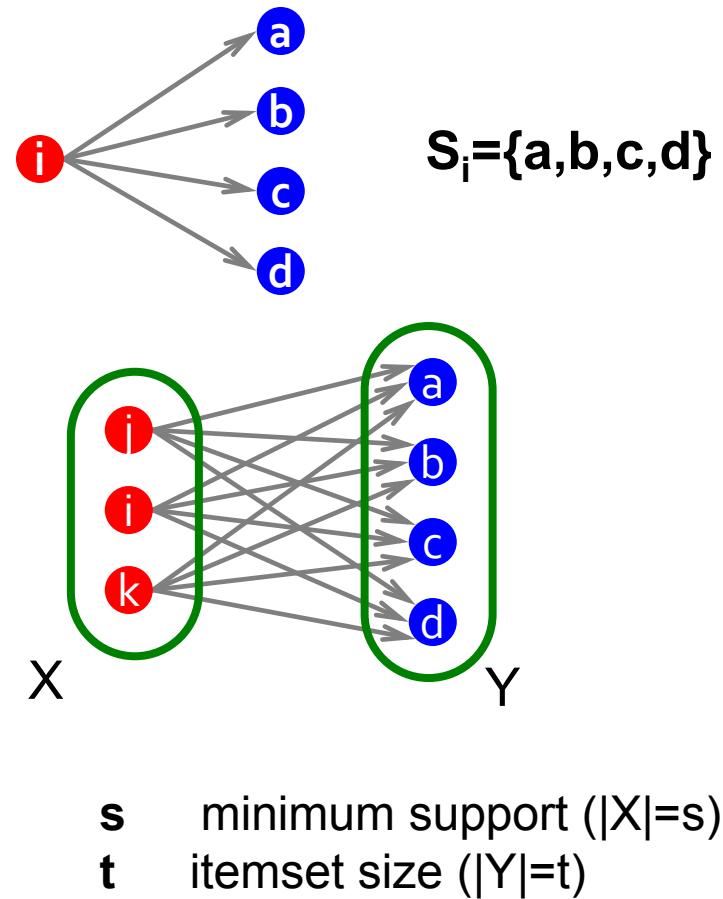
- **Market basket analysis.** Setting:
 - **Market:** Universe U of n items
 - **Baskets:** m subsets of U : $S_1, S_2, \dots, S_m \subseteq U$
(S_i is a set of items one person bought)
 - **Support:** Frequency threshold f
- **Goal:**
 - Find all subsets T s.t. $T \subseteq S_i$ of at least f sets S_i
(items in T were bought together at least f times)
- **What's the connection between the itemsets and complete bipartite graphs?**

From Itemsets to Bipartite $K_{s,t}$

Frequent itemsets = complete bipartite graphs!

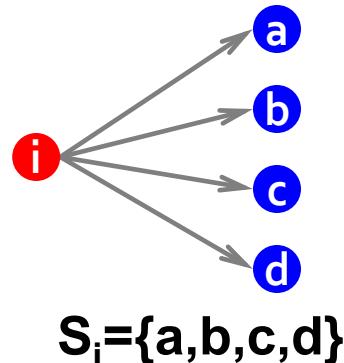
- How?

- View each node i as a set S_i of nodes i points to
- $K_{s,t} =$ a set Y of size t that occurs in s sets S_i
- Looking for $K_{s,t} \rightarrow$ set of frequency threshold to s and look at layer t – all frequent sets of size t



From Itemsets to Bipartite $K_{s,t}$

View each node i as a set S_i of nodes i points to



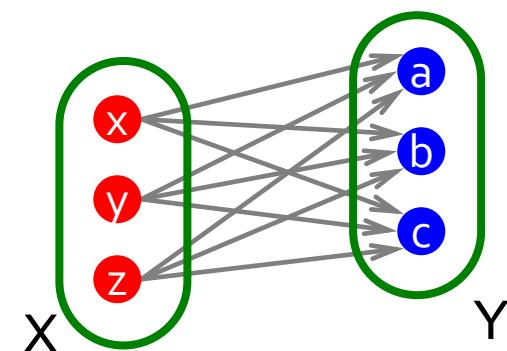
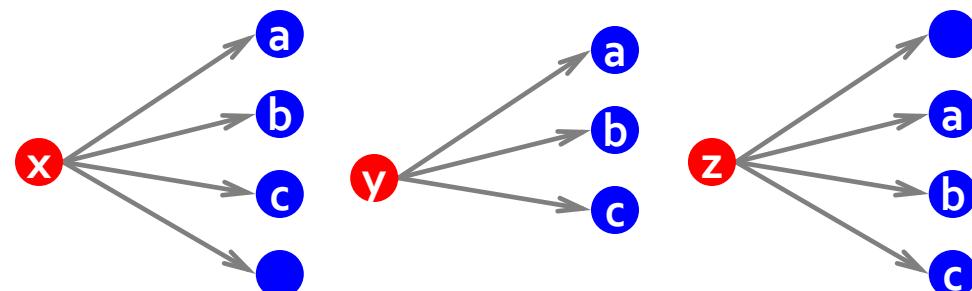
Find frequent itemsets:

s minimum support
t itemset size

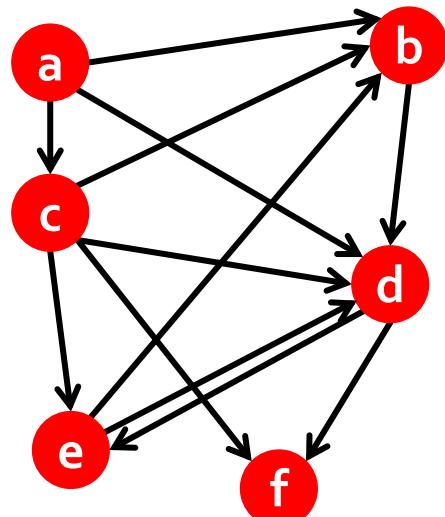
We found $K_{s,t}$!

$K_{s,t}$ = a set Y of size t that occurs in s sets S_i

Say we find a **frequent itemset** $Y=\{a,b,c\}$ of supp s
So, there are s nodes that link to all of $\{a,b,c\}$:



Example (1)



Itemsets:

$$a = \{b, c, d\}$$

$$b = \{d\}$$

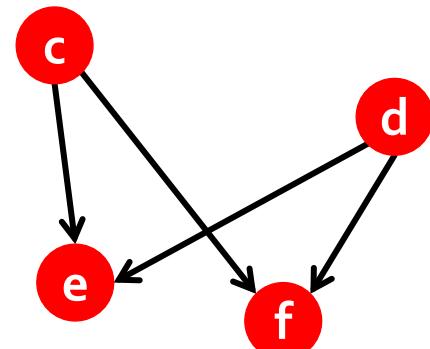
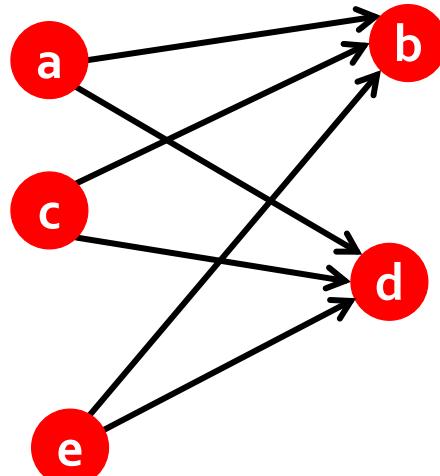
$$c = \{b, d, e, f\}$$

$$d = \{e, f\}$$

$$e = \{b, d\}$$

$$f = \{\}$$

- **Support threshold $s=2$**
 - $\{b, d\}$: support 3
 - $\{e, f\}$: support 2
- **And we just found 2 bipartite subgraphs:**



Example (2)

■ Example of a community from a web graph

A community of Australian fire brigades

Nodes on the right

NSW Rural Fire Service Internet Site
NSW Fire Brigades
Sutherland Rural Fire Service
CFA: County Fire Authority
“The National Cente...ted Children’s Ho...
CRAFTI Internet Connexions-INFO
Welcome to Blackwoo... Fire Safety Serv...
The World Famous Guestbook Server
Wilberforce County Fire Brigade
NEW SOUTH WALES FIR...ES 377 STATION
Woronora Bushfire Brigade
Mongarlowe Bush Fire – Home Page
Golden Square Fire Brigade
FIREBREAK Home Page
Guises Creek Volunt...fficial Home Page...

Nodes on the left

New South Wales Fir...ial Australian Links
Feuerwehrlinks Australien
FireNet Information Network
The Cherrybrook Rur...re Brigade Home Page
New South Wales Fir...ial Australian Links
Fire Departments, F... Information Network
The Australian Firefighter Page
Kristiansand brannv...dens brannvesener...
Australian Fire Services Links
The 911 F,P,M., Fir...mp; Canada A Section
Feuerwehrlinks Australien
Sanctuary Point Rural Fire Brigade
Fire Trails “l...ghters around the...
FireSafe – Fire and Safety Directory
Kristiansand Firede...departments of th...

More Stream Mining

Bloom Filters

Sampling Streams

Counting Distinct Items

Computing Moments

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Filtering Stream Content

- To motivate the Bloom-filter idea, consider a web crawler.
- It keeps, centrally, a list of all the URL's it has found so far.
- It assigns these URL's to any of a number of parallel tasks; these tasks stream back the URL's they find in the links they discover on a page.
- It needs to filter out those URL's it has seen before.

Role of the Bloom Filter

- A Bloom filter placed on the stream of URL's will declare that certain URL's have been seen before.
- Others will be declared new, and will be added to the list of URL's that need to be crawled.
- Unfortunately, the Bloom filter can have false positives.
 - It can declare a URL has been seen before when it hasn't.
 - But if it says "never seen," then it is truly new.

How a Bloom Filter Works

- A *Bloom filter* is an array of bits, together with a number of hash functions.
- The argument of each hash function is a stream element, and it returns a position in the array.
- Initially, all bits are 0.
- When input x arrives, we set to 1 the bits $h(x)$, for each hash function h .

Example: Bloom Filter

- Use $N = 11$ bits for our filter.
- Stream elements = integers.
- Use two hash functions:
 - $h_1(x) =$
 - Take odd-numbered bits from the right in the binary representation of x .
 - Treat it as an integer i .
 - Result is i modulo 11.
 - $h_2(x) = \text{same, but take even-numbered bits.}$

Example – Continued

Stream element	h_1	h_2	Filter contents
			000000000000
$25 = 1\textcolor{magenta}{1}001$	5	2	00\textcolor{blue}{1}00100000
$159 = \textcolor{magenta}{1}001\textcolor{magenta}{1}111$	7	0	\textcolor{blue}{1}010010\textcolor{blue}{1}000
$585 = \textcolor{magenta}{1}001001001$	9	7	101001010\textcolor{blue}{1}0

Bloom Filter Lookup

- Suppose element y appears in the stream, and we want to know if we have seen y before.
- Compute $h(y)$ for each hash function y .
- If all the resulting bit positions are 1, say we have seen y before.
- If at least one of these positions is 0, say we have not seen y before.

Example: Lookup

- Suppose we have the same Bloom filter as before, and we have set the filter to 10100101010.
- Lookup element $y = 118 = 1110110$ (binary).
- $h_1(y) = 14$ modulo 11 = 3.
- $h_2(y) = 5$ modulo 11 = 5.
- Bit 5 is 1, but bit 3 is 0, so we are sure y is not in the set.

Performance of Bloom Filters

- Probability of a false positive depends on the density of 1's in the array and the number of hash functions.
 - $= (\text{fraction of 1's})^{\# \text{ of hash functions}}$.
- The number of 1's is approximately the number of elements inserted times the number of hash functions.
 - But collisions lower that number slightly.

Throwing Darts

- Turning random bits from 0 to 1 is like throwing d darts at t targets, at random.
- How many targets are hit by at least one dart?
- Probability a given target is hit by a given dart = $1/t$.
- Probability none of d darts hit a given target is $(1-1/t)^d$.
- Rewrite as $(1-1/t)^{t(d/t)} \sim e^{-d/t}$.

Example: Throwing Darts

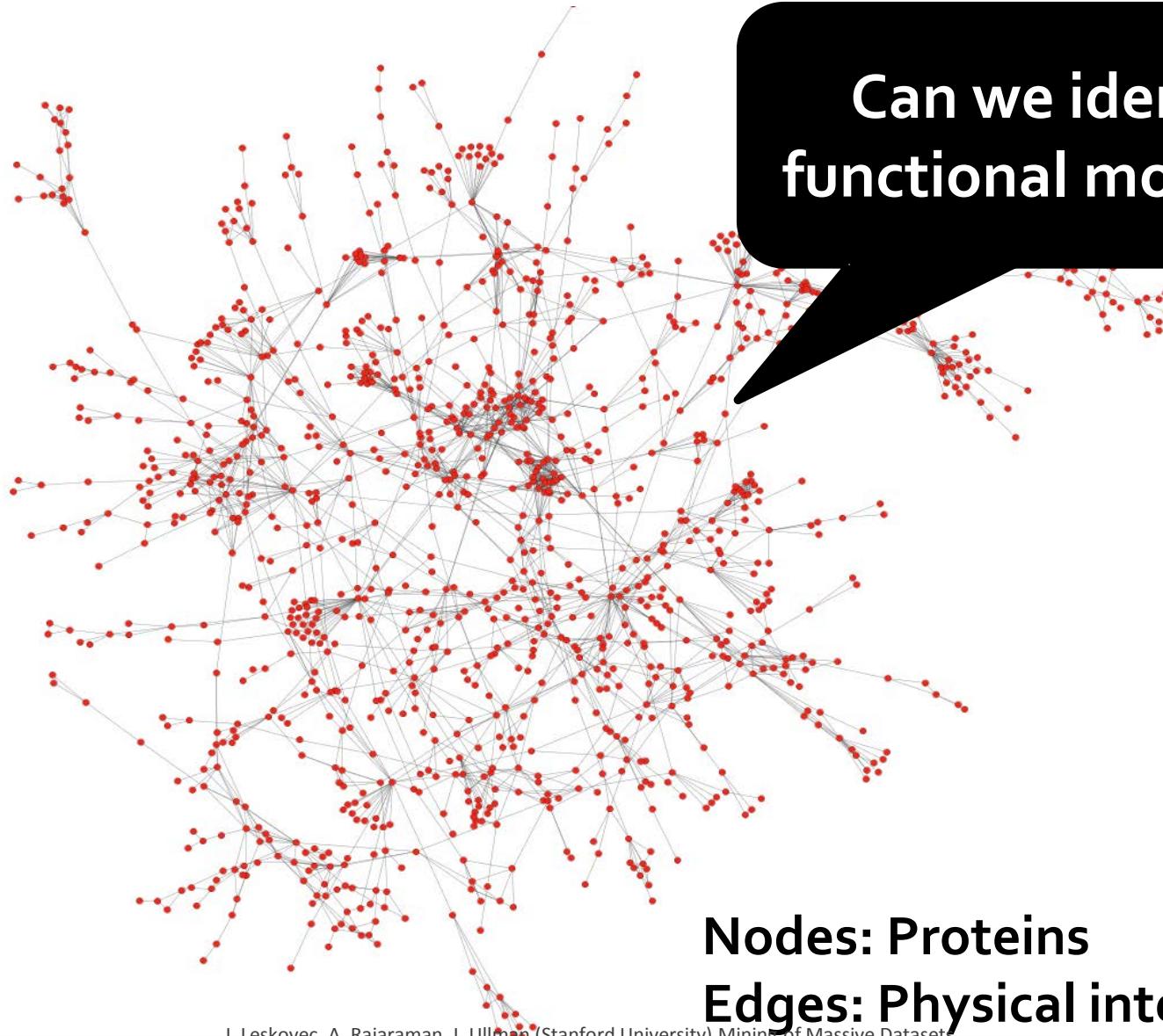
- Suppose we use an array of 1 billion bits, 5 hash functions, and we insert 100 million elements.
- That is, $t = 10^9$, and $d = 5 \cdot 10^8$.
- The fraction of 0's that remain will be $e^{-1/2} = 0.607$.
- Density of 1's = 0.393.
- Probability of a false positive = $(0.393)^5 = 0.00937$.

Community Detection in Graphs

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University

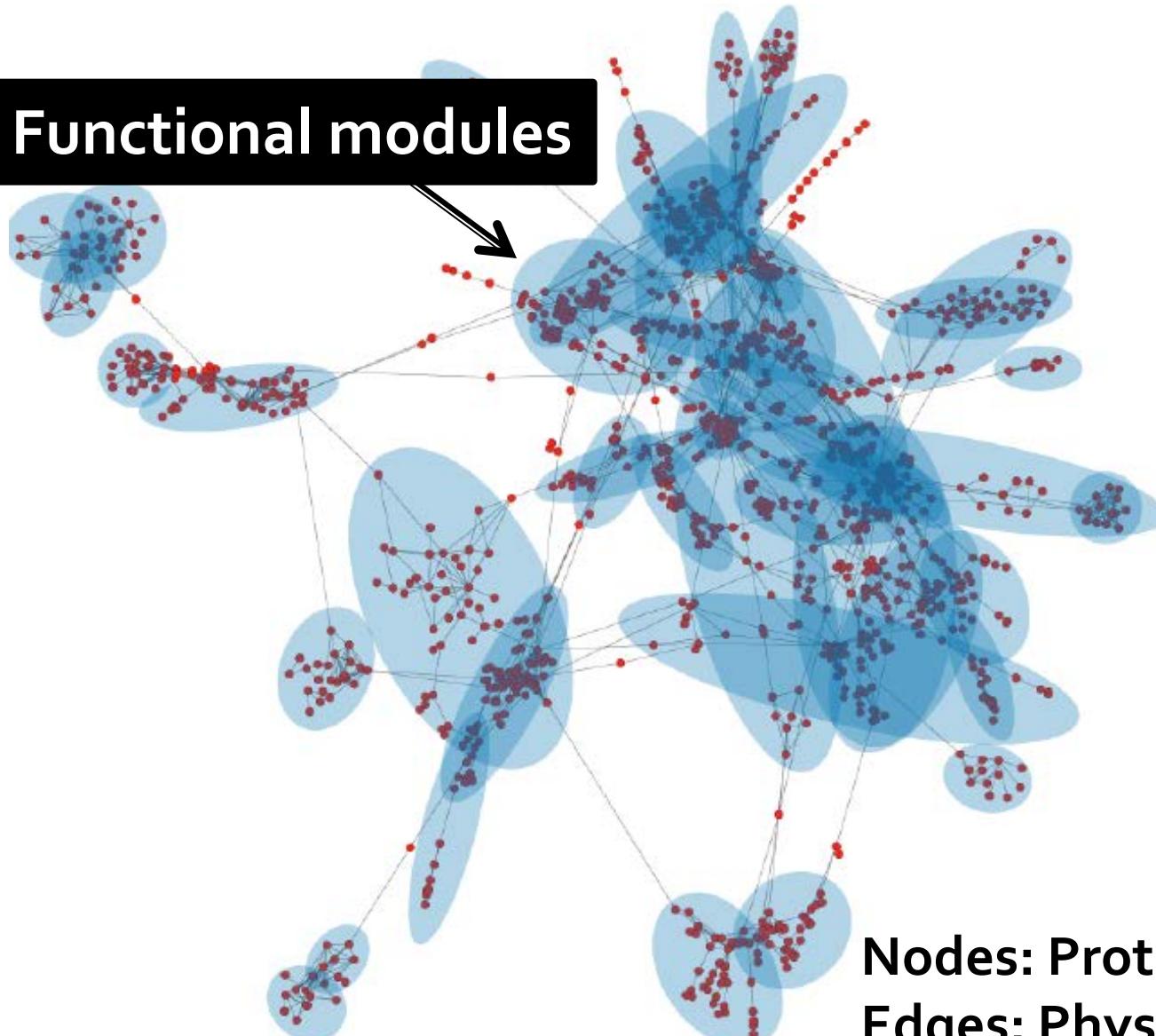


Protein-Protein Interactions

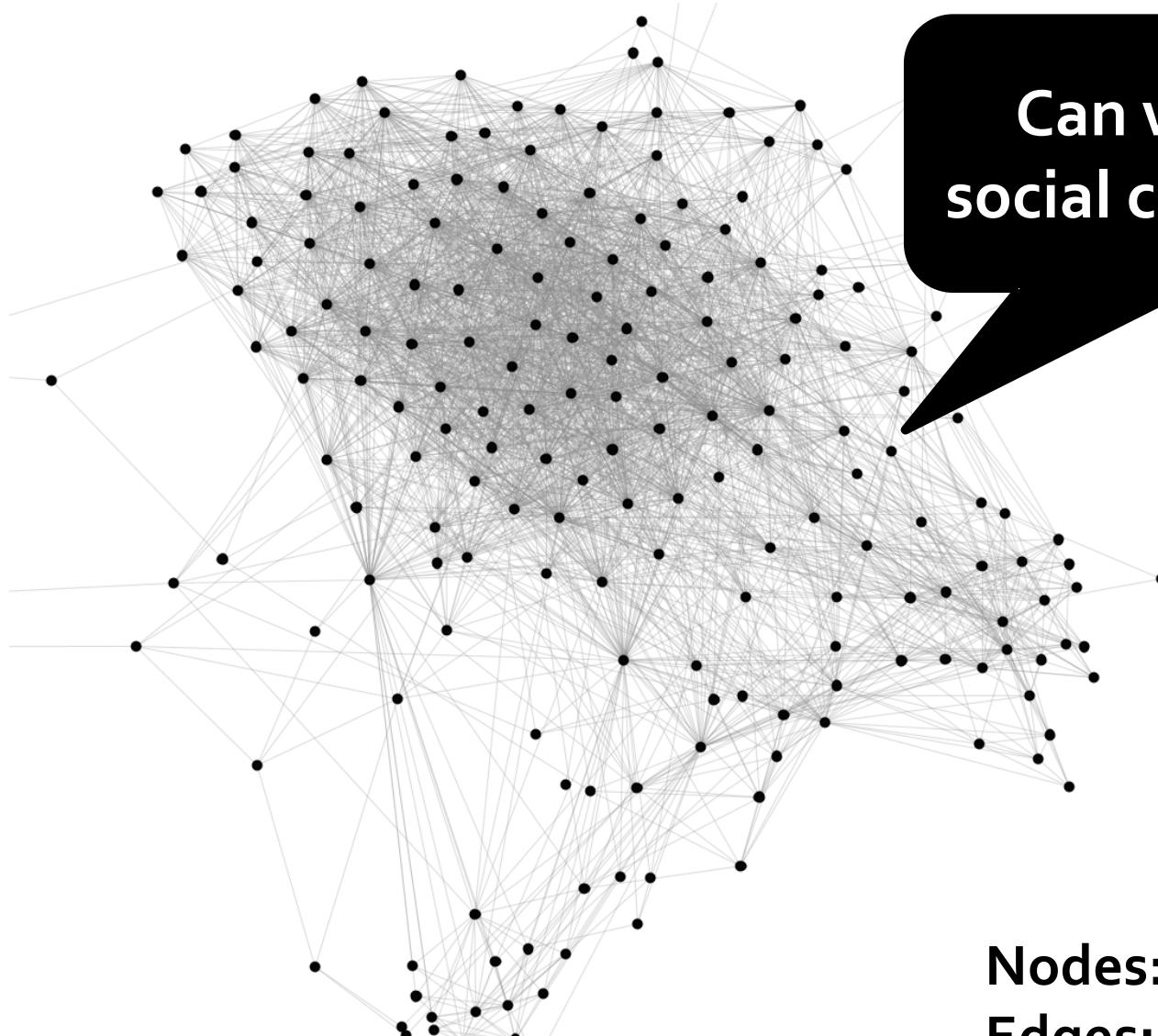


Protein-Protein Interactions

Functional modules



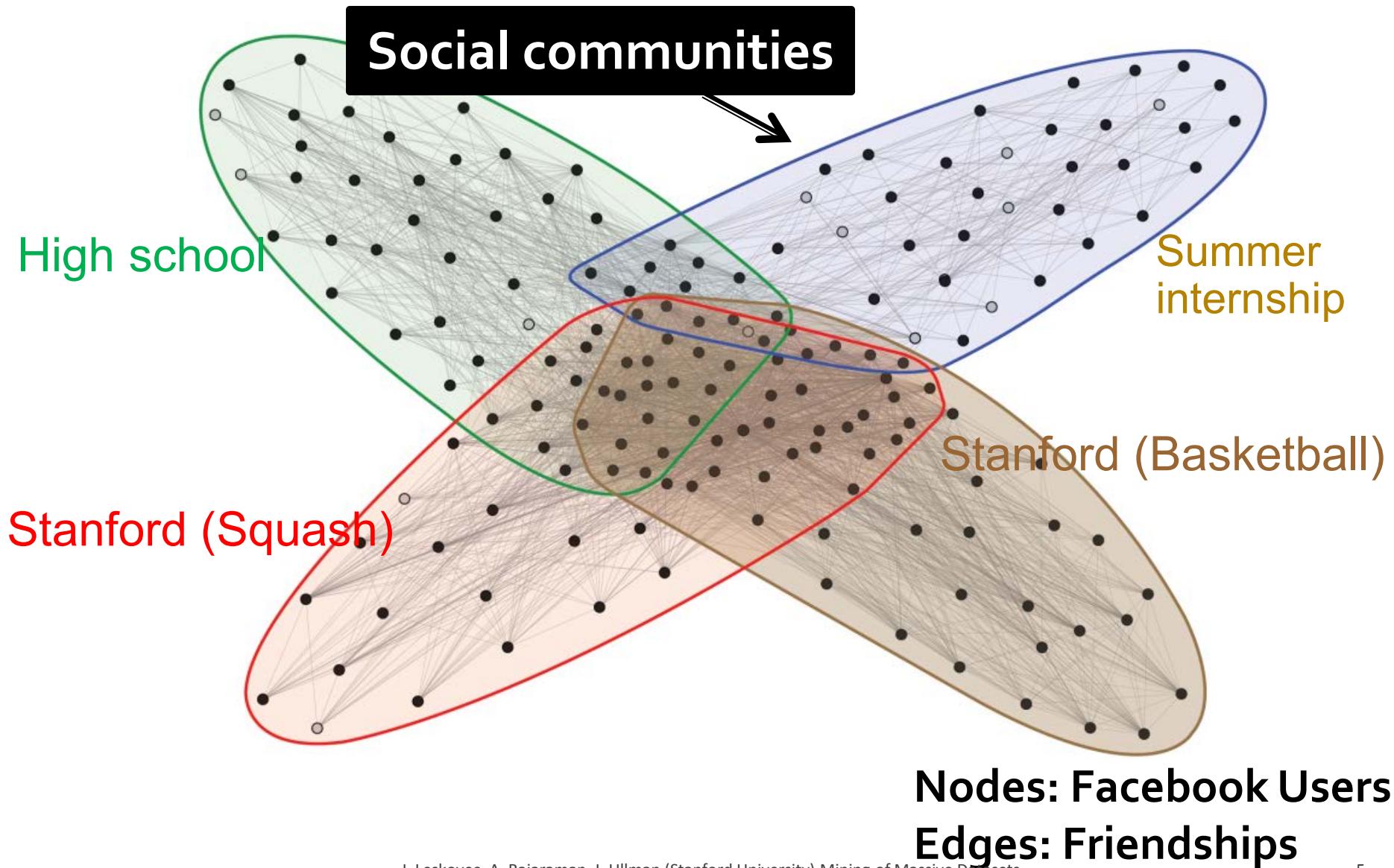
Facebook Network



Can we identify
social communities?

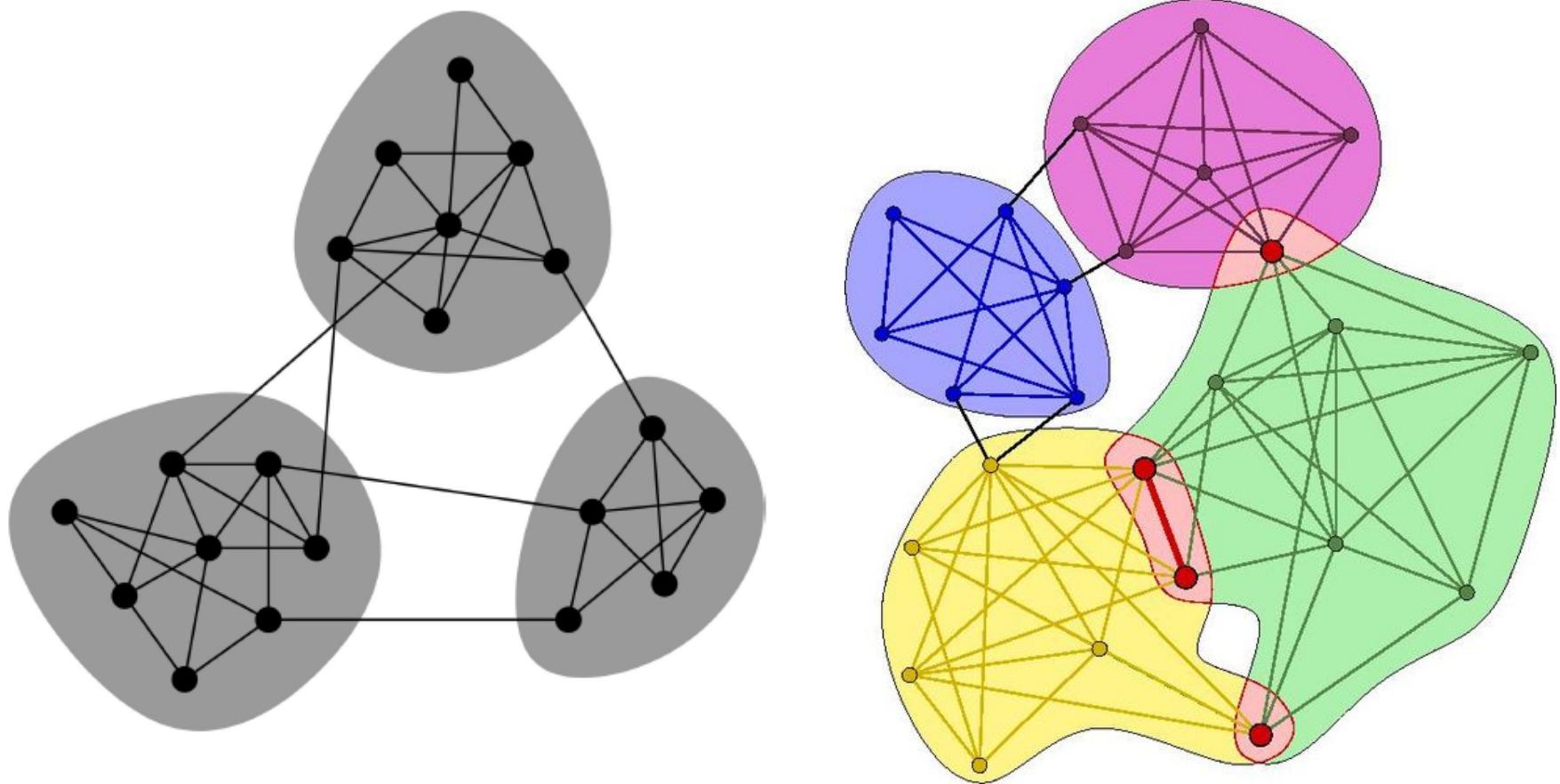
Nodes: Facebook Users
Edges: Friendships

Facebook Network



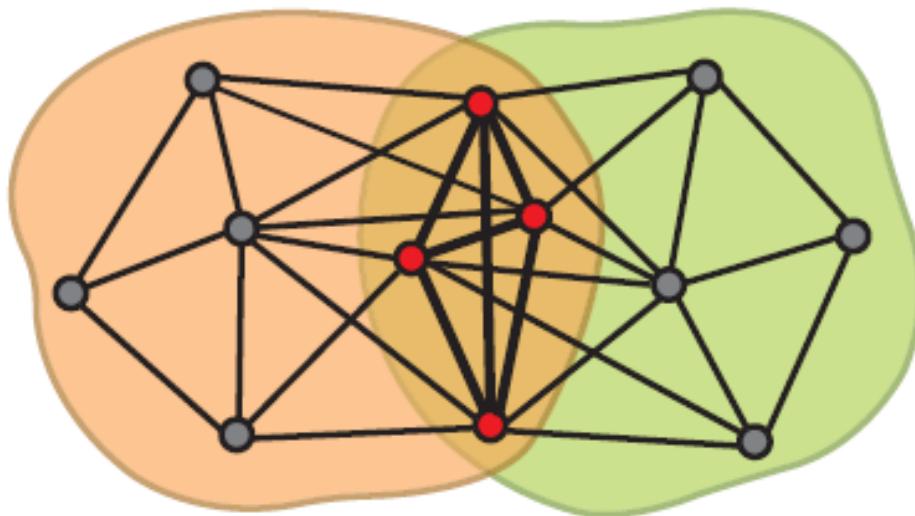
Overlapping Communities

- Non-overlapping vs. overlapping communities



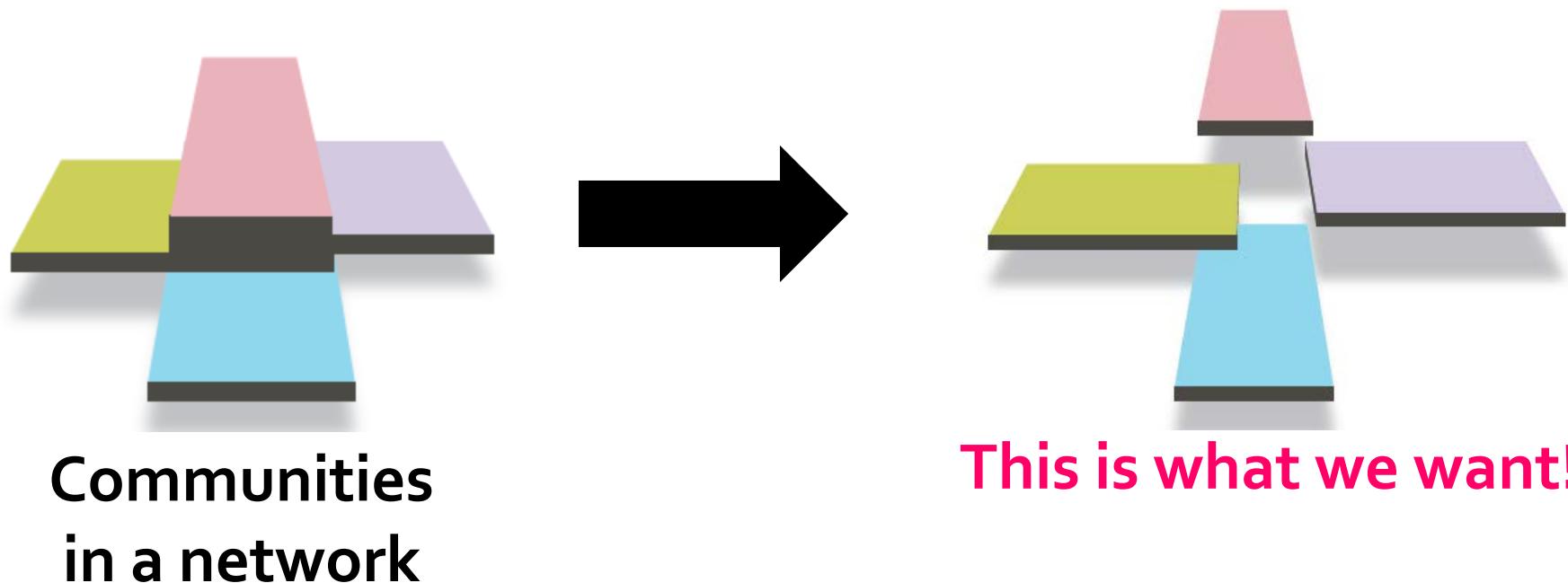
Communities as Tiles!

- What is the structure of community overlaps:
Edge density in the overlaps is higher!



Communities as “tiles”

Recap so far...



Counting 1's

Approximating Counts
Exponentially Growing Blocks
DGIM Algorithm

Counting Bits

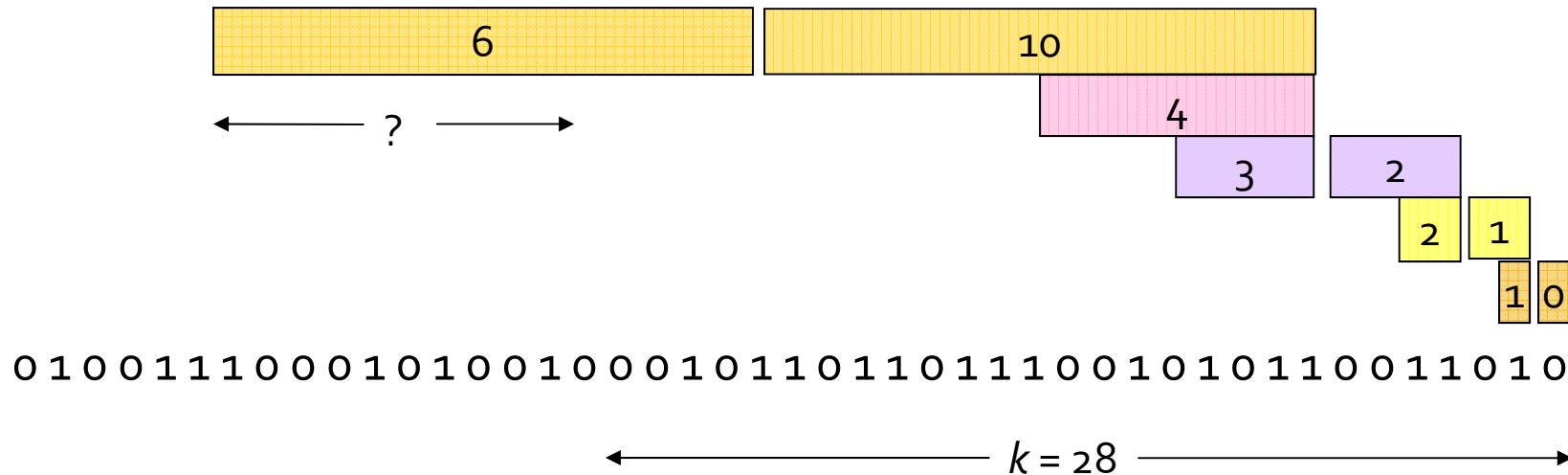
- **Problem:** given a stream of 0's and 1's, be prepared to answer queries of the form “how many 1's in the last k bits?” where $k \leq N$.
- **Obvious solution:** store the most recent N bits.
- But answering the query will take $O(k)$ time.
 - Very possibly too much time.
- And the space requirements can be too great.
 - Especially if there are many streams to be managed in main memory at once, or N is huge.

Something That Doesn't (Quite) Work

- Summarize exponentially increasing blocks of the stream, looking backward.
- Drop small blocks if they begin at the same point as a larger region or a larger region begins to their right.
 - Thus, never more than two blocks of any size.

Example

We can construct the count of the last k bits, except we're not sure how many of the last 6 are included.



What's Good?

- Stores only $O(\log^2 N)$ bits.
 - $O(\log N)$ counts of $\log_2 N$ bits each.
- Easy update as more bits enter.
- Error in count no greater than the number of 1's in the “unknown” area.

What's Not So Good?

- As long as the 1's are fairly evenly distributed, the error due to the unknown region is small – no more than 50%.
- But it could be that all the 1's are in the unknown area at the end.
- In that case, the error is unbounded.

Fixup

- Instead of summarizing fixed-length blocks, summarize blocks with specific numbers of 1's.
 - Let the block *sizes* (number of 1's) increase exponentially.
- When there are few 1's in the window, block sizes stay small, so errors are small.

DGIM Method

- Name refers to the inventors:
 - Datar, Gionis, Indyk, and Motwani.
- Store $O(\log^2 N)$ bits per stream.
- Gives approximate answer, never off by more than 50%.
 - Error factor can be reduced to any fraction > 0 , with more complicated algorithm and proportionally more stored bits.

Timestamps

- Each bit in the stream has a *timestamp*, starting 0, 1, ...
- Record timestamps modulo N (the window size), so we can represent any relevant timestamp in $O(\log_2 N)$ bits.

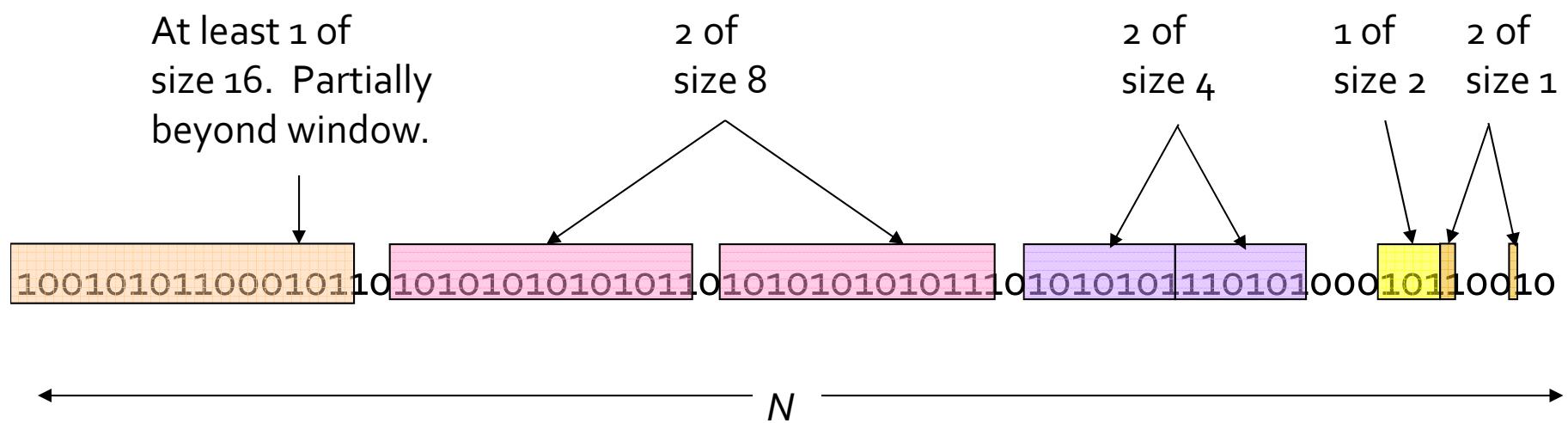
Buckets

- A *bucket* is a segment of the window; it is represented by a record consisting of:
 1. The timestamp of its end [$O(\log N)$ bits].
 2. The number of 1's between its beginning and end [$O(\log \log N)$ bits].
 - Number of 1's = *size* of the bucket.
- **Constraint on buckets:** number of 1's must be a power of 2.
 - That explains the $\log \log N$ in (2).

Representing a Stream by Buckets

- Either one or two buckets with the same power-of-2 number of 1's.
- Buckets do not overlap.
- Buckets are sorted by size.
 - Earlier buckets are not smaller than later buckets.
- Buckets disappear when their end-time is $> N$ time units in the past.

Example: Bucketized Stream



Updating Buckets

- When a new bit comes in, drop the last (oldest) bucket if its end-time is prior to N time units before the current time.
- If the current bit is 0, no other changes are needed.

Updating Buckets: Input = 1

- If the current bit is 1:
 1. Create a new bucket of size 1, for just this bit.
 - End timestamp = current time.
 2. If there are now three buckets of size 1, combine the oldest two into a bucket of size 2.
 3. If there are now three buckets of size 2, combine the oldest two into a bucket of size 4.
 4. And so on ...

Example

The image displays a vertical sequence of six binary strings, each consisting of 16 bits. The strings are color-coded into four segments: orange (bits 0-3), pink (bits 4-7), purple (bits 8-11), and yellow (bits 12-15). The strings are as follows:

- String 1: 10010101100010110 0101010101010110 010101010101110 01010101110101000 10110010
- String 2: 0010101100010110 0101010101010110 010101010101110 01010101110101000 101100101
- String 3: 0010101100010110 0101010101010110 010101010101110 01010101110101000 101100101101
- String 4: 0101100010110 0101010101010110 010101010101110 01010101110101000 101100101101
- String 5: 0101100010110 010101010101010110 010101010101110 01010101110101000 101100101101
- String 6: 0101100010110 0101010101010110 010101010101110 01010101110101000 101100101101

The strings appear to be part of a search or comparison operation, possibly a bit-vector search or a comparison of two sets of binary data. The color-coding might represent different fields or bytes within the binary strings.

Querying

- To estimate the number of 1's in the most recent $k \leq N$ bits:
 1. Restrict your attention to only those buckets whose end time stamp is at most k bits in the past.
 2. Sum the sizes of all these buckets but the oldest.
 3. Add half the size of the oldest bucket.
- Remember: we don't know how many 1's of the last bucket are still within the window.

Error Bound

- Suppose the oldest bucket within range has size 2^i .
- Then by assuming 2^{i-1} of its 1's are still within the window, we make an error of at most 2^{i-1} .
- Since there is at least one bucket of each of the sizes less than 2^i , and at least 1 from the oldest bucket, the true sum is no less than 2^i .
- Thus, error at most 50%.

Counting Distinct Elements

Applications

Flajolet-Martin Approximation

Technique

Generalization to Moments

Counting Distinct Elements

- **Problem:** a data stream consists of elements chosen from a set of size n . Maintain a count of the number of distinct elements seen so far.
- **Obvious approach:** maintain the set of elements seen.

Applications

- How many different words are found among the Web pages being crawled at a site?
 - Unusually low or high numbers could indicate artificial pages (spam?).
- How many unique users visited Facebook this month?
- How many different pages link to each of the pages we have crawled.

Estimating Counts

- **Real Problem:** what if we do not have space to store the complete set?
- Estimate the count in an unbiased way.
- Accept that the count may be in error, but limit the probability that the error is large.

Flajolet-Martin Approach

- Pick a hash function h that maps each of the n elements to at least $\log_2 n$ bits.
- For each stream element a , let $r(a)$ be the number of trailing 0's in $h(a)$.
- Record $R = \text{the maximum } r(a) \text{ seen.}$
- Estimate = 2^R .

Why It Works

- The probability that a given $h(a)$ ends in at least i 0's is 2^{-i} .
- If there are m different elements, the probability that $R \geq i$ is $1 - (1 - 2^{-i})^m$.

$$1 - (1 - 2^{-i})^m$$

The term $(1 - 2^{-i})^m$ is enclosed in a rectangular box. Two arrows point upwards from text descriptions to this box: one arrow originates from the text "Prob. all $h(a)$'s end in fewer than i 0's." and the other from the text "Prob. a given $h(a)$ ends in fewer than i 0's."

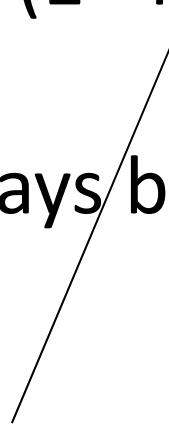
Prob. all $h(a)$'s
end in fewer than
 i 0's.

Prob. a given $h(a)$
ends in fewer than
 i 0's.

Why It Works – (2)

- Since 2^{-i} is small, $1 - (1-2^{-i})^m \approx 1 - e^{-m2^{-i}}$.
- If $2^i \gg m$, $1 - e^{-m2^{-i}} \approx 1 - (1 - m2^{-i}) \approx m/2^i \approx 0$.
- If $2^i \ll m$, $1 - e^{-m2^{-i}} \approx 1$.
- Thus, 2^R will almost always be around m .

First 2 terms of the
Taylor expansion of e^x



Why It Doesn't Work

- $E(2^R)$ is, in principle, infinite.
 - Probability halves when $R \rightarrow R+1$, but value doubles.
- Workaround involves using many hash functions and getting many samples.
- How are samples combined?
 - Average? What if one very large value?
 - Median? All values are a power of 2.

Solution

- Partition your samples into small groups.
 - Log n, where n = size of universal set, suffices.
- Take the average of groups.
- Then take the median of the averages.

Generalization: Moments

- Suppose a stream has elements chosen from a set of n values.
- Let m_i be the number of times value i occurs.
- The k^{th} *moment* is the sum of $(m_i)^k$ over all i .

Special Cases

- 0th moment = number of different elements in the stream.
 - The problem just considered.
- 1st moment = count of the numbers of elements = length of the stream.
 - Easy to compute.
- 2nd moment = *surprise number* = a measure of how uneven the distribution is.

Example: Surprise Number

- Stream of length 100; 11 values appear.
- **Unsurprising:** 10, 9, 9, 9, 9, 9, 9, 9, 9, 9. Surprise # = 910.
- **Surprising:** 90, 1, 1, 1, 1, 1, 1, 1 ,1, 1, 1. Surprise # = 8,110.

AMS Method

- Works for all moments; gives an unbiased estimate.
- We'll just concentrate on 2nd moment.
- Based on calculation of many random variables X .
 - Each requires a count in main memory, so number is limited.

One Random Variable

- Assume stream has length n .
- Pick a random time to start, so that any time is equally likely.
- Let the chosen time have element a in the stream.
- $X = n * ((\text{twice the number of } a\text{'s in the stream starting at the chosen time}) - 1)$.
 - Note: store n once, count of a 's for each X .

Expected Value of X

- 2nd moment is $\sum_a (m_a)^2$.
- $E(X) = (1/n)(\sum_{\text{all times } t} n * (\text{twice the number of times the stream element at time } t \text{ appears from that time on}) - 1)$.
- $= \sum_a (1/n)(n)(1+3+5+\dots+2m_a-1)$.
- $= \sum_a (m_a)^2$.
 - Group times by the value seen
 - Time when the last a is seen
 - Time when penultimate a is seen
 - Time when the first a is seen

Problem: Streams Never End

- We assumed there was a number n , the number of positions in the stream.
- But real streams go on forever, so n changes; it is the number of inputs seen so far.

Fixups

1. The variables X have n as a factor – keep n separately; just hold the count in X .
2. Suppose we can only store k counts. We must throw some X 's out as time goes on.
 - Objective: each starting time t is selected with probability k/n .

Solution to (2)

- Choose the first k times for k variables.
- When the n^{th} element arrives ($n > k$), choose it with probability k/n .
- If you choose it, throw one of the previously stored variables out, with equal probability.
- Probability of each of the first $n-1$ positions being chosen:

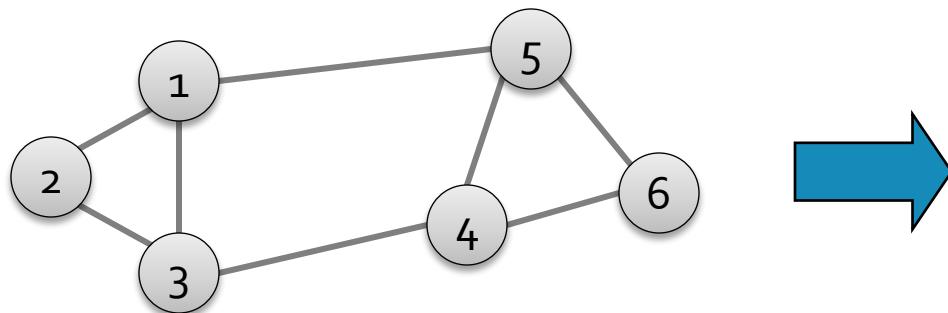
$$(n-k)/n * k/(n-1) + k/n * k/(n-1) * (k-1)/k = k/n$$


n-th position
not chosen Previously
chosen n-th position
chosen Previously
chosen Survives

Matrix Representations

■ Adjacency matrix (A):

- $n \times n$ matrix
- $A = [a_{ij}]$, $a_{ij} = 1$ if edge between node i and j



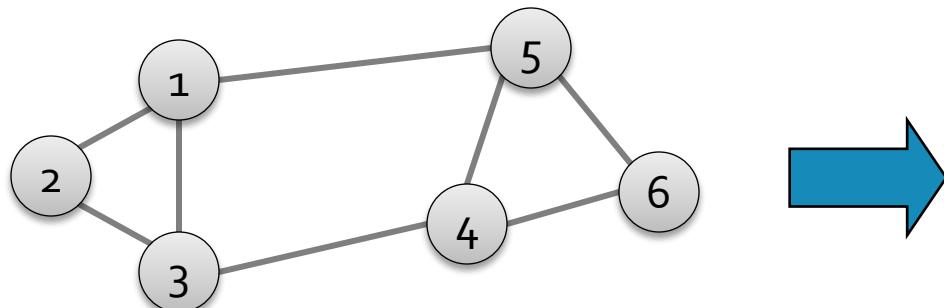
	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

■ Important properties:

- Symmetric matrix
- Eigenvectors are real and orthogonal

Matrix Representations

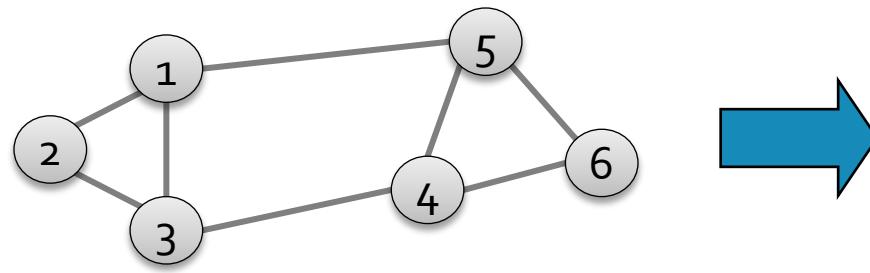
- **Degree matrix (D):**
 - $n \times n$ diagonal matrix
 - $D = [d_{ii}]$, d_{ii} = degree of node i



	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	3	0
6	0	0	0	0	0	2

Matrix Representations

- Laplacian matrix (L):
 - $n \times n$ symmetric matrix



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

- What is trivial eigenpair?

$$L = D - A$$

- $x = (1, \dots, 1)$ then $L \cdot x = 0$ and so $\lambda = \lambda_1 = 0$
- Important properties:
 - Eigenvalues are non-negative real numbers
 - Eigenvectors are real and orthogonal

Analysis of Large Graphs: Detecting clusters

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



New Topic: Graph Data!

High dim.
data

Locality
sensitive
hashing

Clustering

Dimensional
ity
reduction

Graph
data

PageRank,
SimRank

Community
Detection

Spam
Detection

Infinite
data

Filtering
data
streams

Web
advertising

Queries on
streams

Machine
learning

SVM

Decision
Trees

Perceptron,
kNN

Apps

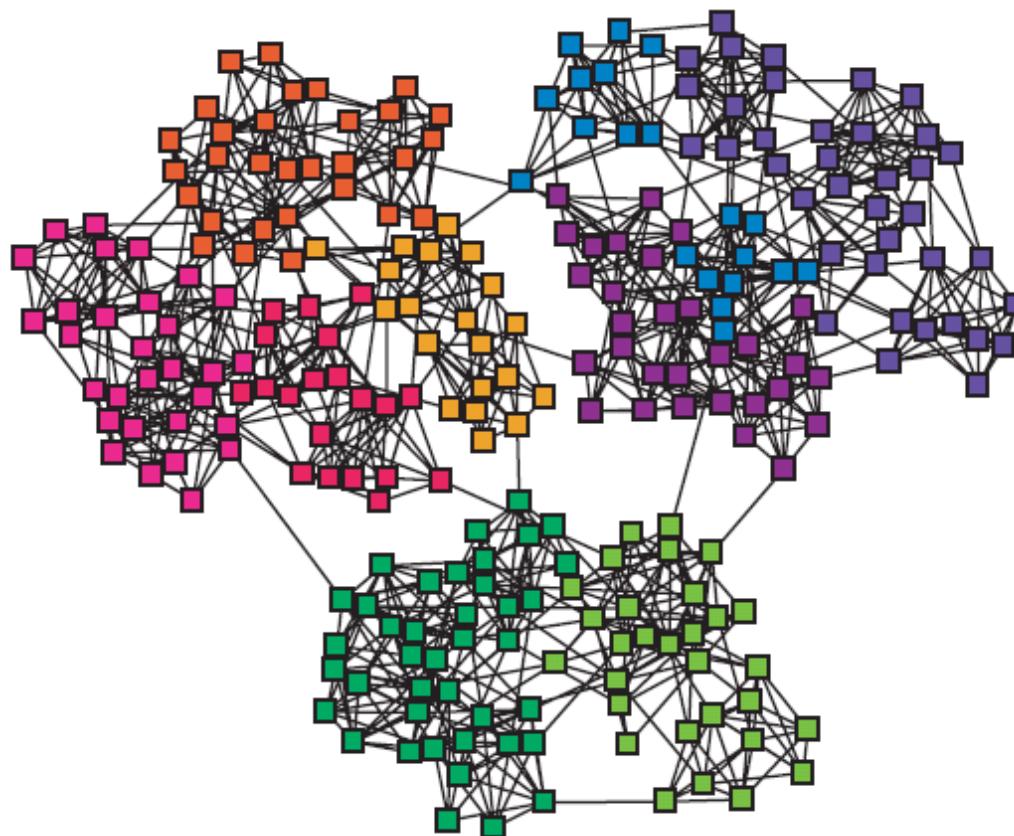
Recommen
der systems

Association
Rules

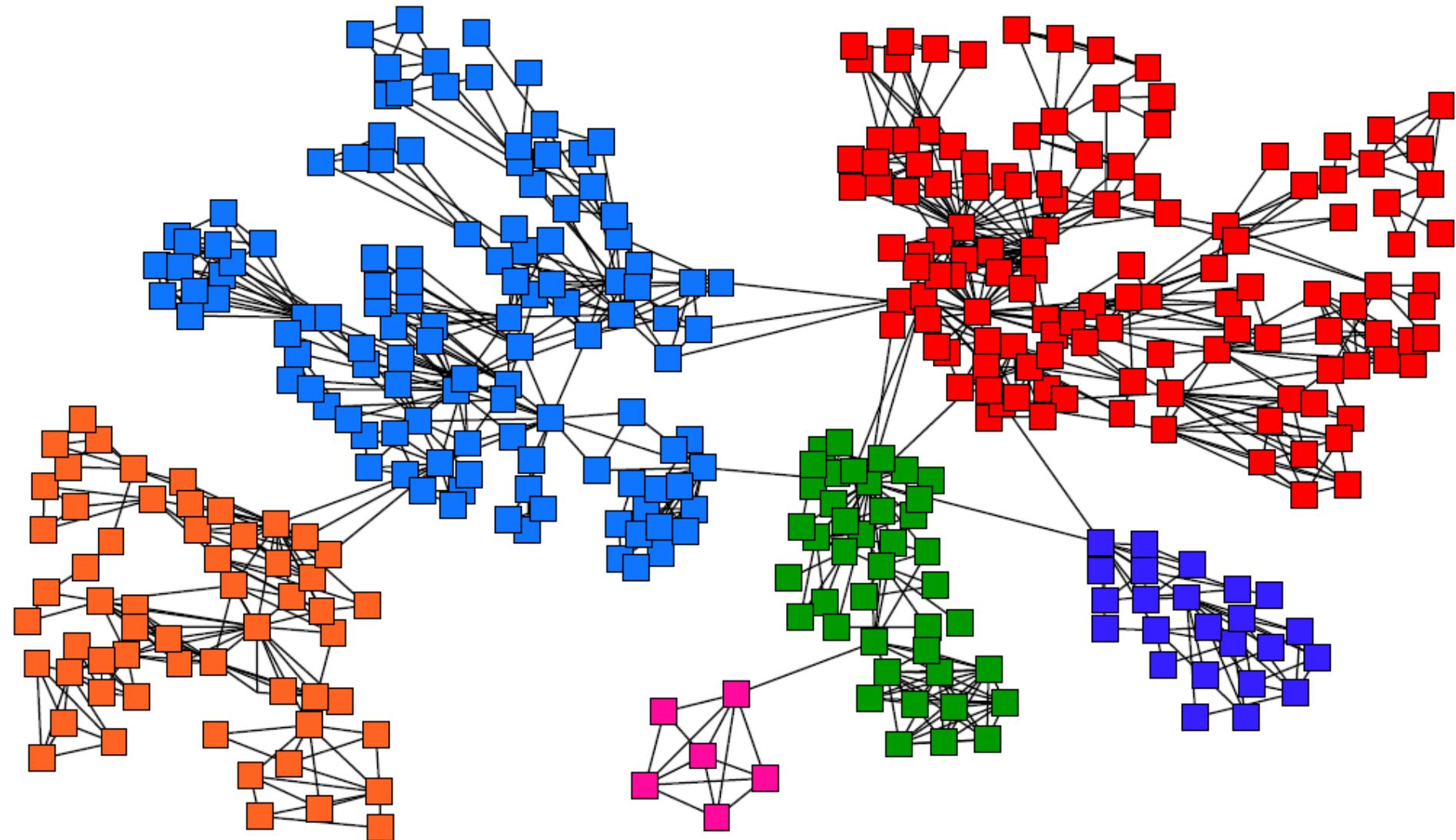
Duplicate
document
detection

Networks & Communities

- We often think of networks being organized into **modules, cluster, communities**:

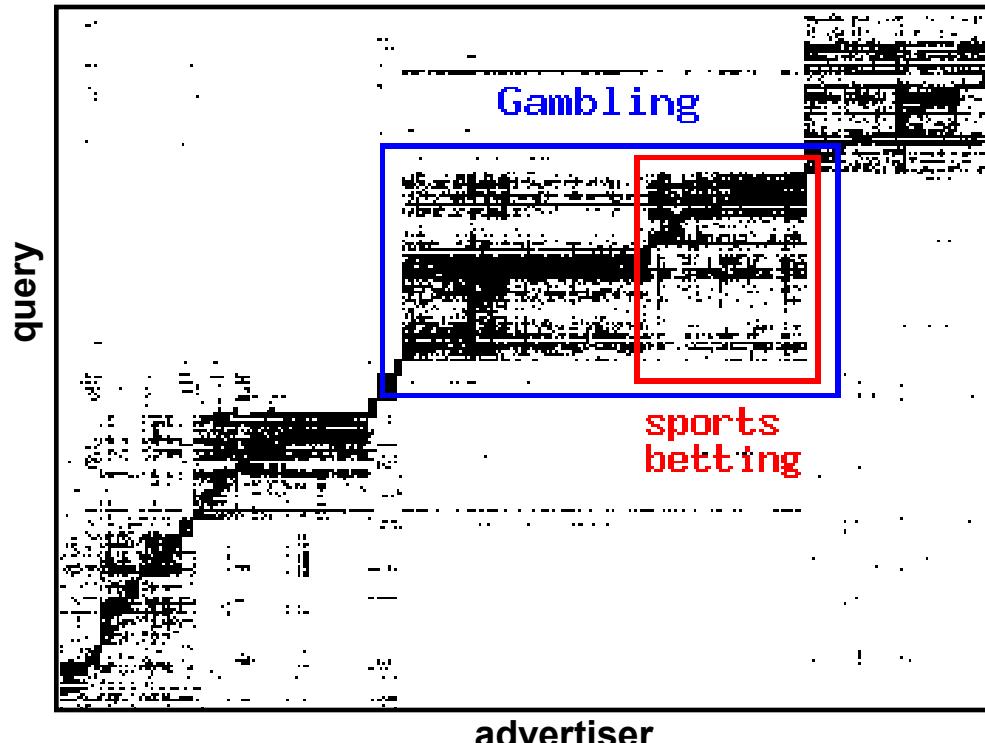


Goal: Find Densely Linked Clusters



Micro-Markets in Sponsored Search

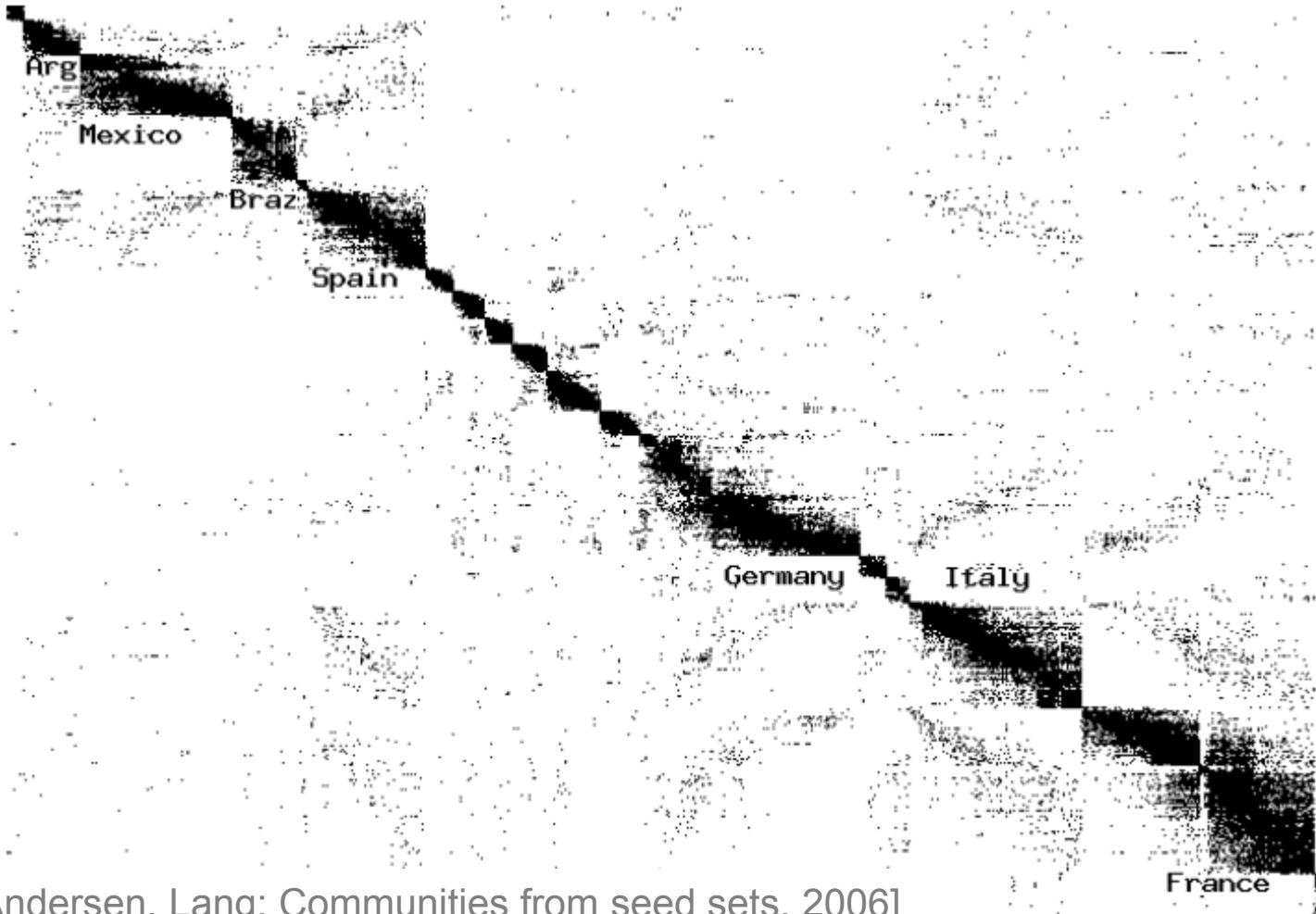
- Find micro-markets by partitioning the query-to-advertiser graph:



[Andersen, Lang: Communities from seed sets, 2006]

Movies and Actors

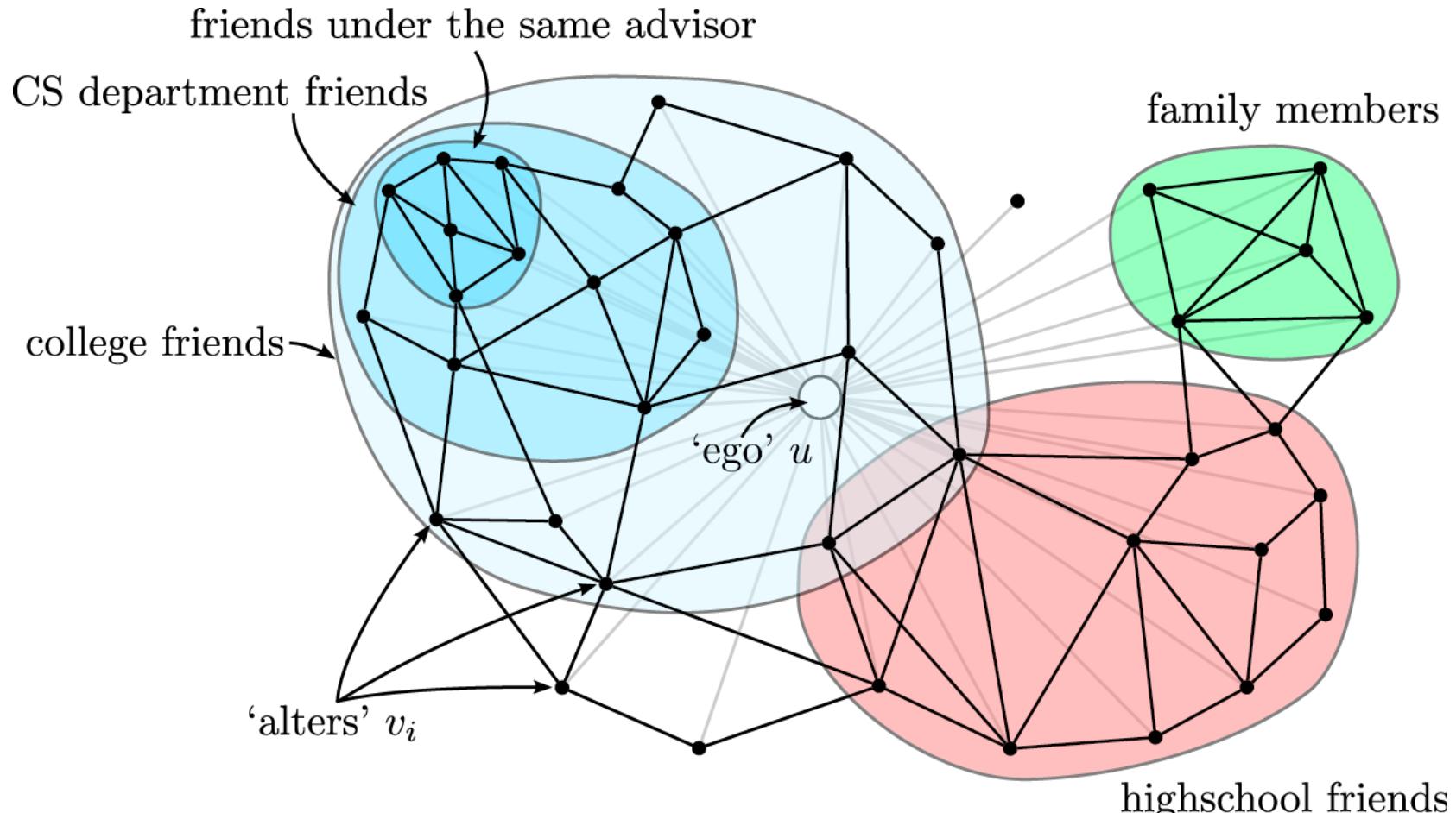
■ Clusters in Movies-to-Actors graph:



[Andersen, Lang: Communities from seed sets, 2006]

Twitter & Facebook

■ Discovering social circles, circles of trust:



Example: d-regular graph

- Suppose all nodes in G have degree d and G is connected
- **What are some eigenvalues/vectors of G ?**

$A \cdot x = \lambda \cdot x$ What is λ ? What x ?

- Let's try: $x = (1, 1, \dots, 1)$
- Then: $A \cdot x = (d, d, \dots, d) = \lambda \cdot x$. So: $\lambda = d$
- We found eigenpair of G : $x = (1, 1, \dots, 1)$, $\lambda = d$

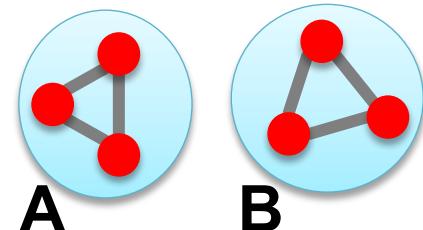
Remember the meaning of $y = A \cdot x$:

$$y_j = \sum_{i=1}^n A_{ij} x_i = \sum_{(j,i) \in E} x_i$$

Example: Graph on 2 components

- **What if G is not connected?**

- G has 2 components, each d -regular



- **What are some eigenvectors?**

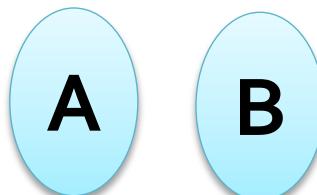
- $x = \text{Put all } 1\text{s on } A \text{ and } 0\text{s on } B \text{ or vice versa}$

- $x' = (\underbrace{1, \dots, 1}_{|A|}, \underbrace{0, \dots, 0}_{|B|})$ then $A \cdot x' = (d, \dots, d, 0, \dots, 0)$

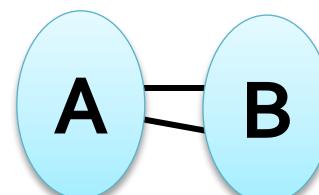
- $x'' = (0, \dots, 0, \underbrace{1, \dots, 1}_{|B|})$ then $A \cdot x'' = (0, \dots, 0, d, \dots, d)$

- And so in both cases the corresponding $\lambda = d$

- **A bit of intuition:**



$$\lambda_1 = \lambda_2$$



$$\lambda_1 \approx \lambda_2$$

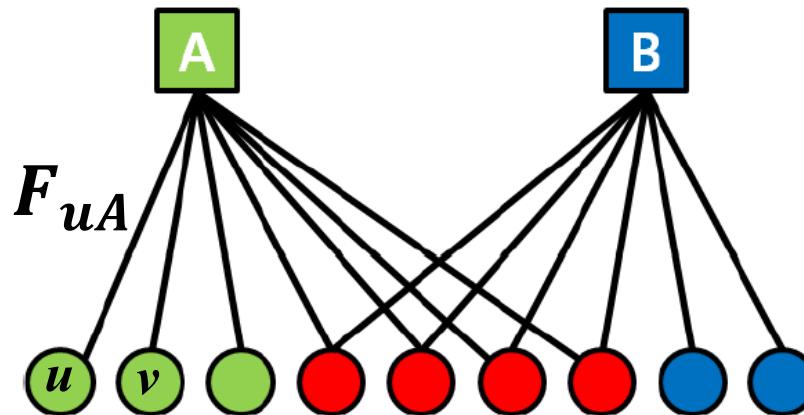
From AGM to BIGCLAM

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



From AGM to BigCLAM

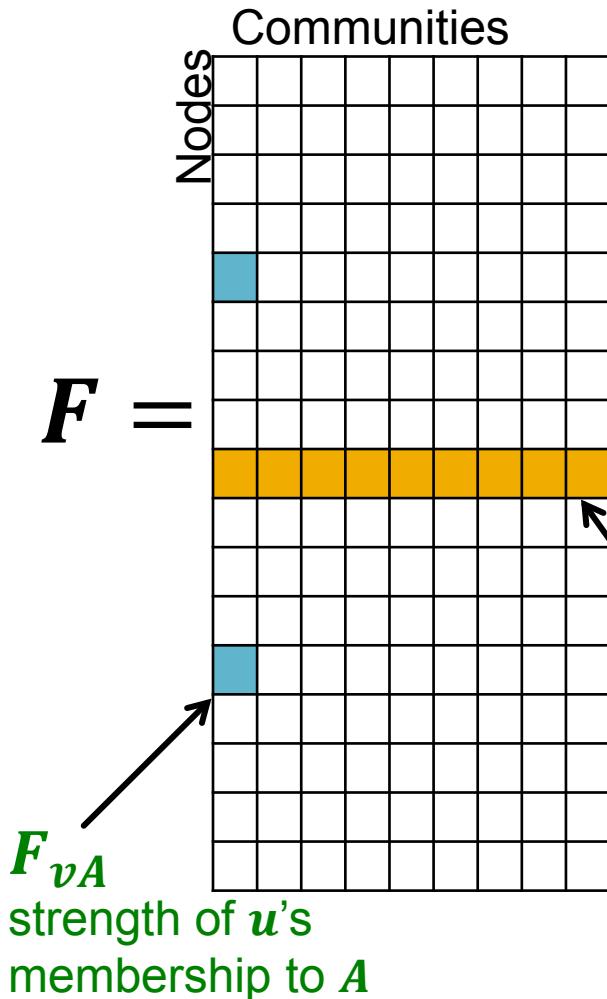
- Relaxation: Memberships have strengths



- F_{uA} : The membership strength of node u to community A ($F_{uA} = 0$: no membership)
- Each community A links nodes independently:
$$P_A(u, v) = 1 - \exp(-F_{uA} \cdot F_{vA})$$

Factor Matrix F

■ Community membership strength matrix F



- $P_A(u, v) = 1 - \exp(-F_{uA} \cdot F_{vA})$
 - Probability of connection is proportional to the product of strengths
 - Notice: If one node doesn't belong to the community ($F_{uC} = 0$) then $P(u, v) = 0$
- Prob. that **at least one** common community C links the nodes:
 - $P(u, v) = 1 - \prod_C (1 - P_C(u, v))$

From AGM to BigCLAM

- Community A links nodes u, v independently:

$$P_A(u, v) = 1 - \exp(-F_{uA} \cdot F_{vA})$$

- Then prob. at least one common C links them:

$$\begin{aligned} P(u, v) &= 1 - \prod_C (1 - P_C(u, v)) \\ &= 1 - \exp(-\sum_C F_{uC} \cdot F_{vC}) \\ &= 1 - \exp(-F_u \cdot F_v^T) \end{aligned}$$

- For example:

$$F_u: \begin{array}{|c|c|c|c|} \hline 0 & 1.2 & 0 & 0.2 \\ \hline \end{array}$$

$$F_v: \begin{array}{|c|c|c|c|} \hline 0.5 & 0 & 0 & 0.8 \\ \hline \end{array}$$

$$F_w: \begin{array}{|c|c|c|c|} \hline 0 & 1.8 & 1 & 0 \\ \hline \end{array}$$

Node community
membership strengths

Then: $F_u \cdot F_v^T = 0.16$

And: $P(u, v) = 1 - \exp(-0.16) = 0.14$

But: $P(u, w) = 0.88$

$P(v, w) = 0$

Mining Data Streams

The Stream Model
Sliding Windows
Counting 1's

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Data Management Vs. Stream Management

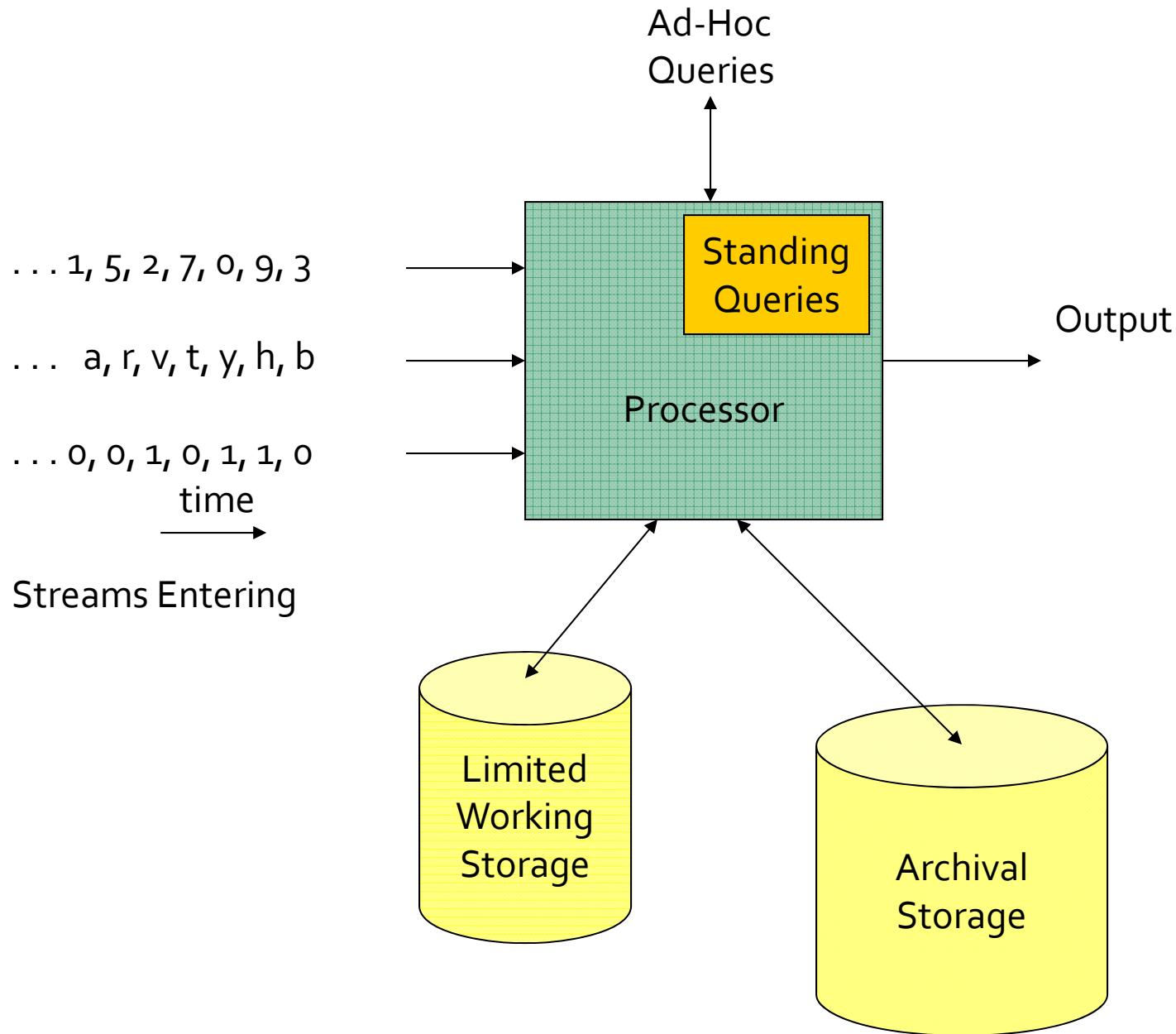
- In a DBMS, input is under the control of the programming staff.
 - SQL INSERT commands or bulk loaders.
- Stream Management is important when the input rate is controlled externally.
 - Example: Google search queries.

The Stream Model

- Input tuples enter at a rapid rate, at one or more input ports.
- The system cannot store the entire stream accessibly.
- How do you make critical calculations about the stream using a limited amount of (primary or secondary) memory?

Two Forms of Query

1. *Ad-hoc queries*: Normal queries asked one time about streams.
 - Example: What is the maximum value seen so far in stream S ?
2. *Standing queries*: Queries that are, in principle, asked about the stream at all times.
 - Example: Report each new maximum value ever seen in stream S .



Applications

- Mining query streams.
 - Google wants to know what queries are more frequent today than yesterday.
- Mining click streams.
 - Yahoo! wants to know which of its pages are getting an unusual number of hits in the past hour.
- IP packets can be monitored at a switch.
 - Gather information for optimal routing.
 - Detect denial-of-service attacks.

Sliding Windows

- A useful model of stream processing is that queries are about a *window* of length N – the N most recent elements received.
 - **Alternative:** elements received within a time interval T .
- **Interesting case:** N is so large it cannot be stored in main memory.
 - Or, there are so many streams that windows for all cannot be stored.

q w e r t y u i o p a s d f g h j k l z x c v b n m

q w e r t y u i o p a s d f g h j k l z x c v b n m

q w e r t y u i o p a s d f g h j k l z x c v b n m

q w e r t y u i o p a s d f g h j k l z x c v b n m

← Past Future →

Example: Averages

- Stream of integers.
- Window of size N .
- **Standing query:** what is the average of the integers in the window?
- For the first N inputs, sum and count to get the average.
- Afterward, when a new input i arrives, change the average by adding $(i - j)/N$, where j is the oldest integer in the window.

Sampling a Stream

What Doesn't Work
Sampling Based on Hash Values

When Sampling Doesn't Work

- Suppose Google would like to examine its stream of search queries for the past month to find out what fraction of them were unique – asked only once.
- But to save time, we are only going to sample $1/10^{\text{th}}$ of the stream.
- The fraction of unique queries in the sample will not be the fraction for the stream as a whole.
 - In fact, we can't even adjust the sample's fraction to give the correct answer.

Example: Unique Search Queries

- The length of the sample is 10% of the length of the whole stream.
- Suppose a query is unique.
 - It has a 10% chance of being in the sample.
- Suppose a query occurs exactly twice in the stream.
 - It has an 18% chance of appearing exactly once in the sample.
- And so on ... The fraction of unique queries in the stream is unpredictably large.

Sampling by Value

- Our mistake: we sampled based on the position in the stream, rather than the value of the stream element.
- Hash search queries to 10 buckets 0, 1,..., 9.
- Sample = all search queries that hash to bucket 0.
 - All or none of the instances of a query are selected.
 - Therefore the fraction of unique queries in the sample is the same as for the stream as a whole.

Controlling the Sample Size

- **Problem:** What if the total sample size is limited?
- **Solution:** Hash to a large number of buckets.
- Adjust the set of buckets accepted for the sample, so your sample size stays within bounds.

Example: Fixed Sample Size

- Suppose we start our search-query sample at 10%, but we want to limit the size.
- Hash to, say, 100 buckets, 0, 1,..., 99.
 - Take for the sample those elements hashing to buckets 0 through 9.
- If the sample gets too big, get rid of bucket 9.
- Still too big, get rid of 8, and so on.

Sampling Key-Value Pairs

- Our technique generalizes to any form of data that we can see as tuples (K, V) , where K is the “key” and V is a “value.”
- **Distinction:** We want our sample to be based on picking some set of keys only, not pairs.
 - In the search-query example, the data was “all key.”
- Hash keys to some number of buckets.
- Sample then consists of all key-value pairs with a key that goes into one of the selected buckets.

Example: Salary Ranges

- Data = tuples of the form (EmplD, Dept, Salary).
- **Query:** What is the average range of salaries within a department?
- Key = Dept.
- Value = (EmplD, Salary).
- Sample picks some departments, has salaries for all employees of that department, including its min and max salaries.

BigCLAM: How to find F

- **Task:** Given a network $G(V, E)$, estimate F
 - Find F that maximizes the likelihood:

$$\arg \max_F \prod_{(u,v) \in E} p(u, v) \prod_{(u,v) \notin E} (1 - p(u, v))$$

- where: $P(u, v) = 1 - \exp(-F_u \cdot F_v^T)$
- Many times we take the logarithm of the likelihood, and call it log-likelihood: $l(F) = \log P(G|F)$

- **Goal:** Find F that maximizes $l(F)$:

$$l(F) = \sum_{(u,v) \in E} \log(1 - \exp(-F_u F_v^T)) - \sum_{(u,v) \notin E} F_u F_v^T$$

BigCLAM: V1.0

$$l(F_u) = \sum_{v \in \mathcal{N}(u)} \log(1 - \exp(-F_u F_v^T)) - \sum_{v \notin \mathcal{N}(u)} F_u F_v^T$$

- Compute gradient of a single row F_u of F :

$$\nabla l(F_u) = \sum_{v \in \mathcal{N}(u)} F_v \frac{\exp(-F_u F_v^T)}{1 - \exp(-F_u F_v^T)} - \sum_{v \notin \mathcal{N}(u)} F_v$$

- Coordinate gradient ascent:

$\mathcal{N}(u)$.. Set out outgoing neighbors

- Iterate over the rows of F :

- Compute gradient $\nabla l(F_u)$ of row u (while keeping others fixed)
 - Update the row F_u : $F_u \leftarrow F_u + \eta \nabla l(F_u)$
 - Project F_u back to a non-negative vector: If $F_{uC} < 0$: $F_{uC} = 0$

- This is slow! Computing $\nabla l(F_u)$ takes linear time!

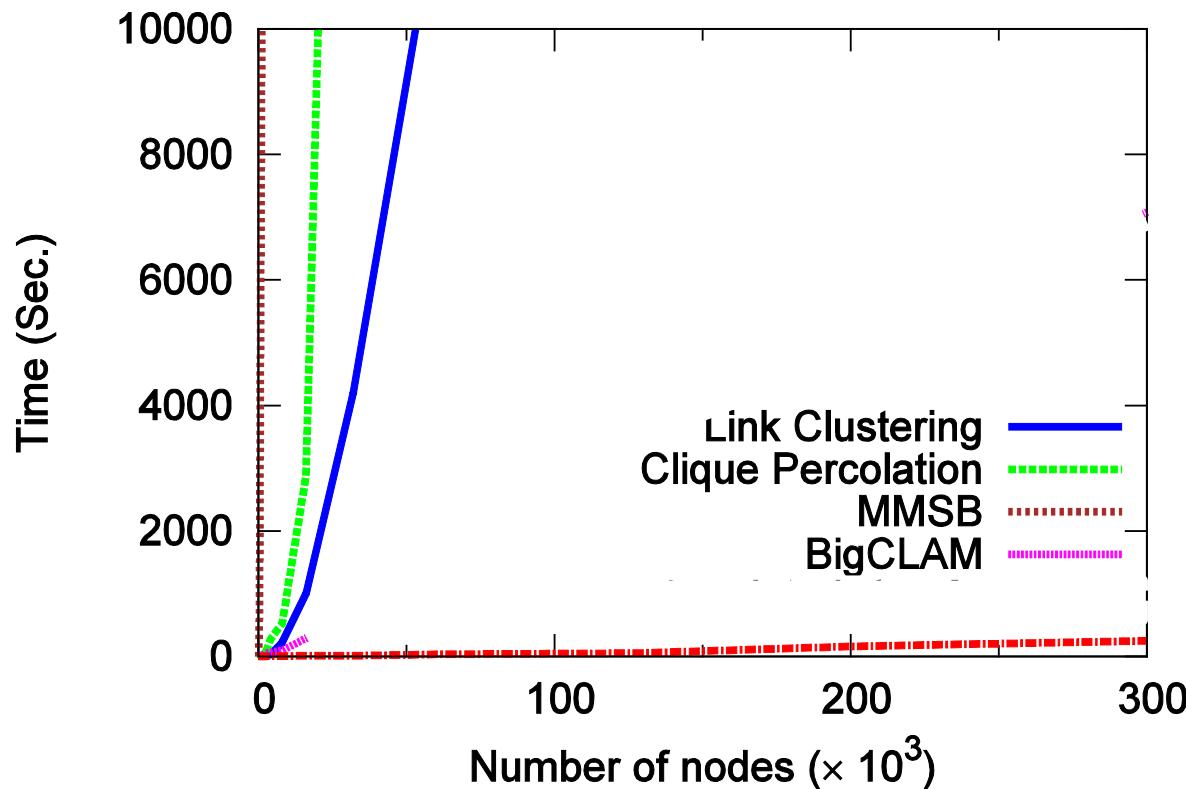
BigCLAM: V2.0

- However, we notice:

$$\sum_{v \notin \mathcal{N}(u)} F_v = \left(\sum_v F_v - F_u - \sum_{v \in \mathcal{N}(u)} F_v \right)$$

- We cache $\sum_v F_v$
- So, computing $\sum_{v \notin \mathcal{N}(u)} F_v$ now takes **linear time** in the degree $|\mathcal{N}(u)|$ of u
 - In networks degree of a node is much smaller to the total number of nodes in the network, so this is a significant speedup!

BigClam: Scalability



- **BigCLAM takes 5 minutes for 300k node nets**
 - Other methods take 10 days
- **Can process networks with 100M edges!**

More details at...

- [Overlapping Community Detection at Scale: A Nonnegative Matrix Factorization Approach](#) by J. Yang, J. Leskovec. *ACM International Conference on Web Search and Data Mining (WSDM)*, 2013.
- [Detecting Cohesive and 2-mode Communities in Directed and Undirected Networks](#) by J. Yang, J. McAuley, J. Leskovec. *ACM International Conference on Web Search and Data Mining (WSDM)*, 2014.
- [Community Detection in Networks with Node Attributes](#) by J. Yang, J. McAuley, J. Leskovec. *IEEE International Conference On Data Mining (ICDM)*, 2013.

Spectral Clustering Algorithms

- **Three basic stages:**

- **1) Pre-processing**

- Construct a matrix representation of the graph

- **2) Decomposition**

- Compute eigenvalues and eigenvectors of the matrix
 - Map each point to a lower-dimensional representation based on one or more eigenvectors

- **3) Grouping**

- Assign points to two or more clusters, based on the new representation

Spectral Partitioning Algorithm

■ 1) Pre-processing:

- Build Laplacian matrix L of the graph



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

■ 2) Decomposition:

- Find eigenvalues λ and eigenvectors x of the matrix L



$$\lambda =$$

0.0
1.0
3.0
3.0
4.0
5.0

$$X =$$

0.4	0.3	-0.5	-0.2	-0.4	-0.5
0.4	0.6	0.4	-0.4	0.4	0.0
0.4	0.3	0.1	0.6	-0.4	0.5
0.4	-0.3	0.1	0.6	0.4	-0.5
0.4	-0.3	-0.5	-0.2	0.4	0.5
0.4	-0.6	0.4	-0.4	-0.4	0.0

- Map vertices to corresponding components of λ_2

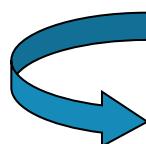
1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6



How do we now find the clusters?

Spectral Partitioning

- 3) **Grouping:**
 - Sort components of reduced 1-dimensional vector
 - Identify clusters by splitting the sorted vector in two
- **How to choose a splitting point?**
 - Naïve approaches:
 - Split at **0** or median value
 - More expensive approaches:
 - Attempt to minimize normalized cut in 1-dimension (sweep over ordering of nodes induced by the eigenvector)



1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6

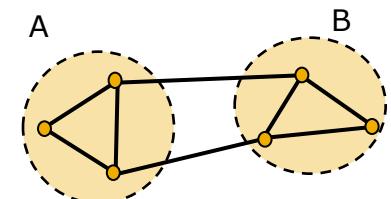


Split at 0:

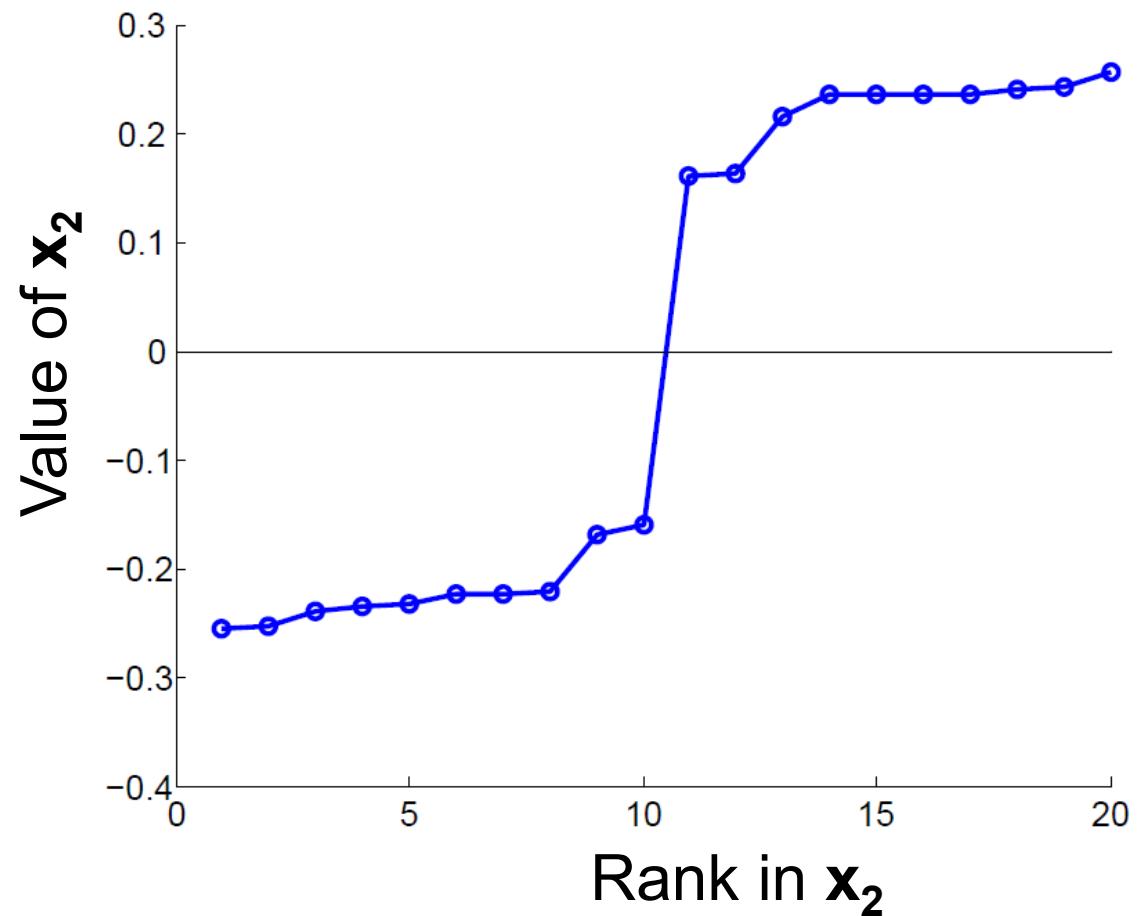
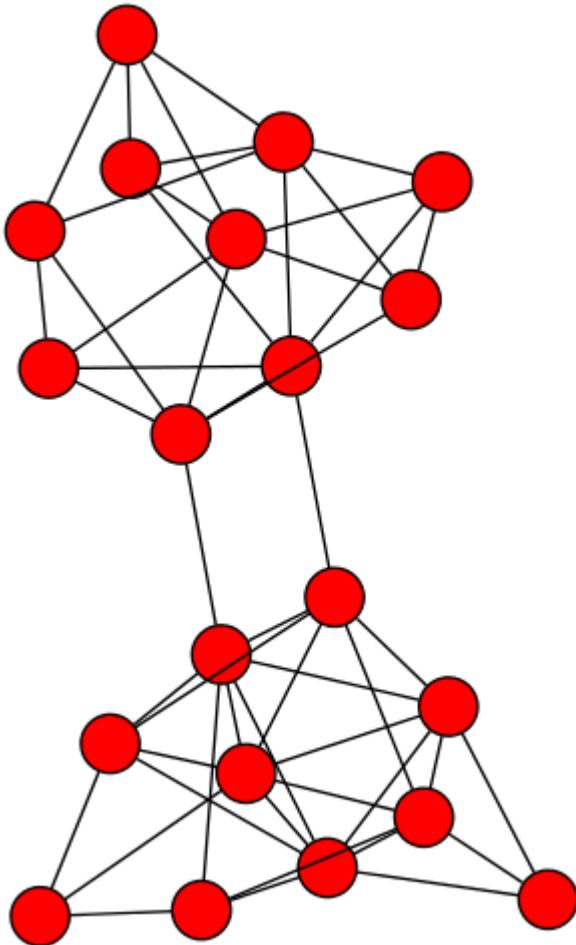
Cluster A: Positive points
Cluster B: Negative points

1	0.3
2	0.6
3	0.3

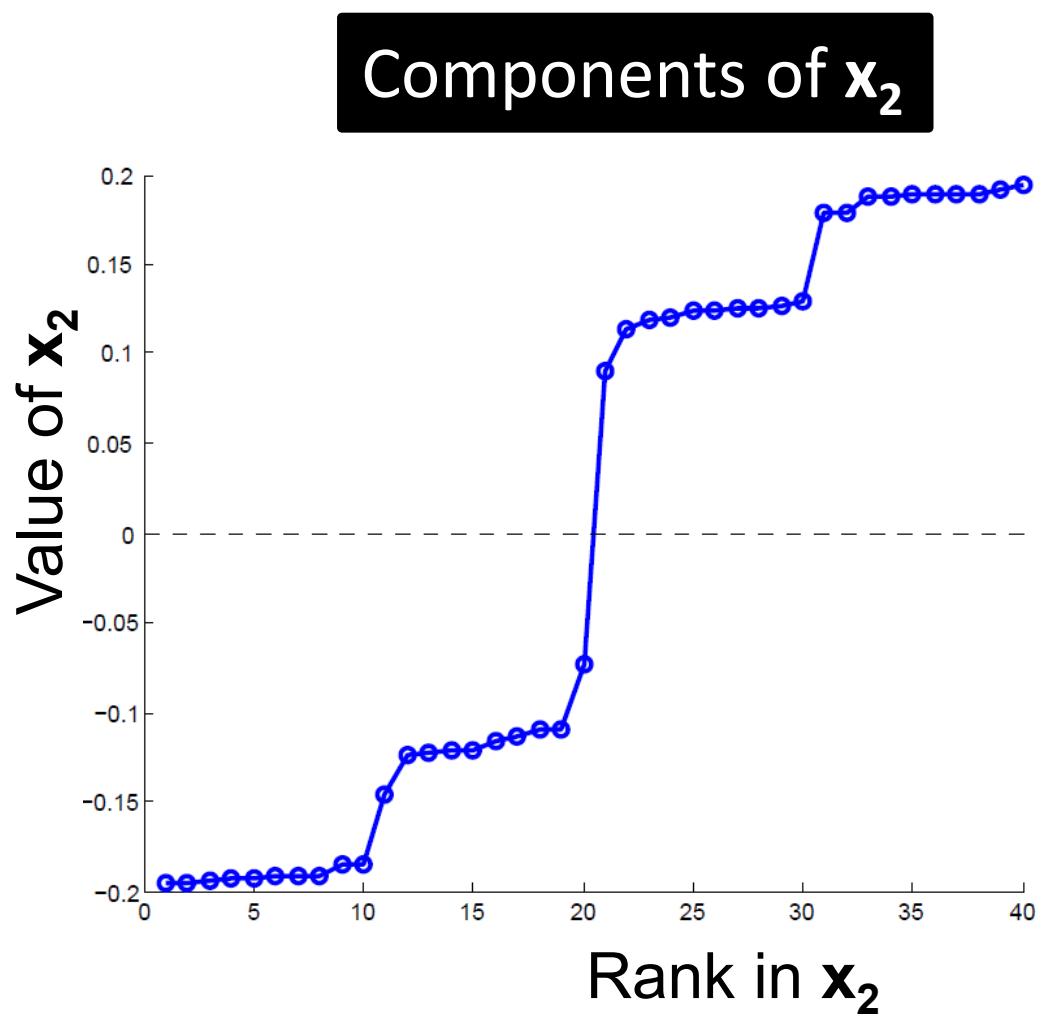
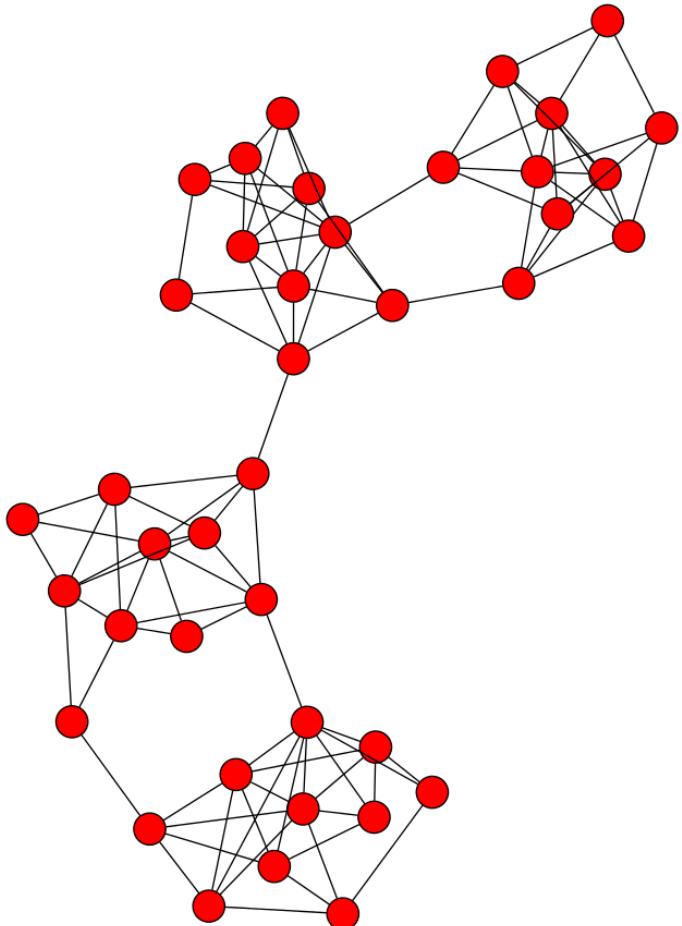
4	-0.3
5	-0.3
6	-0.6



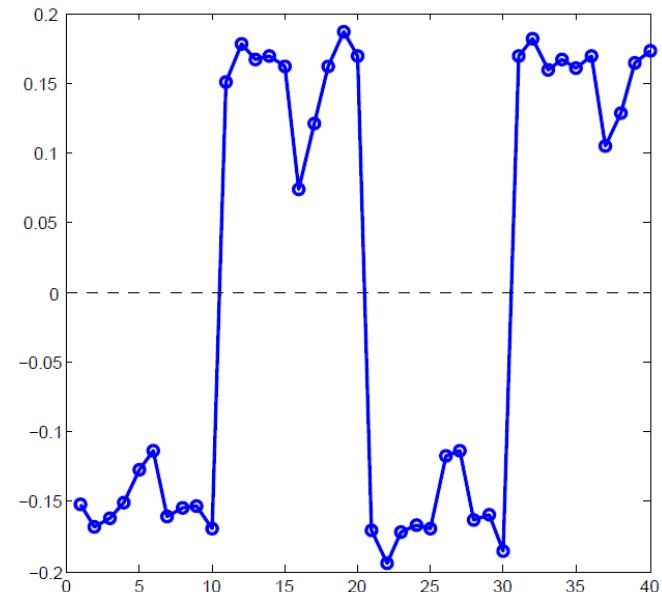
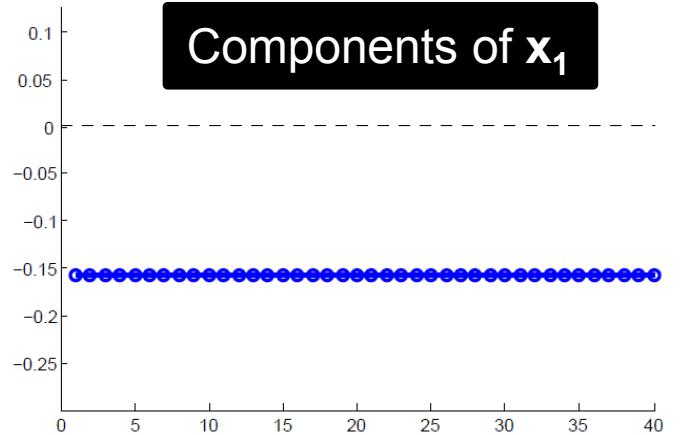
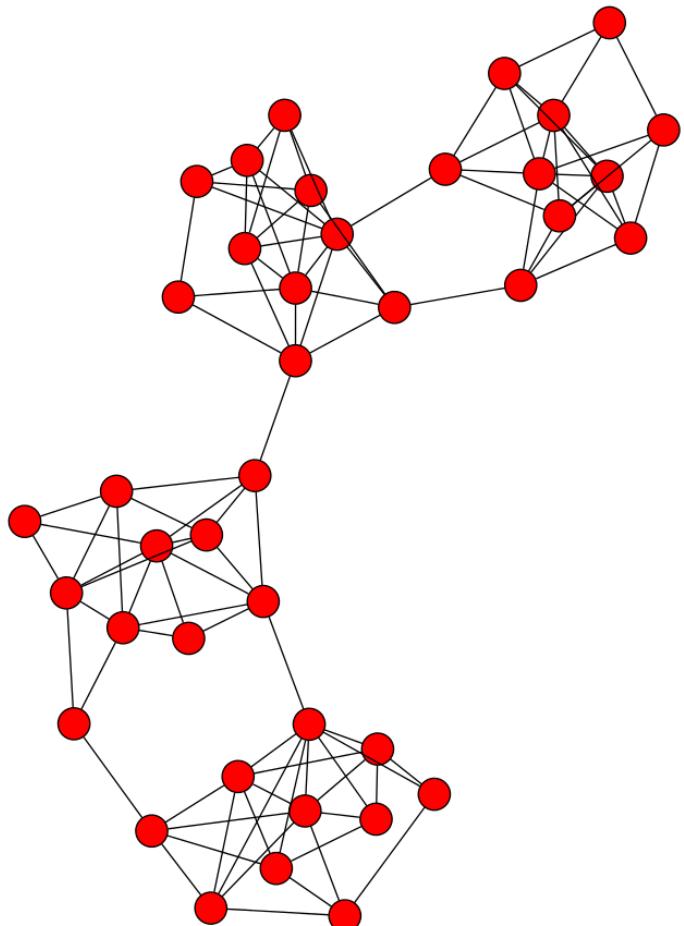
Example: Spectral Partitioning



Example: Spectral Partitioning



Example: Spectral Partitioning



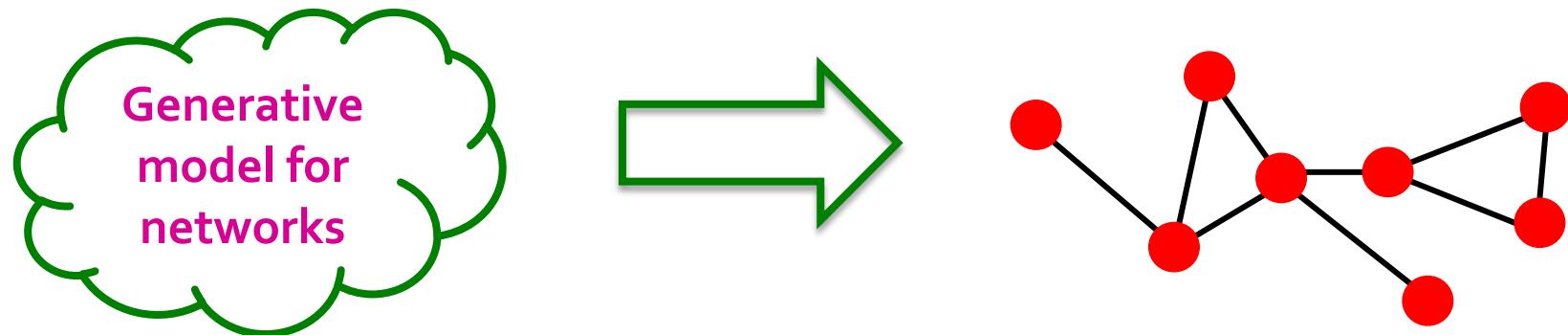
Components of \mathbf{x}_3

k-Way Spectral Clustering

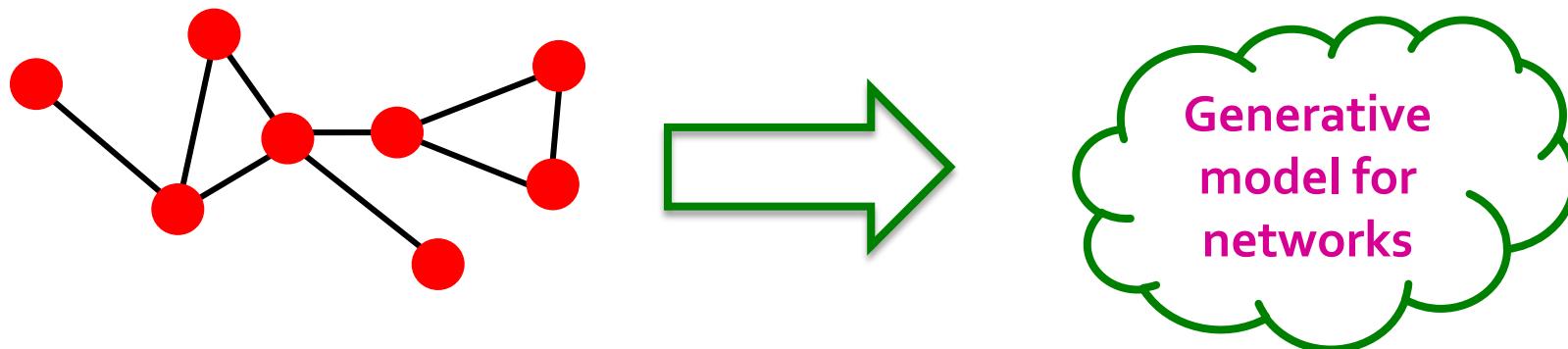
- **How do we partition a graph into k clusters?**
- **Two basic approaches:**
 - **Recursive bi-partitioning** [Hagen et al., '92]
 - Recursively apply bi-partitioning algorithm in a hierarchical divisive manner
 - Disadvantages: Inefficient, unstable
 - **Cluster multiple eigenvectors** [Shi-Malik, '00]
 - Build a reduced space from multiple eigenvectors
 - Commonly used in recent papers
 - A preferable approach...

Plan of attack

- 1) Given a model, we generate the network:

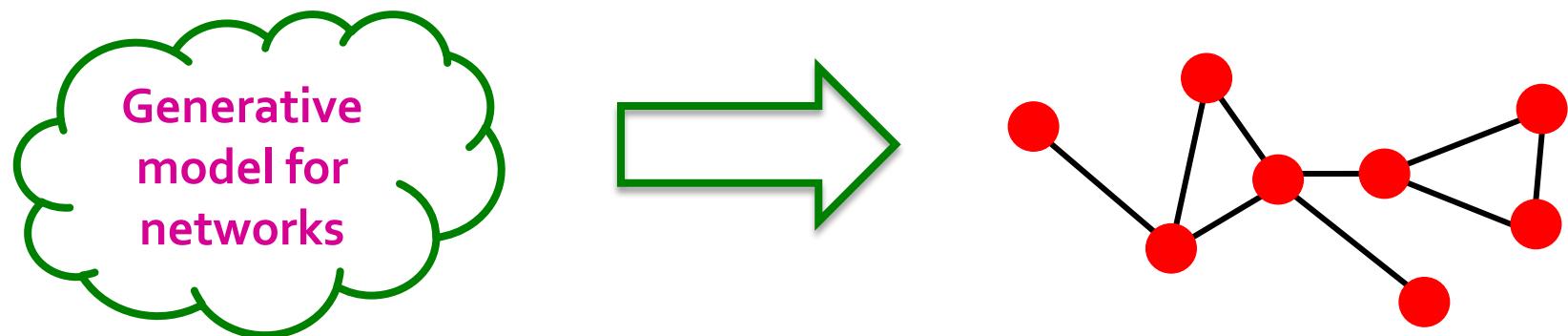


- 2) Given a network, find the “best” model



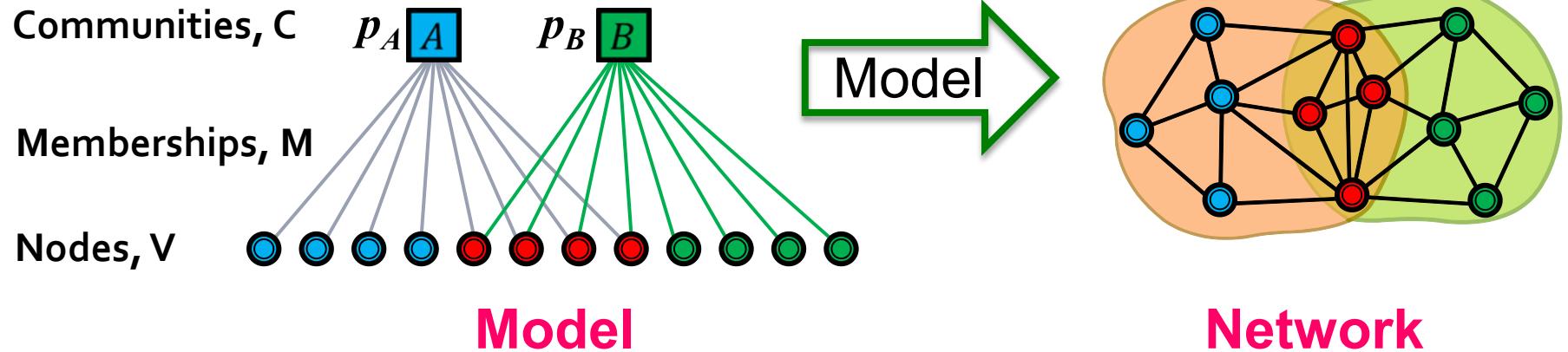
Model of networks

- Goal: Define a model that can generate networks
 - The model will have a set of “parameters” that we will later want to estimate (and detect communities)



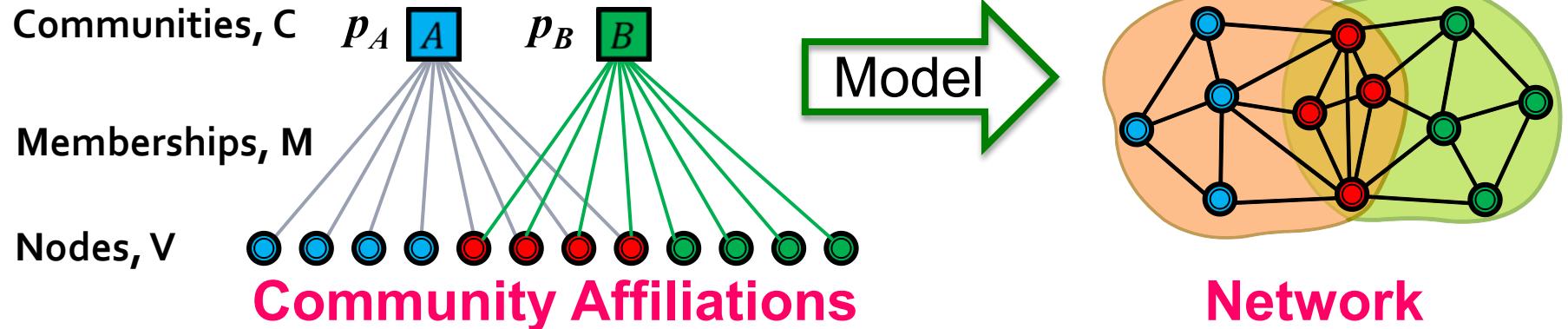
- Q: Given a set of nodes, how do communities “generate” edges of the network?

Community-Affiliation Graph



- **AMG: Affiliation Graph Model:** a generative model $B(V, C, M, \{p_c\})$ for graphs:
 - Nodes V , Communities C , Memberships M
 - Each community c has a single probability p_c
 - Later we fit the model to networks to detect communities

AGM: Generative Process



- **AGM generates the links: For each**
 - For each pair of nodes in community A , we connect them with prob. p_A
 - **The overall edge probability is:**

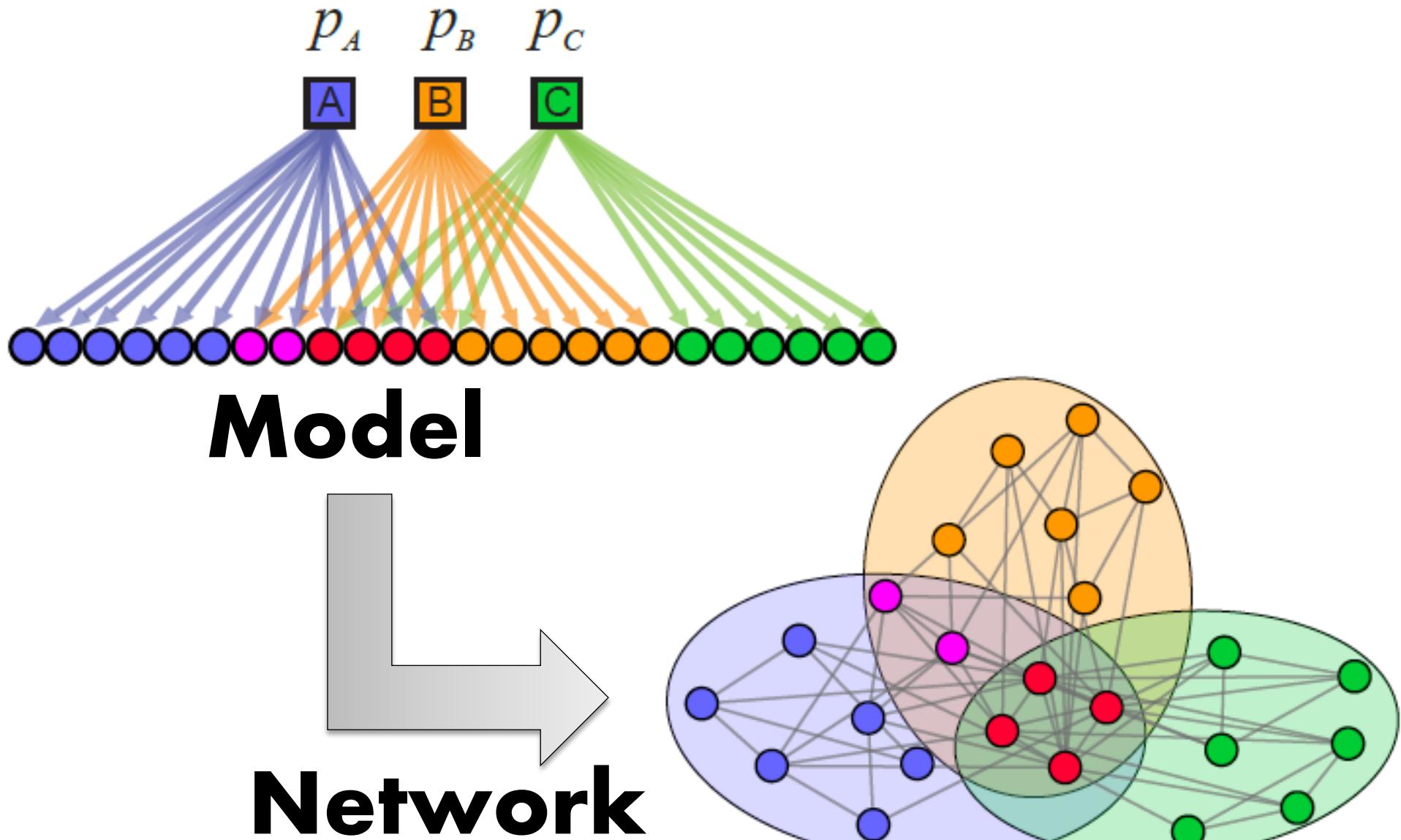
$$P(u, v) = 1 - \prod_{c \in M_u \cap M_v} (1 - p_c)$$

If u, v share no communities: $P(u, v) = \epsilon$

M_u set of communities
node u belongs to

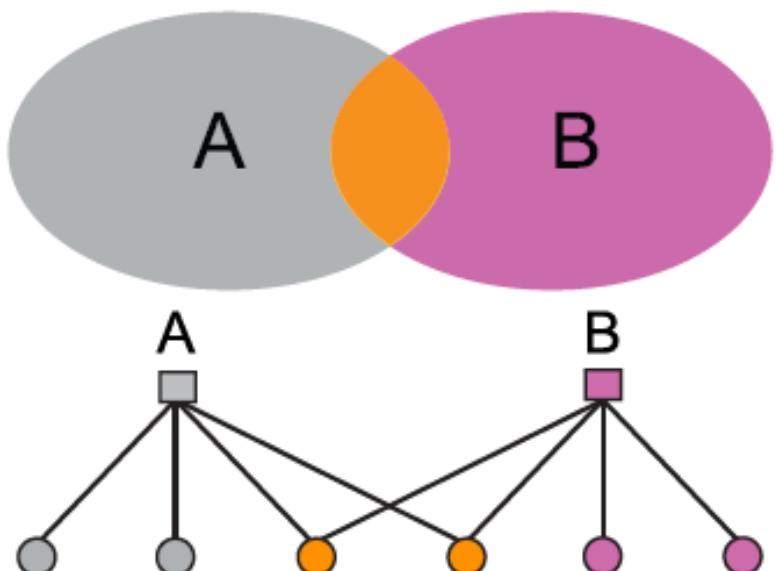
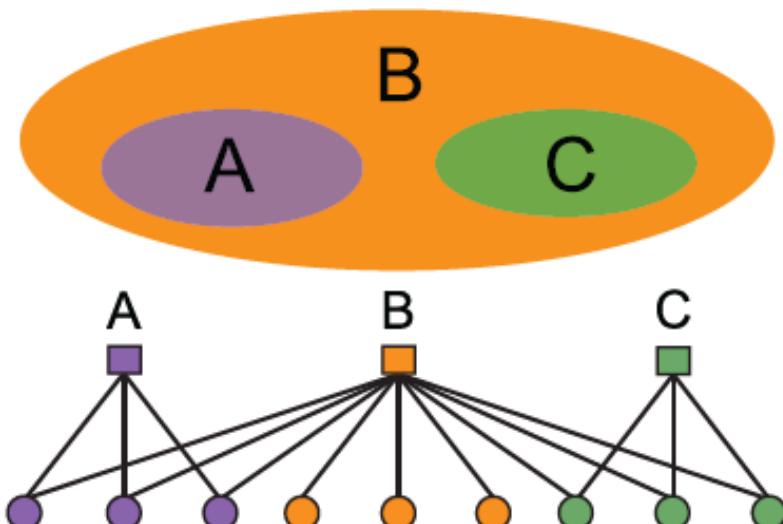
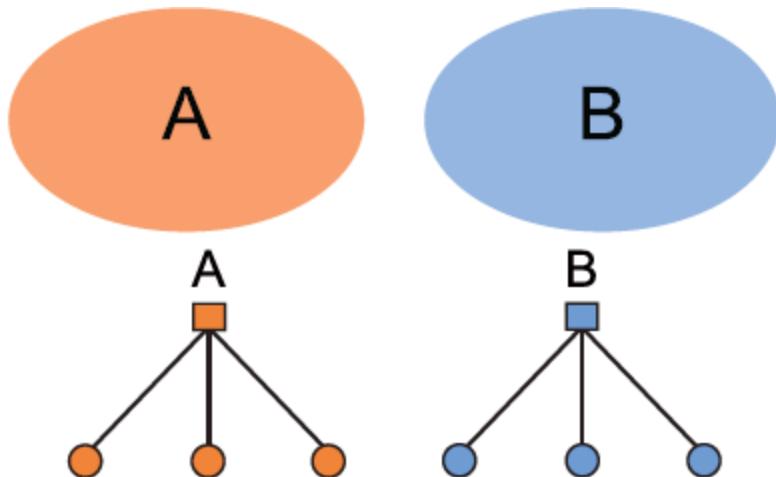
Think of this as an “OR” function: If at least 1 community says “YES” we create an edge

Recap: AGM networks



AGM: Flexibility

- AGM can express a variety of community structures:
Non-overlapping,
Overlapping, Nested

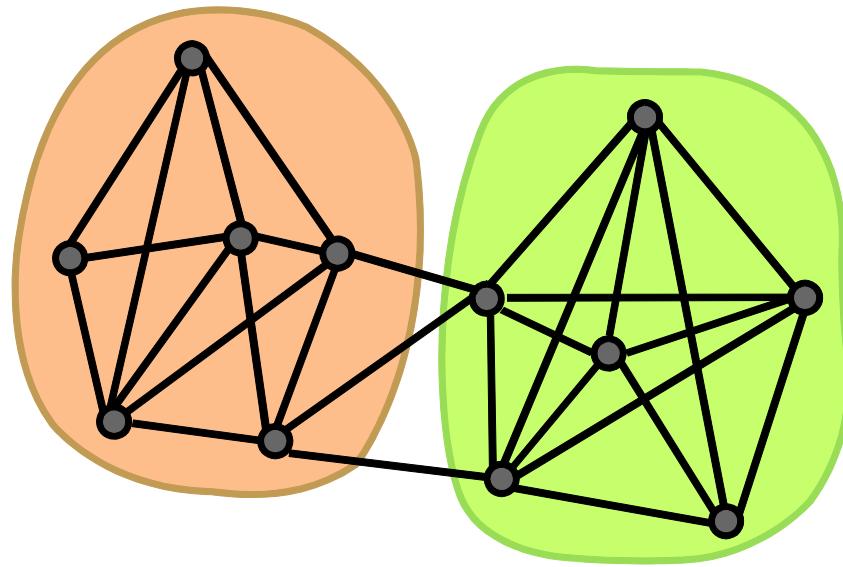


Spectral Graph Partitioning: Graph Laplacian Matrix

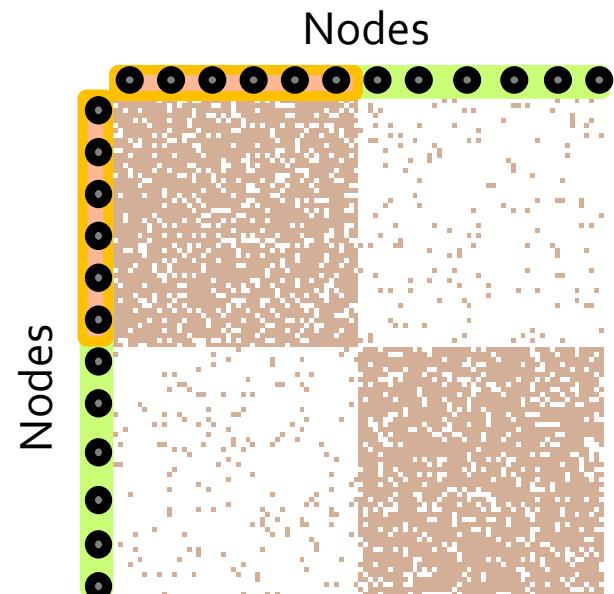
Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Finding Clusters



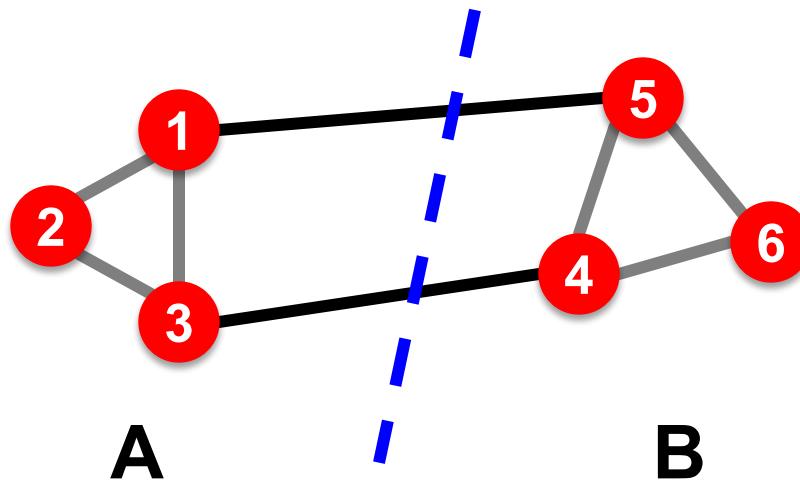
Network



Adjacency matrix

Graph Partitioning

- **Task:** Partition the graph into two pieces such the resulting pieces have low conductance



- **How do we efficiently find a good partition?**
 - **Problem:** Computing optimal cut is NP-hard

Spectral Graph Partitioning

- A : adjacency matrix of undirected \mathbf{G}
 - $A_{ij} = 1$ if (i, j) is an edge, else 0
- x is a vector in \mathbb{R}^n with components (x_1, \dots, x_n)
 - Think of it as a label/value of each node of G
- **What is the meaning of $A \cdot x$?**

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$y_i = \sum_{j=1}^n A_{ij} x_j = \sum_{(i,j) \in E} x_j$$

- **Entry y_i is a sum of labels x_j of neighbors of i**

What is the meaning of $A \cdot x$?

- **j^{th} coordinate of $A \cdot x$** :

- Sum of the x -values of neighbors of j
- Make this a new value at node j

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$
$$A \cdot x = \lambda \cdot x$$

- **Spectral Graph Theory:**

- Analyze the “spectrum” of matrix representing G
- **Spectrum:** Eigenvectors x_i of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues λ_i : $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$

$$\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$$

Dimensionality Reduction: CUR Decomposition

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



CUR Decomposition

Frobenius norm:
 $\|X\|_F = \sqrt{\sum_{ij} X_{ij}^2}$

- Goal: Express A as a product of matrices C, U, R
Make $\|A - C \cdot U \cdot R\|_F$ small
- “Constraints” on C and R :

$$\left(\begin{array}{c|c|c} \textcolor{red}{|} & \textcolor{blue}{|} & \textcolor{darkbrown}{|} \\ & A & \\ \textcolor{red}{|} & \textcolor{blue}{|} & \textcolor{darkbrown}{|} \end{array} \right) \approx \left(\begin{array}{c|c|c|c|c|c} \textcolor{red}{|} & \textcolor{red}{|} & \textcolor{red}{|} & \textcolor{blue}{|} & \textcolor{blue}{|} & \textcolor{darkbrown}{|} \\ C & & & & & \end{array} \right) \cdot \left(\begin{array}{c} U \end{array} \right) \cdot \left(\begin{array}{c} R \end{array} \right)$$

A C U R

CUR Decomposition

Frobenius norm:
 $\|X\|_F = \sqrt{\sum_{ij} X_{ij}^2}$

- Goal: Express A as a product of matrices C, U, R
Make $\|A - C \cdot U \cdot R\|_F$ small
- “Constraints” on C and R :

$$\begin{pmatrix} \text{---} \\ A \\ \text{---} \end{pmatrix} \approx \begin{pmatrix} \text{---} \\ C \\ \text{---} \end{pmatrix} \cdot \begin{pmatrix} \text{---} \\ U \\ \text{---} \end{pmatrix} \cdot \begin{pmatrix} \text{---} \\ R \\ \text{---} \end{pmatrix}$$

A C U R

Pseudo-inverse of
the intersection of C and R

CUR: Provably good approx. to SVD

- **Let:**

\mathbf{A}_k be the “best” rank k approximation to \mathbf{A} (that is, \mathbf{A}_k is SVD of \mathbf{A})

Theorem [Mahoney & Drineas]

CUR in $O(m \cdot n)$ time achieves

- $\|\mathbf{A} - \text{CUR}\|_F \leq \|\mathbf{A} - \mathbf{A}_k\|_F + \varepsilon \|\mathbf{A}\|_F$

with probability at least $1 - \delta$, by picking

- $O(k \log(1/\delta)/\varepsilon^2)$ columns, and
- $O(k^2 \log^3(1/\delta)/\varepsilon^6)$ rows

In practice:
Pick $4k$ cols/rows

CUR: How it Works

- Sampling columns (similarly for rows):
- ColumnSelect algorithm:

Input: matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, sample size c

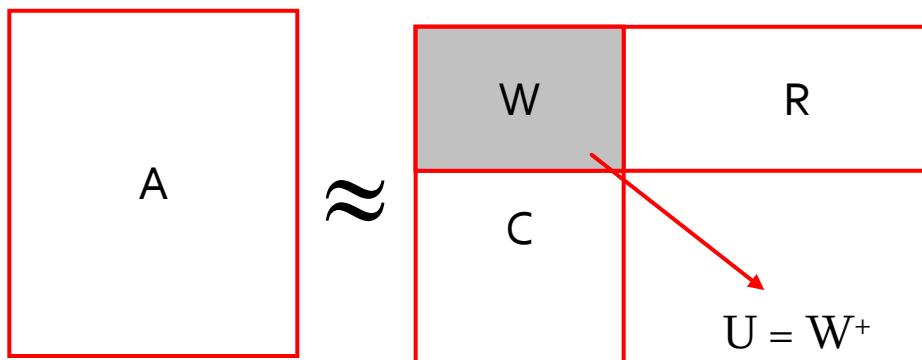
Output: $\mathbf{C}_d \in \mathbb{R}^{m \times c}$

1. for $x = 1 : n$ [column distribution]
2. $P(x) = \sum_i \mathbf{A}(i, x)^2 / \sum_{i,j} \mathbf{A}(i, j)^2$
3. for $i = 1 : c$ [sample columns]
4. Pick $j \in 1 : n$ based on distribution $P(x)$
5. Compute $\mathbf{C}_d(:, i) = \mathbf{A}(:, j) / \sqrt{cP(j)}$

Note this is a randomized algorithm, same column can be sampled more than once

Computing U

- Let \mathbf{W} be the “intersection” of sampled columns \mathbf{C} and rows \mathbf{R}
 - Let SVD of $\mathbf{W} = \mathbf{X} \mathbf{Z} \mathbf{Y}^T$
- Then: $\mathbf{U} = \mathbf{W}^+ = \mathbf{Y} \mathbf{Z}^+ \mathbf{X}^T$
 - \mathbf{Z}^+ : **reciprocals of non-zero singular values**: $Z_{ii}^+ = 1/Z_{ii}$
 - \mathbf{W}^+ is the “**pseudoinverse**”



Why pseudoinverse works?
 $\mathbf{W} = \mathbf{X} \mathbf{Z} \mathbf{Y}$ then $\mathbf{W}^{-1} = \mathbf{X}^{-1} \mathbf{Z}^{-1} \mathbf{Y}^{-1}$
Due to orthonormality
 $\mathbf{X}^{-1} = \mathbf{X}^T$ and $\mathbf{Y}^{-1} = \mathbf{Y}^T$
Since \mathbf{Z} is diagonal $\mathbf{Z}^{-1} = 1/Z_{ii}$
Thus, if \mathbf{W} is non-singular,
pseudoinverse is the true
inverse

CUR: Provably good approx. to SVD

- For example:

- Select $c = O\left(\frac{k \log k}{\epsilon^2}\right)$ columns of A using **ColumnSelect algorithm**
- Select $r = O\left(\frac{k \log k}{\epsilon^2}\right)$ rows of A using **ColumnSelect algorithm**
- Set $U = W^+$
- **Then:** $\|A - CUR\|_F \leq (2 + \epsilon) \|A - A_k\|_F$ with probability 98%

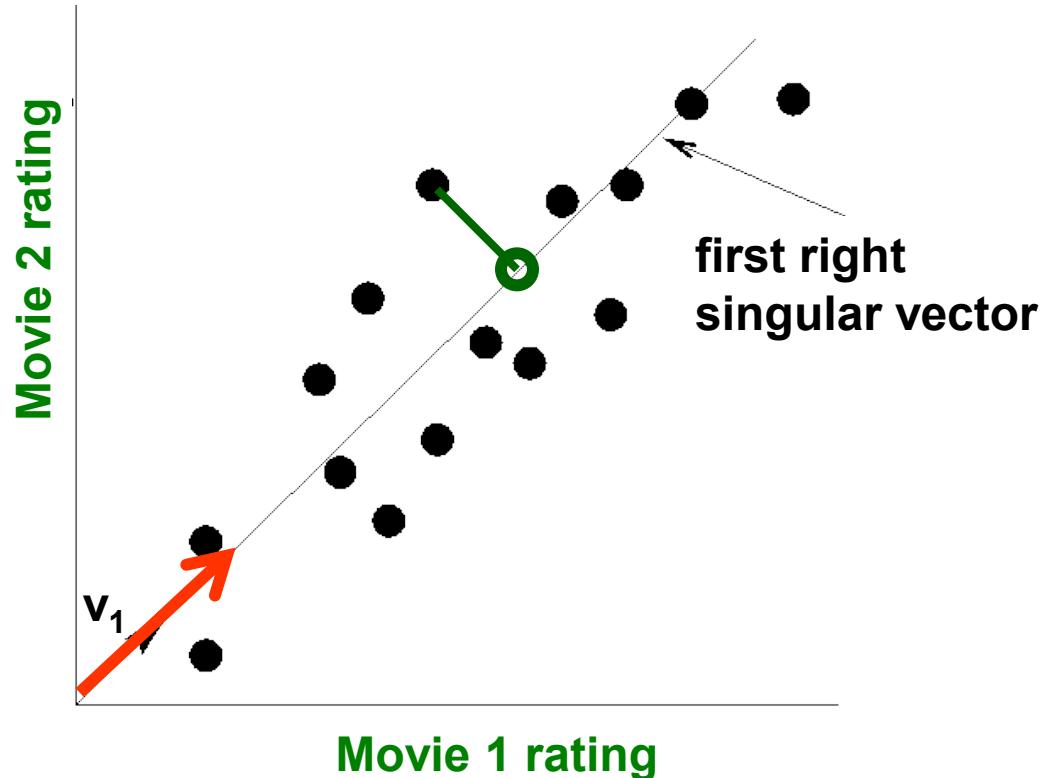
Dimensionality Reduction with SVD

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



SVD – Dimensionality Reduction

- SVD gives ‘best’ axis to project on:
 - ‘best’ = min sum of squares of projection errors
- In other words, minimum reconstruction error

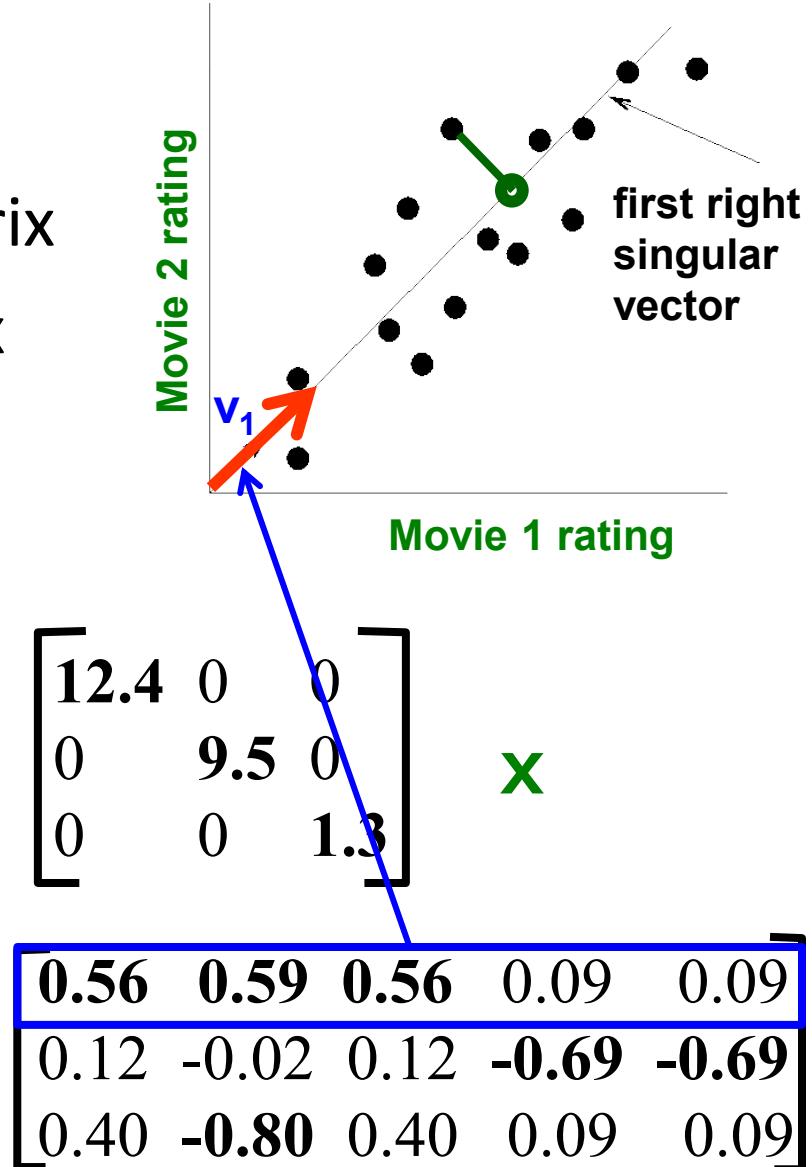


SVD – Dimensionality Reduction

■ $A = U \Sigma V^T$ - example:

- V : “movie-to-concept” matrix
- U : “user-to-concept” matrix

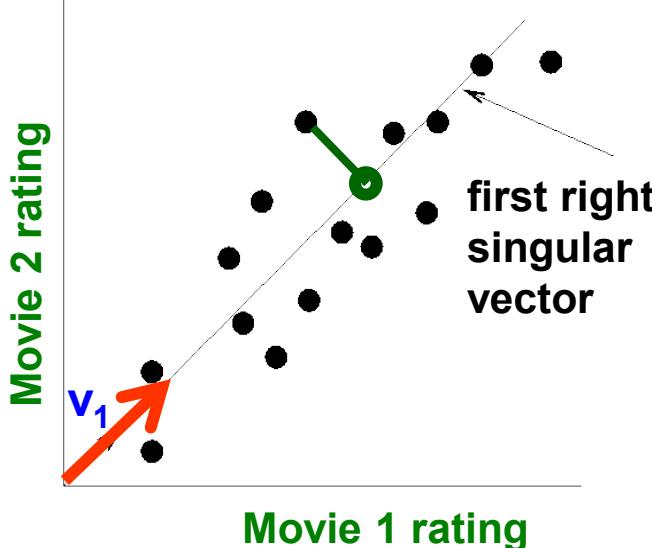
$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$



SVD – Dimensionality Reduction

- $A = U \Sigma V^T$ - example:

variance ('spread')
on the v_1 axis


$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

The diagram illustrates the decomposition of a matrix A into U, Sigma, and V^T. The matrix A represents movie ratings. The matrix on the right is circled in green, with a green arrow pointing to it from the text "variance ('spread') on the v_1 axis". This matrix represents the variance along the first principal component (v_1). The matrix in the middle is also circled in green, representing the singular value matrix Σ . The matrix on the far right is the matrix V^T , which represents the right singular vectors. A scatter plot shows the data points and the first right singular vector v_1 as a red arrow originating from the origin.

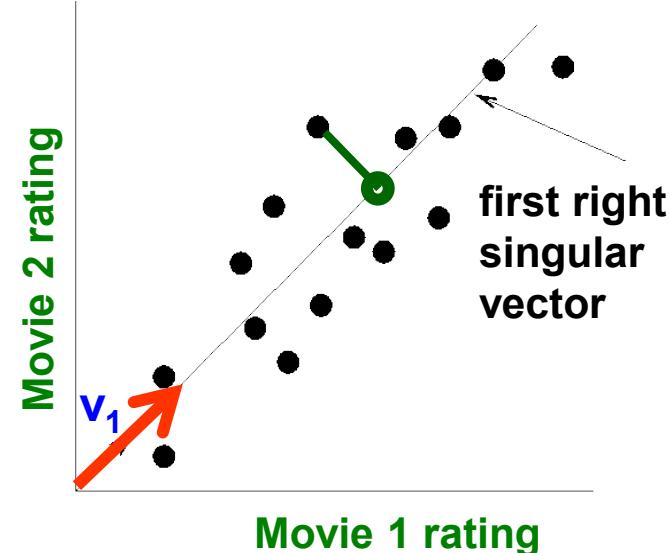
SVD – Dimensionality Reduction

$A = U \Sigma V^T$ - example:

- $U \Sigma$: Gives the coordinates of the points in the projection axis

1	1	1	0	0
3	3	3	0	0
4	4	4	0	0
5	5	5	0	0
0	2	0	4	4
0	0	0	5	5
0	1	0	2	2

Projection of users on the “Sci-Fi” axis ($(U \Sigma)^T$):



1.61	0.19	-0.01
5.08	0.66	-0.03
6.82	0.85	-0.05
8.43	1.04	-0.06
1.86	-5.60	0.84
0.86	-6.93	-0.87
0.86	-2.75	0.41

SVD – Dimensionality Reduction

More details

- Q: How exactly is dim. reduction done?

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

SVD – Dimensionality Reduction

More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & \cancel{1.3} \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

SVD – Dimensionality Reduction

More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

SVD – Dimensionality Reduction

More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

SVD – Dimensionality Reduction

More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.13 & 0.02 \\ 0.41 & 0.07 \\ 0.55 & 0.09 \\ 0.68 & 0.11 \\ 0.15 & -0.59 \\ 0.07 & -0.73 \\ 0.07 & -0.29 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \end{bmatrix}$$

SVD – Dimensionality Reduction

More details

- Q: How exactly is dim. reduction done?
- A: Set smallest singular values to zero

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} 0.92 & 0.95 & 0.92 & 0.01 & 0.01 \\ 2.91 & 3.01 & 2.91 & -0.01 & -0.01 \\ 3.90 & 4.04 & 3.90 & 0.01 & 0.01 \\ 4.82 & 5.00 & 4.82 & 0.03 & 0.03 \\ 0.70 & 0.53 & 0.70 & 4.11 & 4.11 \\ -0.69 & 1.34 & -0.69 & 4.78 & 4.78 \\ 0.32 & 0.23 & 0.32 & 2.01 & 2.01 \end{bmatrix}$$

Frobenius norm:

$$\|M\|_F = \sqrt{\sum_{ij} M_{ij}^2}$$

$$\|A-B\|_F = \sqrt{\sum_{ij} (A_{ij}-B_{ij})^2}$$

is “small”

CUR: Pros & Cons

+ Easy interpretation

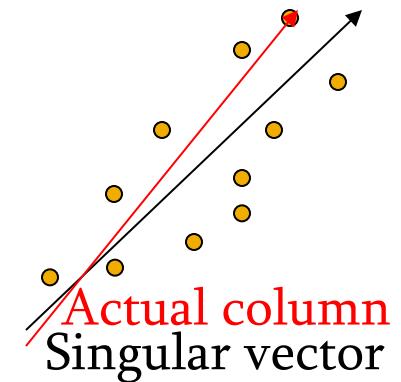
- Since the basis vectors are actual columns and rows

+ Sparse basis

- Since the basis vectors are actual columns and rows

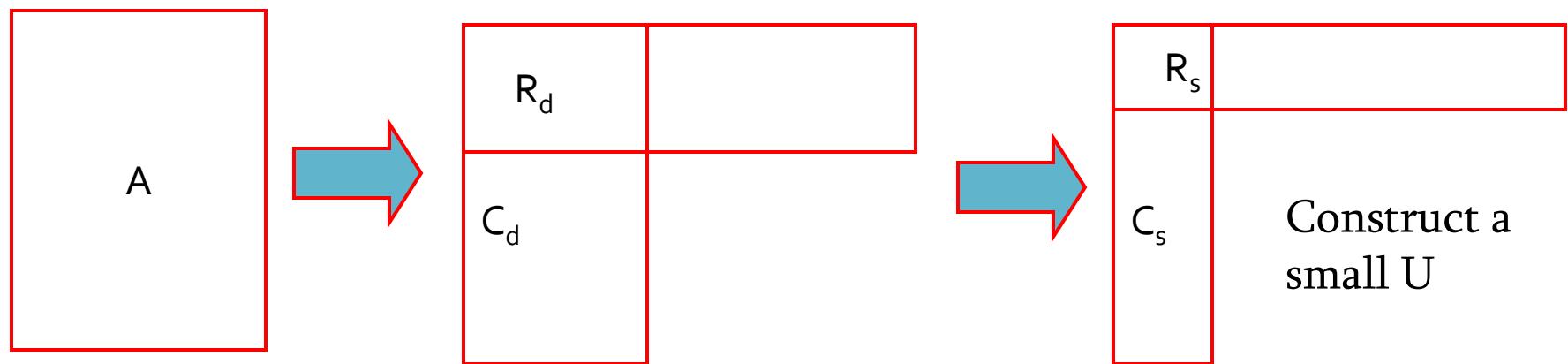
- Duplicate columns and rows

- Columns of large norms will be sampled many times



Solution

- If we want to get rid of the duplicates:
 - Throw them away
 - Scale (multiply) the columns/rows by the square root of the number of duplicates



SVD vs. CUR

$$\text{SVD: } A = U \Sigma V^T$$

Huge but sparse

sparse and small

Big and dense

$$\text{CUR: } A = C U R$$

Huge but sparse

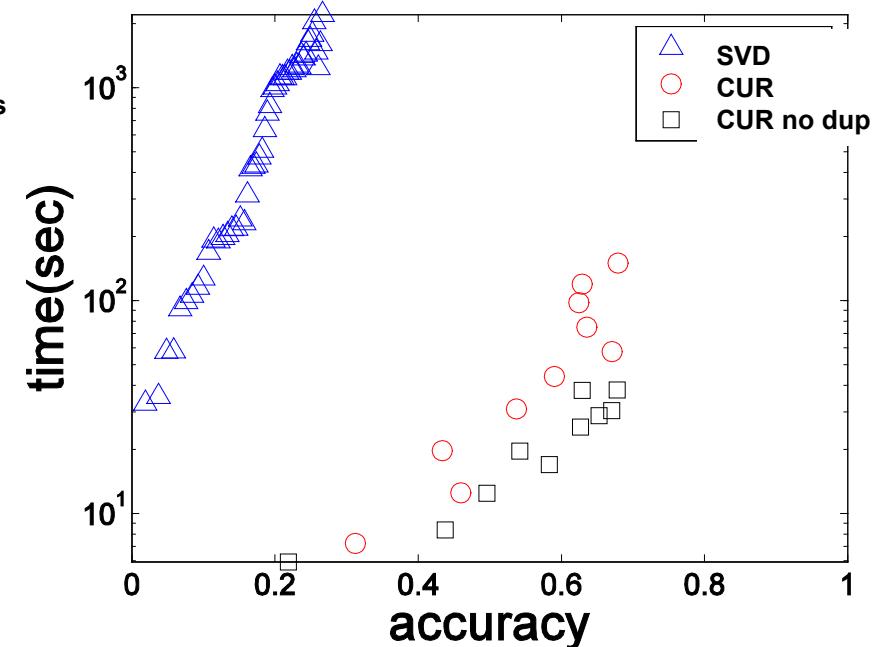
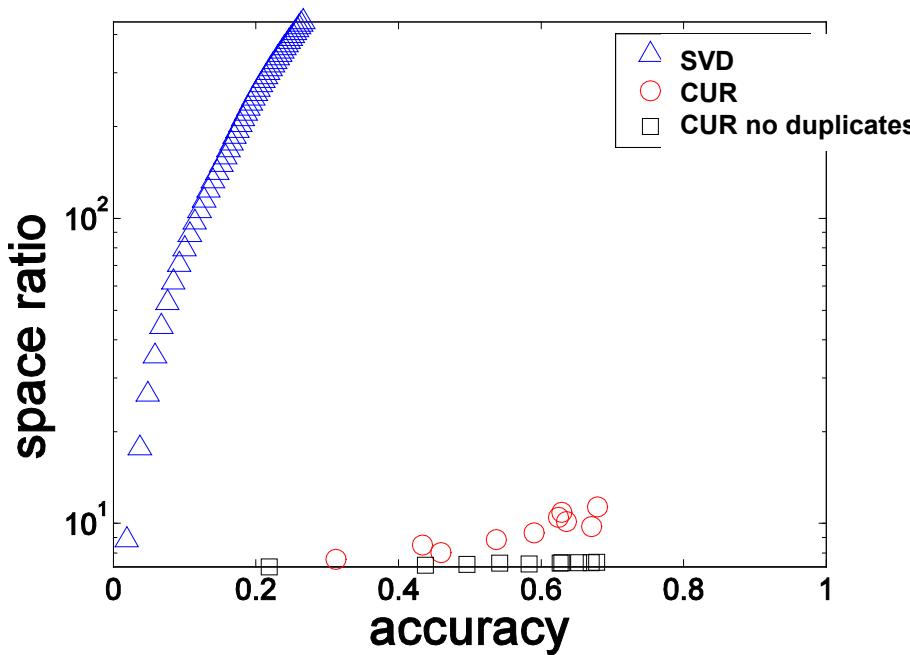
dense but small

Big but sparse

Simple Experiment

- **DBLP bibliographic data**
 - Author-to-conference big sparse matrix
 - A_{ij} : Number of papers published by author i at conference j
 - 428K authors (rows), 3659 conferences (columns)
 - Very sparse
- **Want to reduce dimensionality**
 - How much time does it take?
 - What is the reconstruction error?
 - How much space do we need?

Results: DBLP- big sparse matrix



- **Accuracy:**
 - $1 - \text{relative sum squared errors}$
- **Space ratio:**
 - $\# \text{output matrix entries} / \# \text{input matrix entries}$
- **CPU time**

Finding the Latent Factors

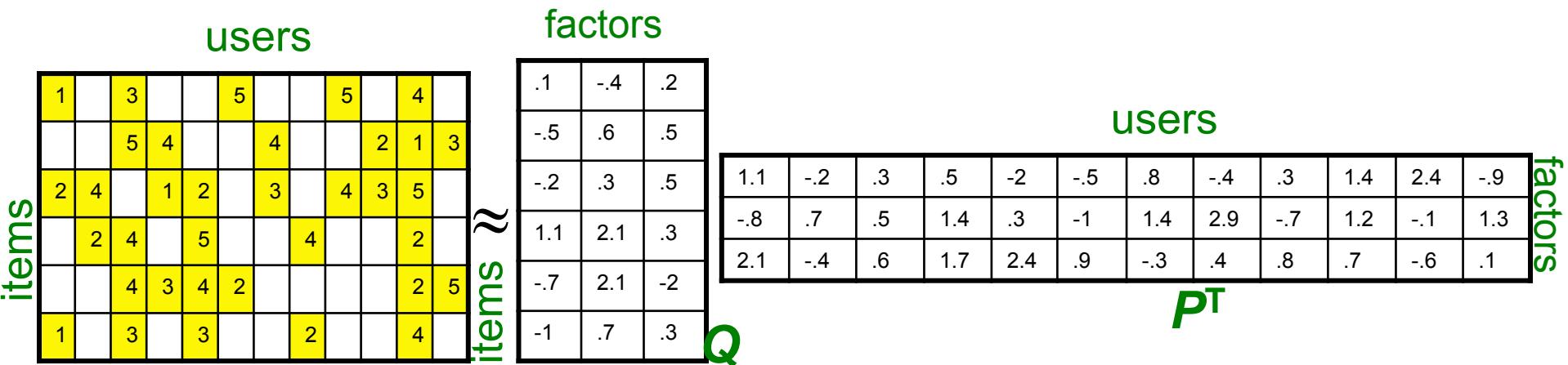
Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Latent Factor Models

- Our goal is to find P and Q such that:

$$\min_{P,Q} \sum_{(i,x) \in R} (r_{xi} - q_i \cdot p_x^T)^2$$



Dealing with Missing Entries

- Want to minimize SSE (that is RMSE) for unseen test data

- Idea: Minimize SSE on training data:

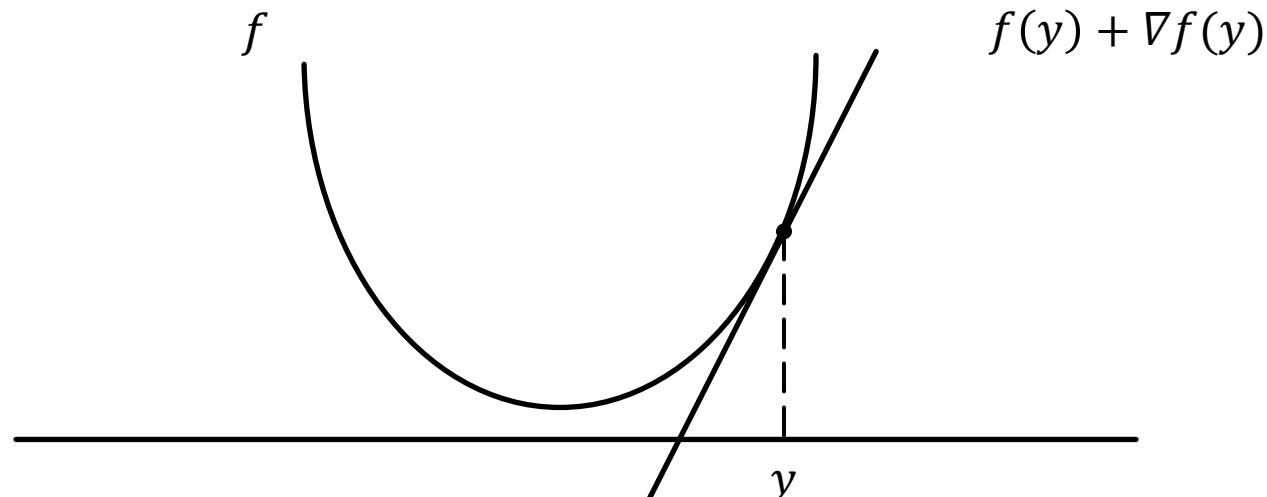
$$f(P, Q) = \sum_{(i,x) \in R} (r_{xi} - q_i \cdot p_x^T)^2$$

- Want large k (# of factors) to capture all the signals
- How to minimize our error function?

1	3	4		
3	5			5
	4	5		5
	3			
	3			
2			?	?
	2	1		?
	3		?	
1				

Detour: Minimizing a function

- A simple way to minimize a function $f(x)$:
 - Compute the take a derivative ∇f
 - Start at some point y and evaluate $\nabla f(y)$
 - Make a step in the reverse direction of the gradient: $y = y - \nabla f(y)$
 - Repeat until converged



Back to Our Problem

- Want to minimize SSE for unseen test data
- Idea: Minimize SSE on training data
 - Want large k (# of factors) to capture all the signals
 - But, SSE on test data begins to rise for $k > 2$
- This is a classical example of **overfitting**:
 - With too much freedom (too many free parameters) the model starts fitting noise
 - That is it fits too well the training data and thus **not generalizing** well to unseen test data

1	3	4			
3	5		5		
4	5		5		
3					
3					
2		?	?	?	?
	2	1		?	?
	3		?		
1					

Dealing with Missing Entries

- To solve overfitting we introduce regularization:

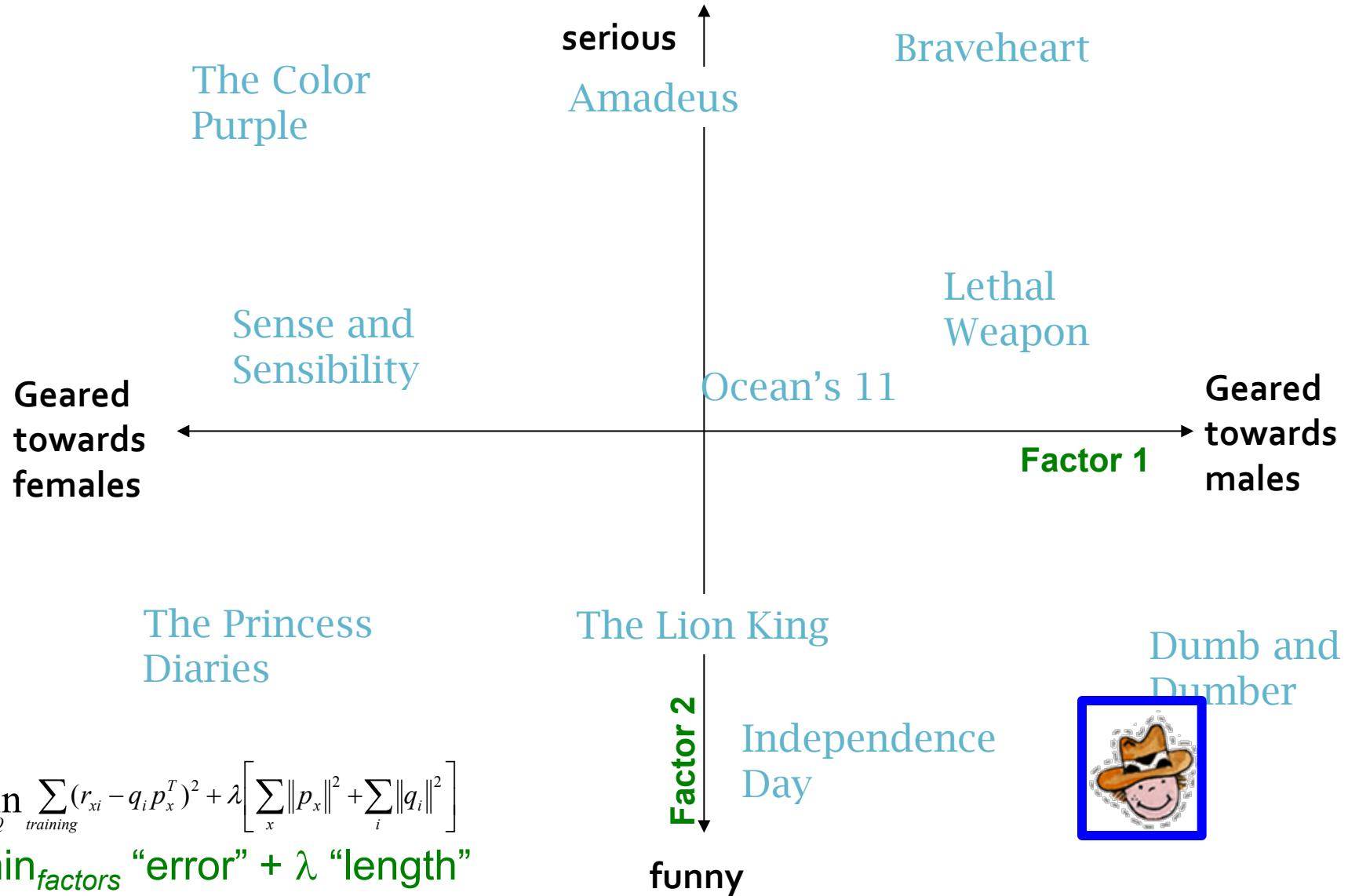
- Allow rich model where there are sufficient data
- Shrink aggressively where data are scarce

1	3	4		5
3	5			
4	5			5
3				
3				
2		?	?	?
	2	1		?
	3		?	
1				

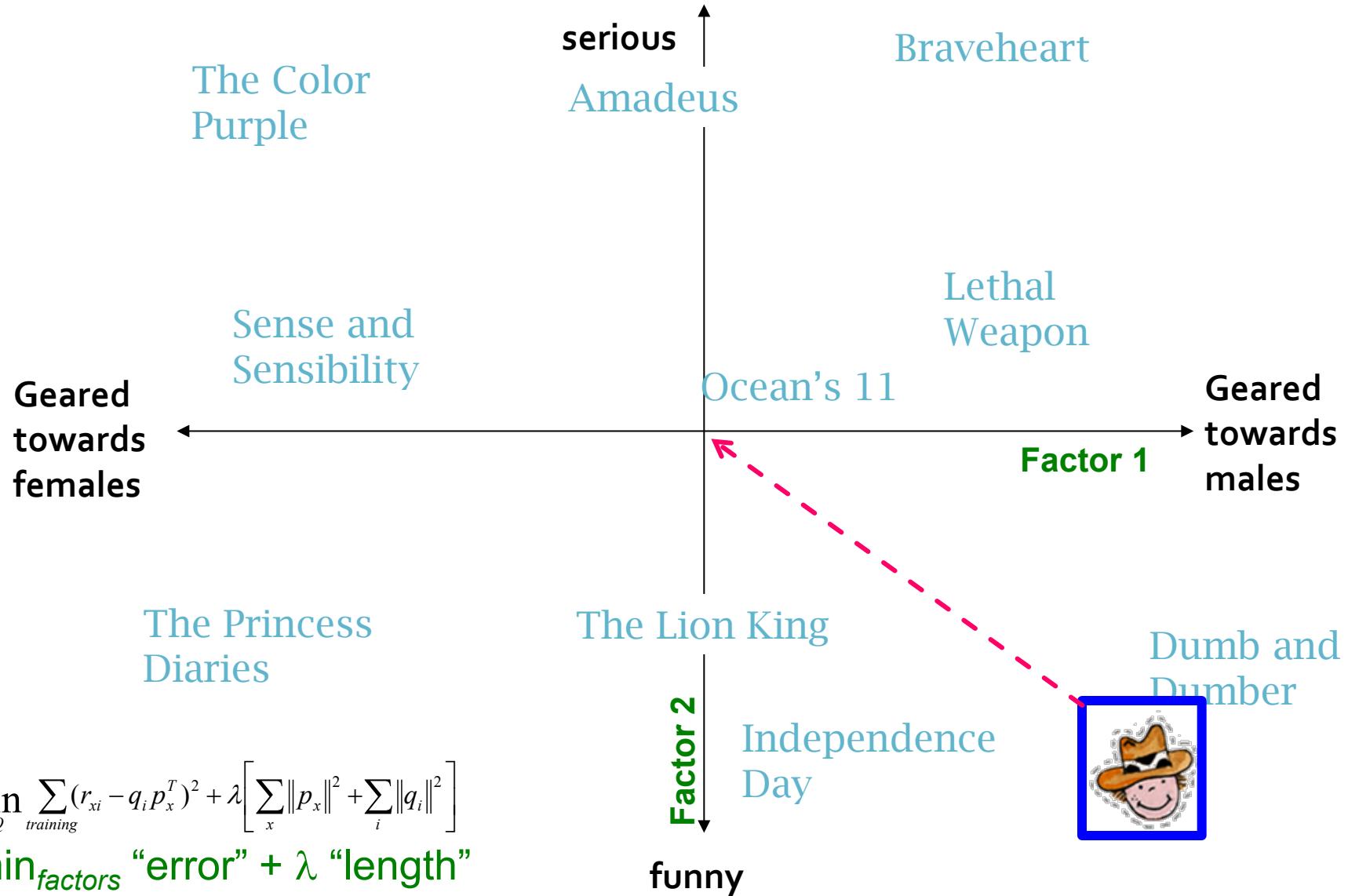
$$\min_{P,Q} \underbrace{\sum_{\text{training}} (r_{xi} - q_i p_x^T)^2}_{\text{"error"}} + \lambda \left[\underbrace{\sum_x \|p_x\|^2}_{\text{"length"}^2} + \underbrace{\sum_i \|q_i\|^2}_{\text{"length"}^2} \right]$$

λ user set regularization parameter

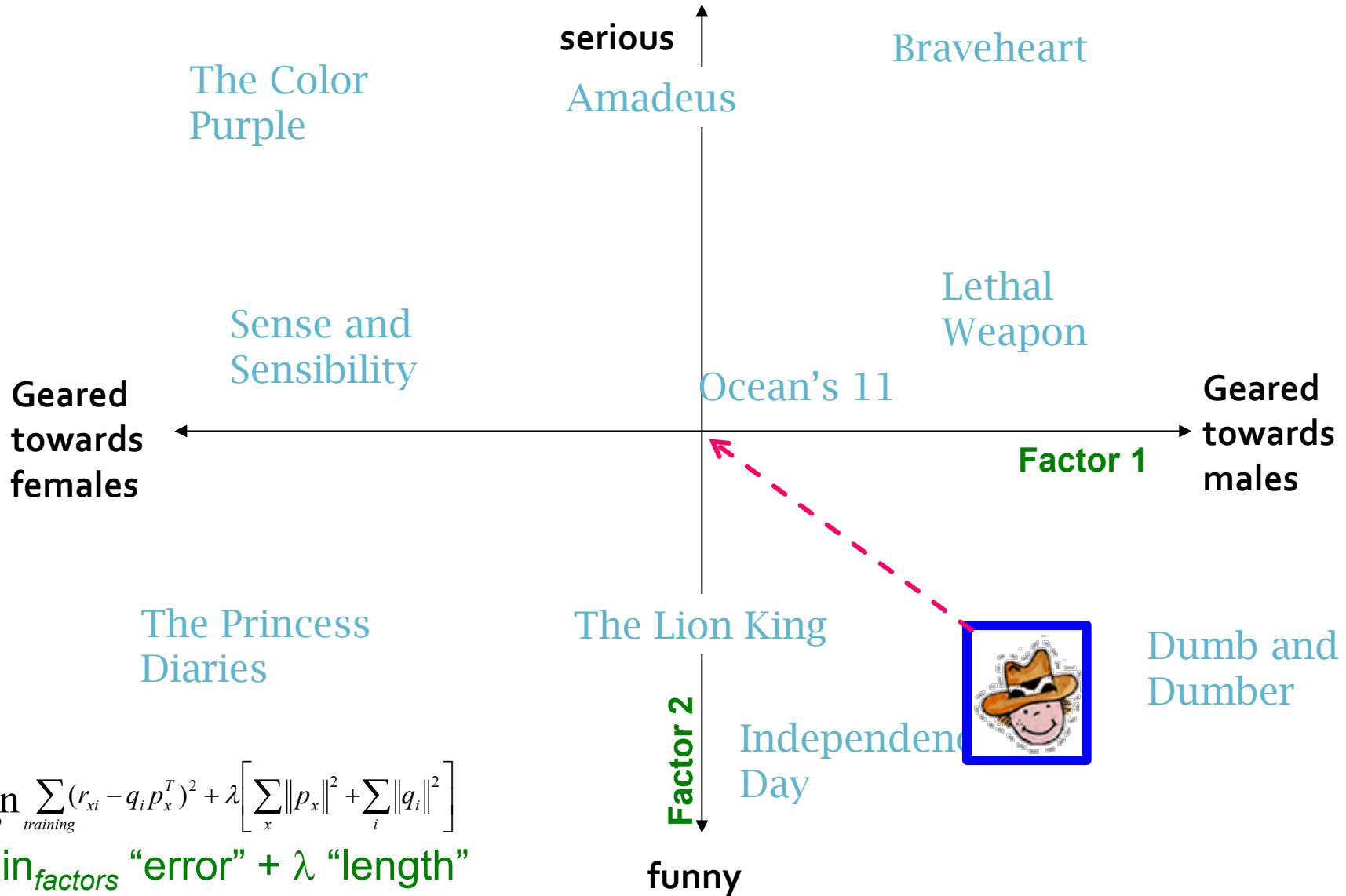
The Effect of Regularization



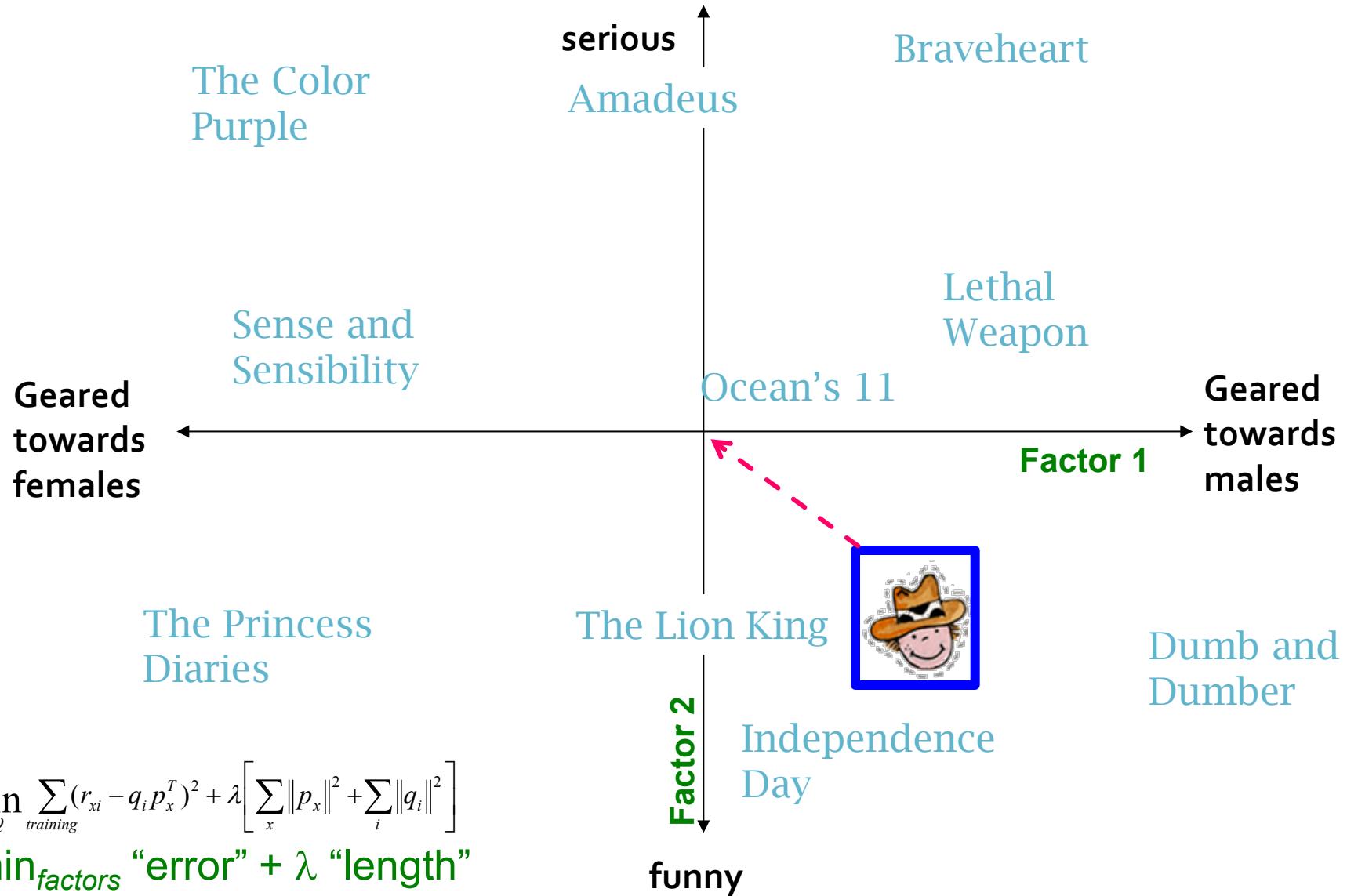
The Effect of Regularization



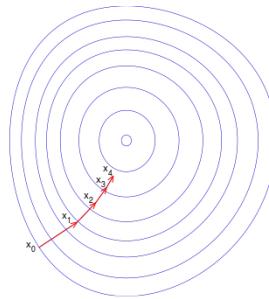
The Effect of Regularization



The Effect of Regularization



Stochastic Gradient Descent



Want to find matrices P and Q :

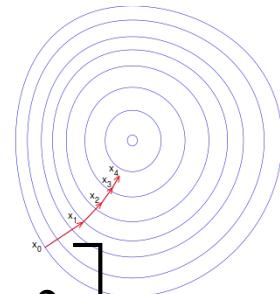
$$\min_{P,Q} \sum_{\text{training}} (r_{xi} - q_i p_x^T)^2 + \lambda \left[\sum_x \|p_x\|^2 + \sum_i \|q_i\|^2 \right]$$

■ Note:

- $\lambda \neq 0$ increases the value of the objective function
- But we do not care about the value of the objective function but P and Q that minimize the value
- And real our goal is to find P and Q on **seen ratings** so that we predict well the **unseen** ratings

1	3	4			
3	5				5
4		5			5
	3				
3					
2			?	?	
				?	
2	1				?
	3				?
1					

Stochastic Gradient Descent



$$\min_{P,Q} \sum_{training} (r_{xi} - q_i p_x^T)^2 + \lambda \left[\sum_x \|p_x\|^2 + \sum_i \|q_i\|^2 \right]$$

■ Gradient decent:

- Initialize P and Q (using SVD, pretend missing ratings are 0)
- Do gradient descent:

$$\blacksquare P \leftarrow P - \eta \cdot \nabla P$$

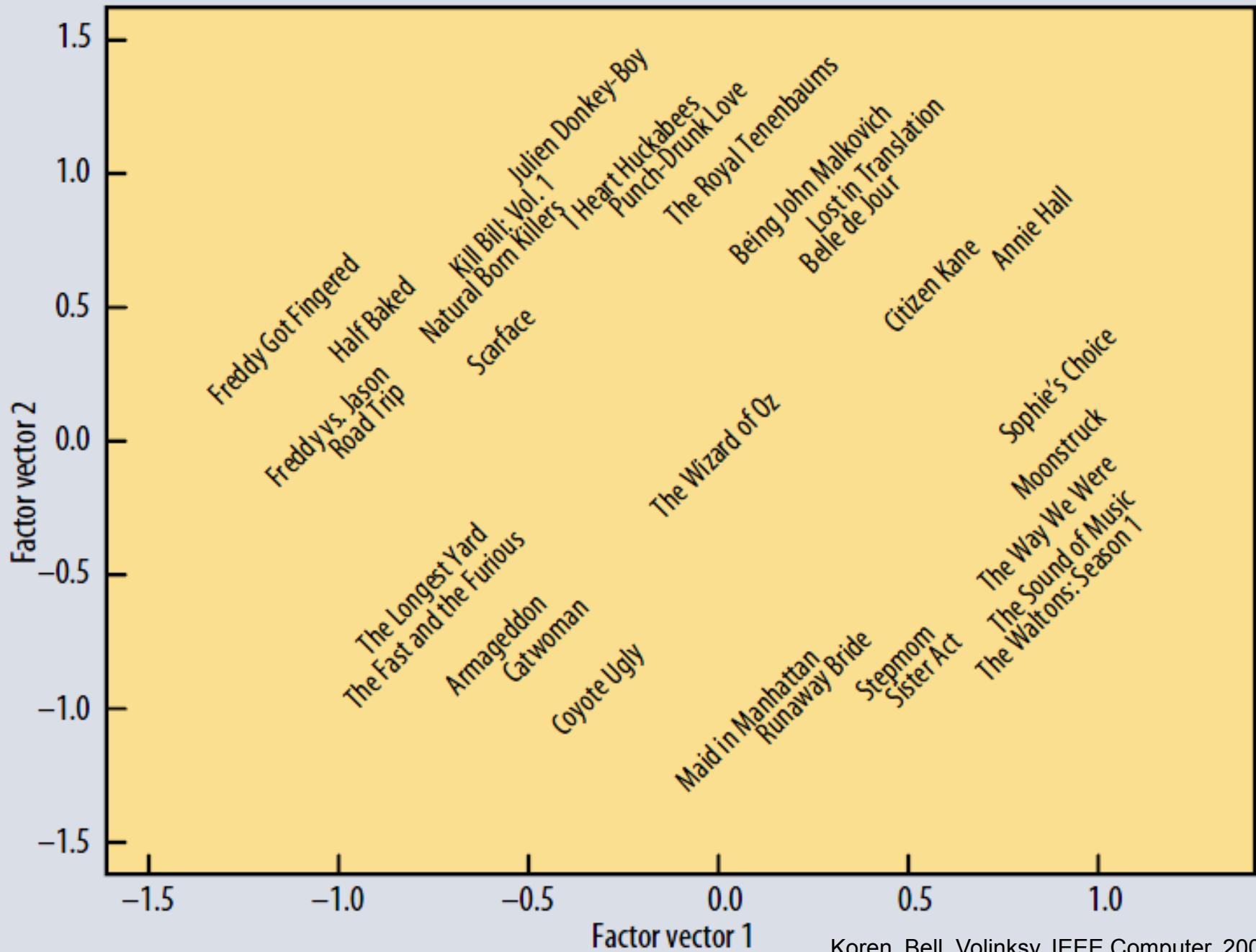
$$\blacksquare Q \leftarrow Q - \eta \cdot \nabla Q$$

where ∇Q is gradient/derivative of matrix Q :

$$\nabla Q = [\nabla q_{ik}] \text{ and } \nabla q_{ik} = \sum_{xi} -2(r_{xi} - q_i p_x^T) p_{xk} + 2\lambda q_{ik}$$

- Here q_{ik} is entry k of row q_i of matrix Q

- And similarly for ∇P



Latent Factor Recommender System

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University

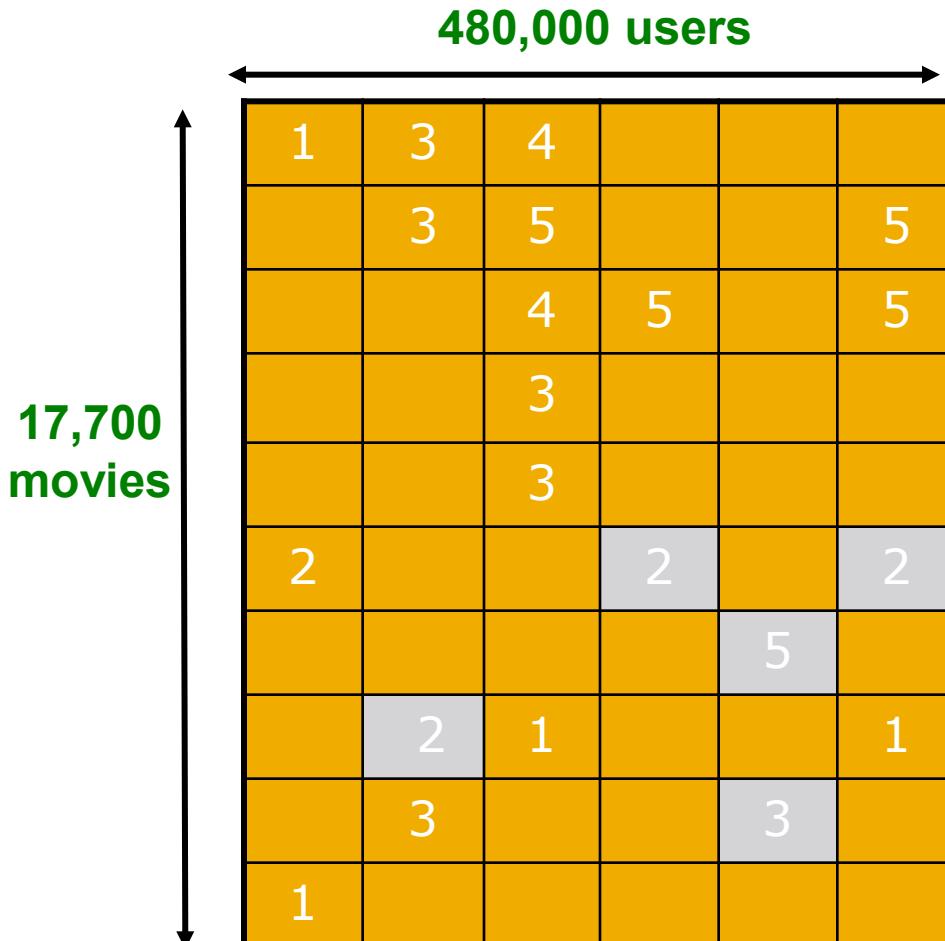


Recommendations via Optimization

- **Goal: Make good recommendations**
 - Quantify goodness using **RMSE**:
Lower RMSE \Rightarrow better recommendations
 - Want to make good recommendations on items that user has not yet seen. **Can't really do this!**
 - **Let's set build a system such that it works well on known (user, item) ratings**
And **hope** the system will also predict well the **unknown ratings**

1	3	4		
	3	5		5
		4	5	5
		3		
		3		
2			?	?
			?	?
	2	1		?
	3		?	
1				

The Netflix Utility Matrix R

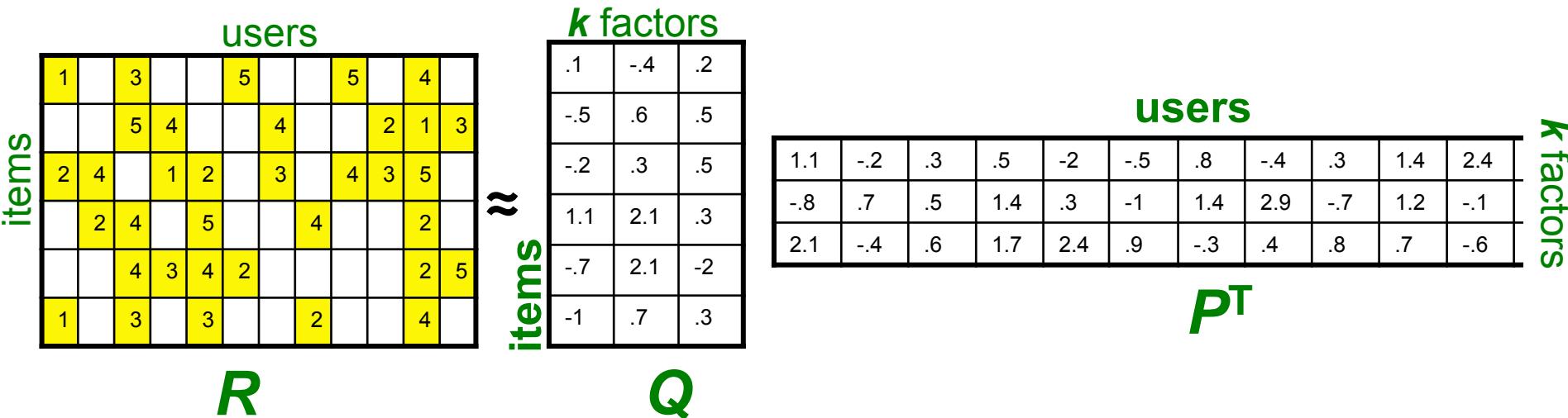


We want our system to predict well the hidden (known) ratings

Latent Factor Models

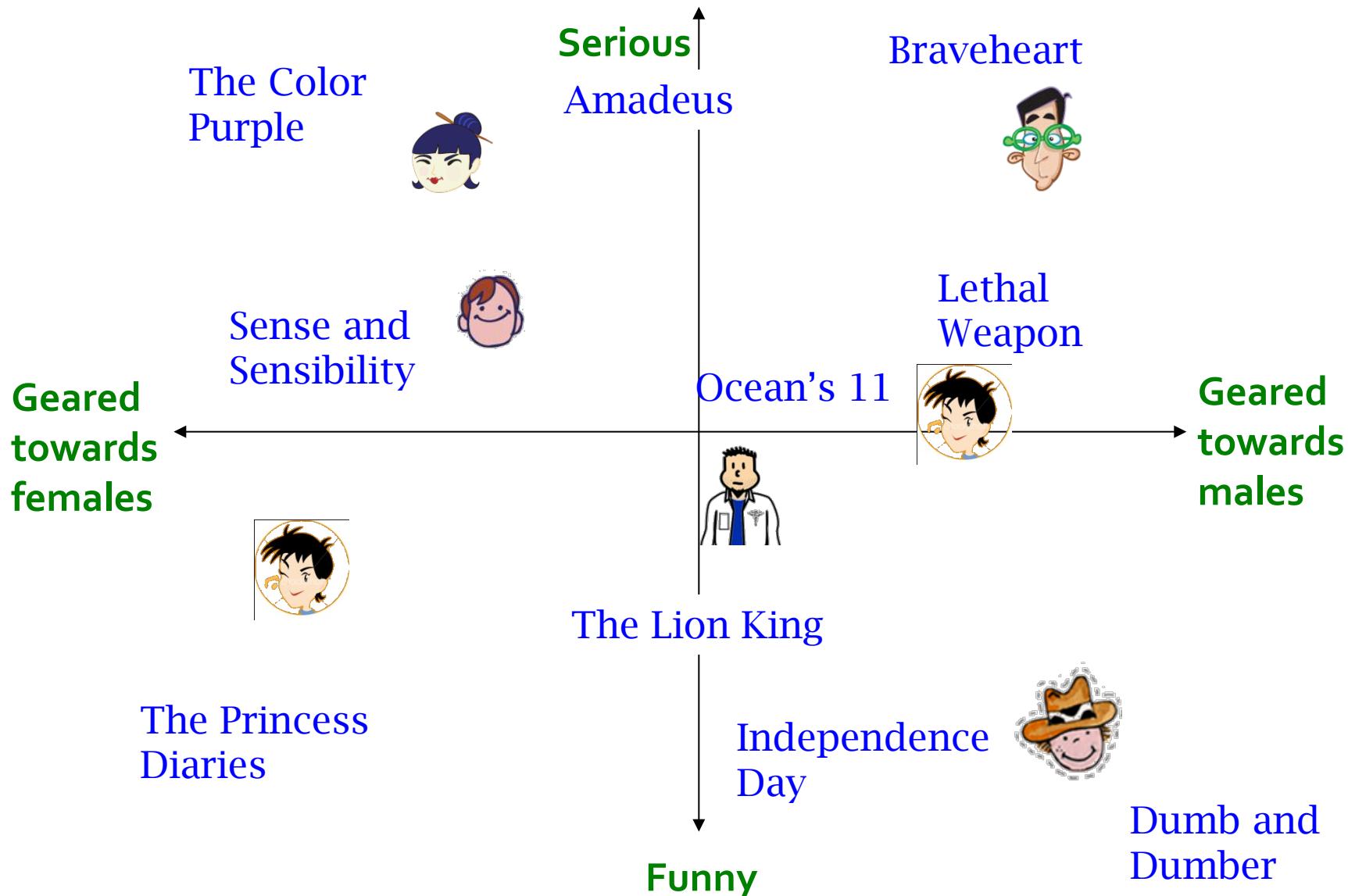
$$\text{SVD: } A = U \Sigma V^T$$

- “SVD” on Netflix data: $R \approx Q \cdot P^T$



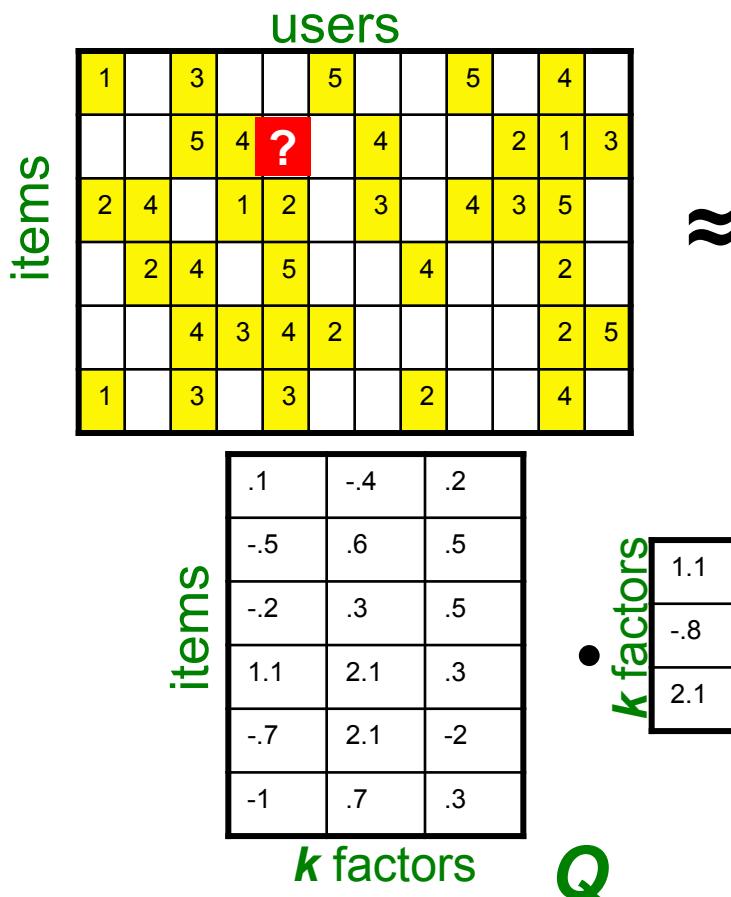
- For now let's assume we can approximate the rating matrix R as a product of “thin” $Q \cdot P^T$
 - R has missing entries but let's ignore that for now!
 - Basically, we will want the reconstruction error to be small on known ratings and we don't care about the values on the missing ones

Latent Factor Models (e.g., SVD)



Ratings as Products of Factors

- How to estimate the missing rating of user x for item i ?



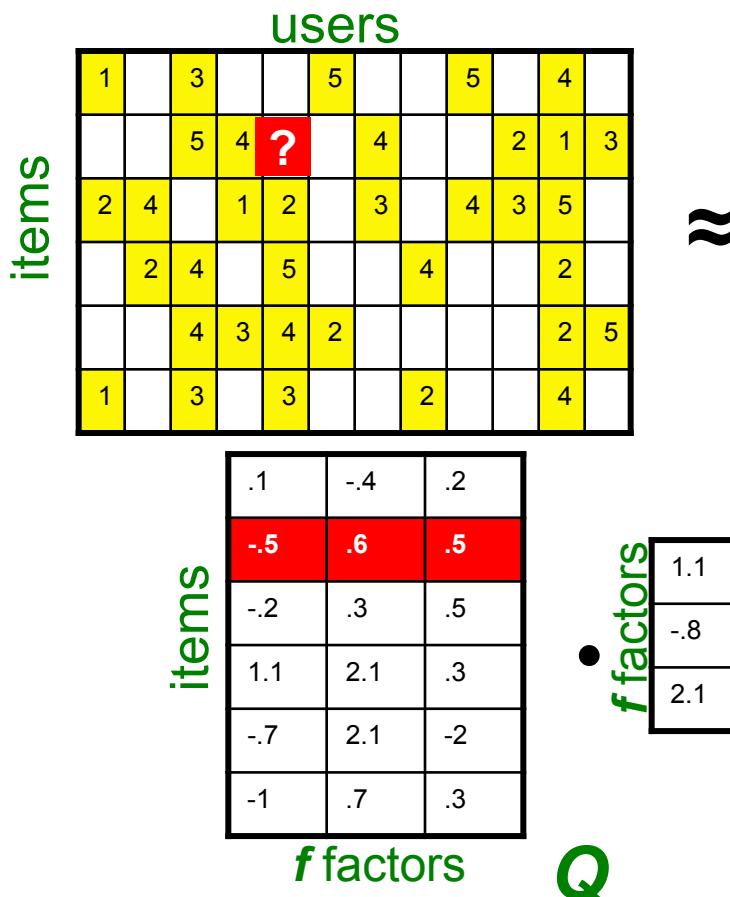
$$\hat{r}_{xi} = q_i \cdot p_x^T$$
$$= \sum_k q_{ik} \cdot p_{xk}$$

q_i = row i of Q

p_x = column x of P^T

Ratings as Products of Factors

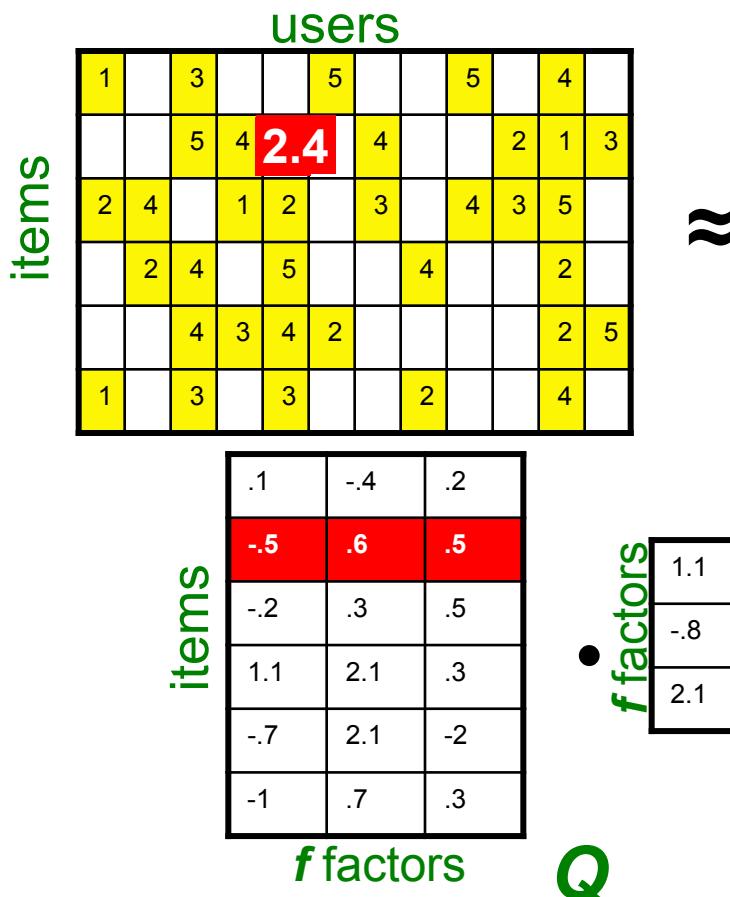
- How to estimate the missing rating of user x for item i ?



$$\begin{aligned}\hat{r}_{xi} &= q_i \cdot p_x^T \\ &= \sum_f q_{if} \cdot p_{xf} \\ q_i &= \text{row } i \text{ of } Q \\ p_x &= \text{column } x \text{ of } P^T\end{aligned}$$

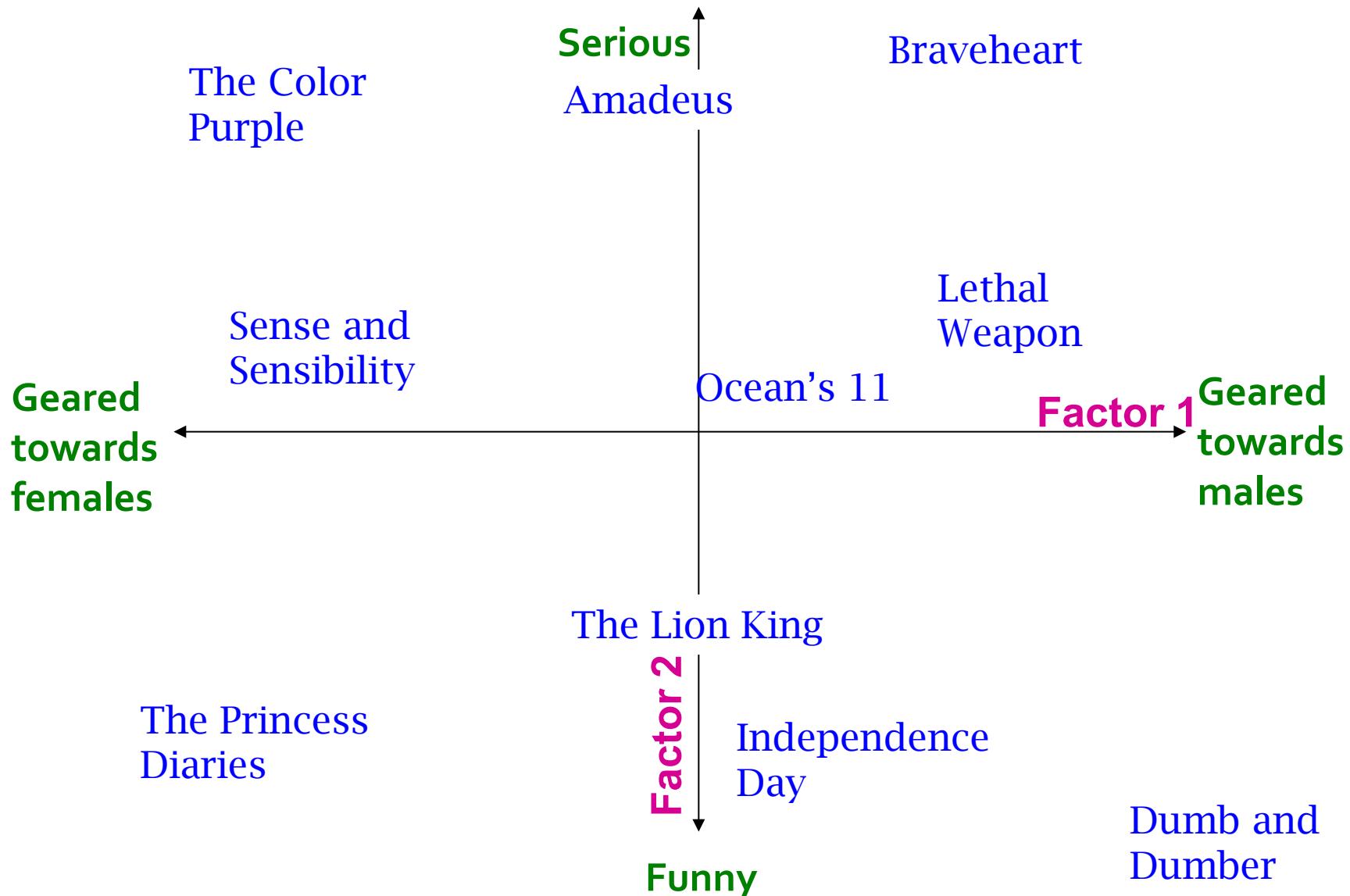
Ratings as Products of Factors

- How to estimate the missing rating of user x for item i ?

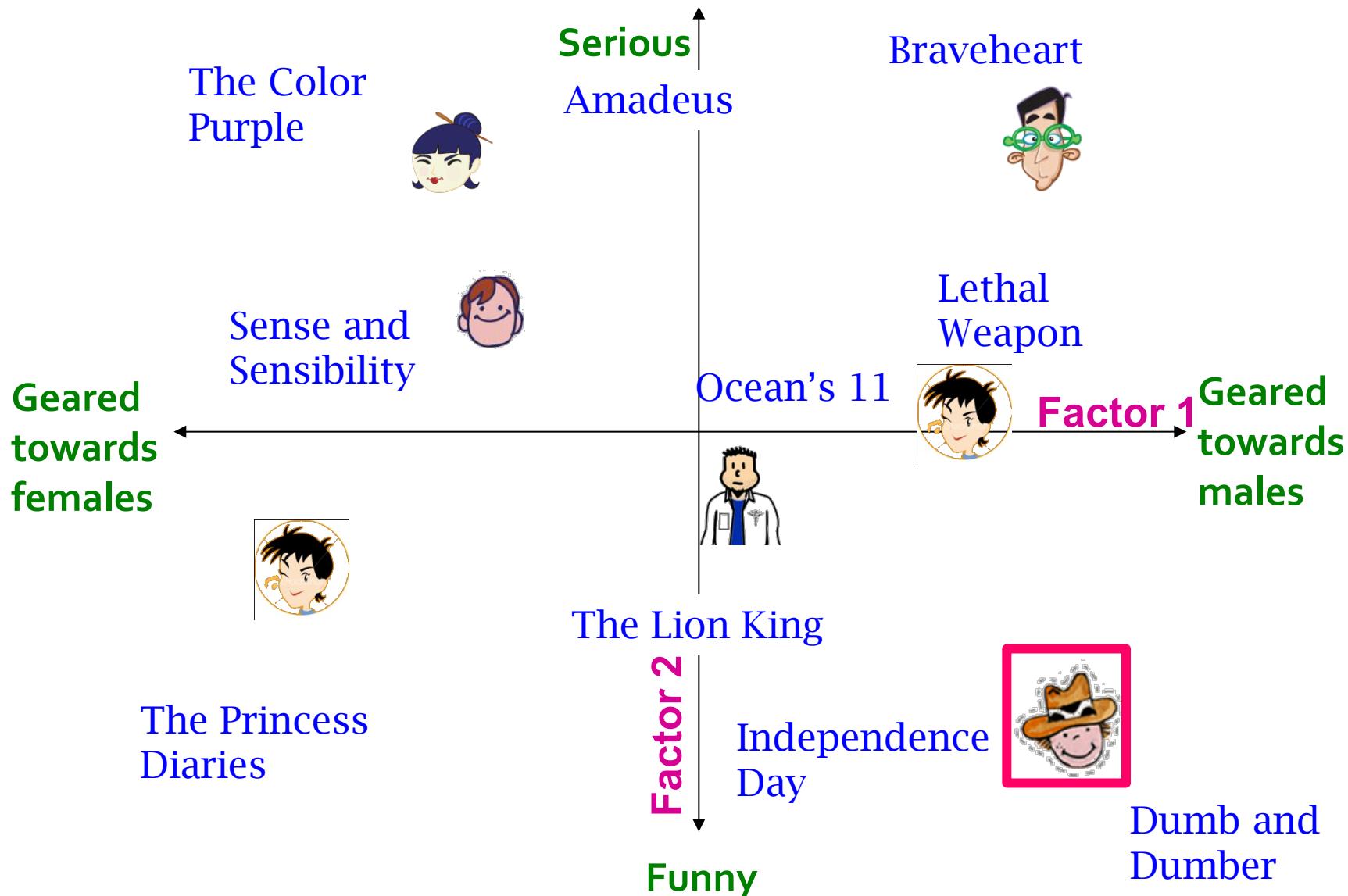


q_i = row i of Q
 p_x = column x of P^T

Latent Factor Models

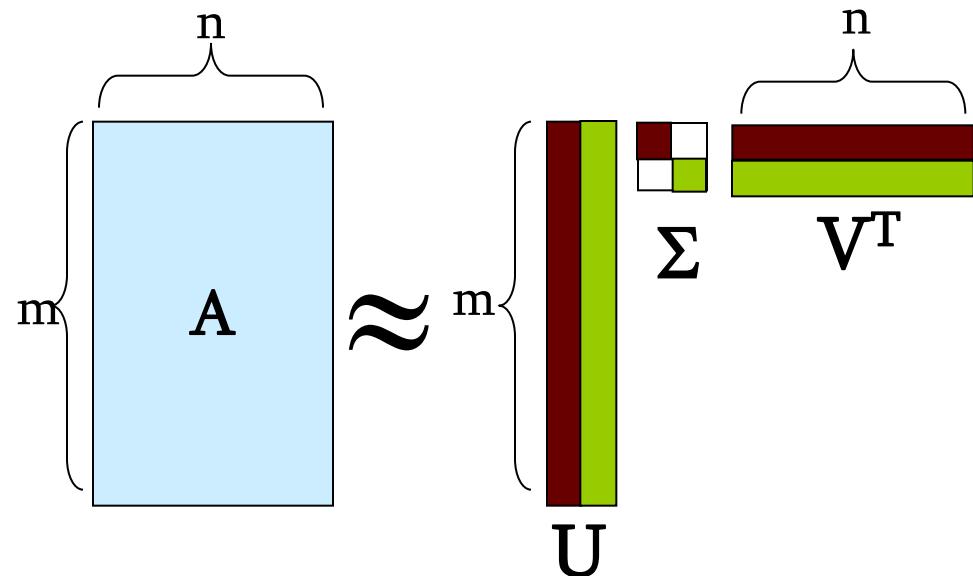


Latent Factor Models



Recap: SVD

- Remember SVD:
 - A : Input data matrix
 - U : Left singular vecs
 - V^T : Right singular vecs
 - Σ : Singular values



- So in our case:

“SVD” on Netflix data: $R \approx Q \cdot P^T$

$$A = R, \quad Q = U, \quad P^T = \Sigma V^T$$

$$\hat{r}_{xi} = q_i \cdot p_x^T$$

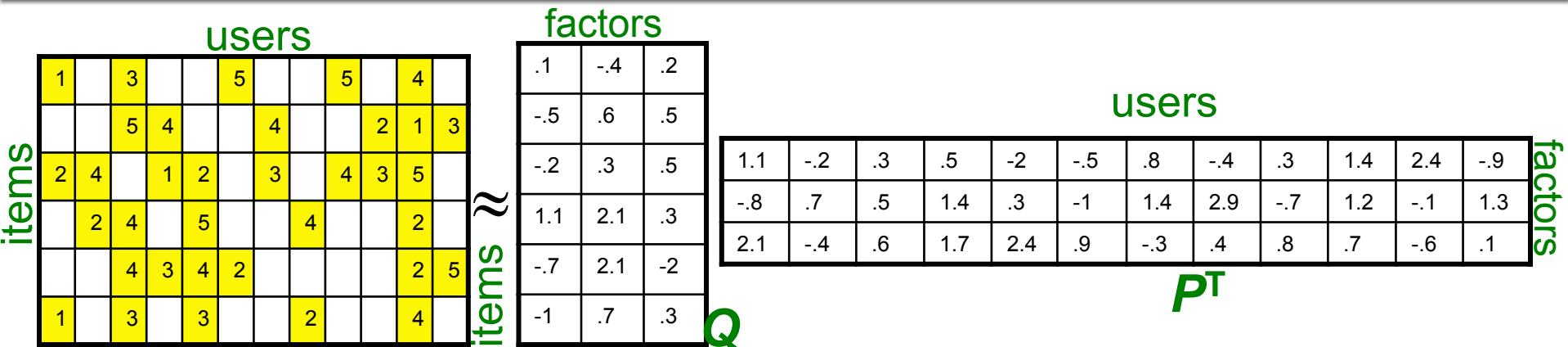
SVD: More good stuff

- We already know that SVD gives minimum reconstruction error (Sum of Squared Errors):

$$\min_{U, V, \Sigma} \sum_{ij \in A} (A_{ij} - [U\Sigma V^T]_{ij})^2$$

- Note two things:
 - SSE and RMSE are monotonically related:
 - $RMSE = \frac{1}{c} \sqrt{SSE}$ Great news: SVD is minimizing RMSE
 - Complication: The sum in SVD error term is over all entries (no-rating interpreted as zero-rating). But our R has missing entries!

Latent Factor Models



- SVD isn't defined when entries are missing!
- Use specialized methods to find P, Q :

$$\min_{P,Q} \sum_{(i,x) \in R} (r_{xi} - q_i \cdot p_x^T)^2$$

- Note:
 - We don't require cols of P, Q to be orthogonal/unit length
 - P, Q map users/movies to a latent space
 - The most popular model among Netflix contestants

SVD Example & Conclusion

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Case study: How to query?

- Q: Find users that like ‘Matrix’
- A: Map query into a ‘concept space’ – how?

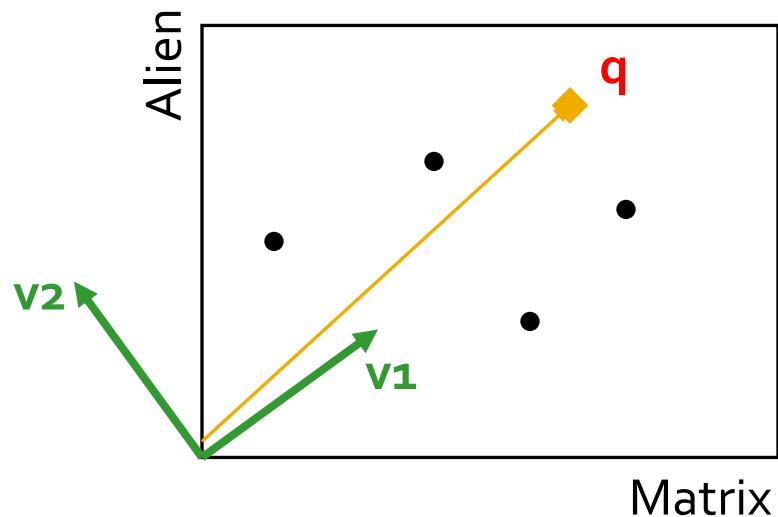
$$\begin{array}{c} \text{SciFi} \\ \uparrow \\ \text{User ID} \\ \downarrow \\ \text{Romance} \\ \uparrow \\ \text{Movie ID} \\ \downarrow \\ \text{Matrix} \quad \text{Alien} \quad \text{Serenity} \quad \text{Casablanca} \quad \text{Amelie} \end{array} \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} = \begin{bmatrix} 0.13 & 0.02 & -0.01 \\ 0.41 & 0.07 & -0.03 \\ 0.55 & 0.09 & -0.04 \\ 0.68 & 0.11 & -0.05 \\ 0.15 & -0.59 & 0.65 \\ 0.07 & -0.73 & -0.67 \\ 0.07 & -0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & -0.69 & -0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

Case study: How to query?

- Q: Find users that like ‘Matrix’
- A: Map query into a ‘concept space’ – how?

$$q = \begin{bmatrix} \text{Matrix} \\ \text{Alien} \\ \text{Serenity} \\ \text{Casablanca} \\ \text{Amelie} \end{bmatrix} \quad \begin{bmatrix} 5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Project into concept space:
Inner product with each
'concept' vector v_i

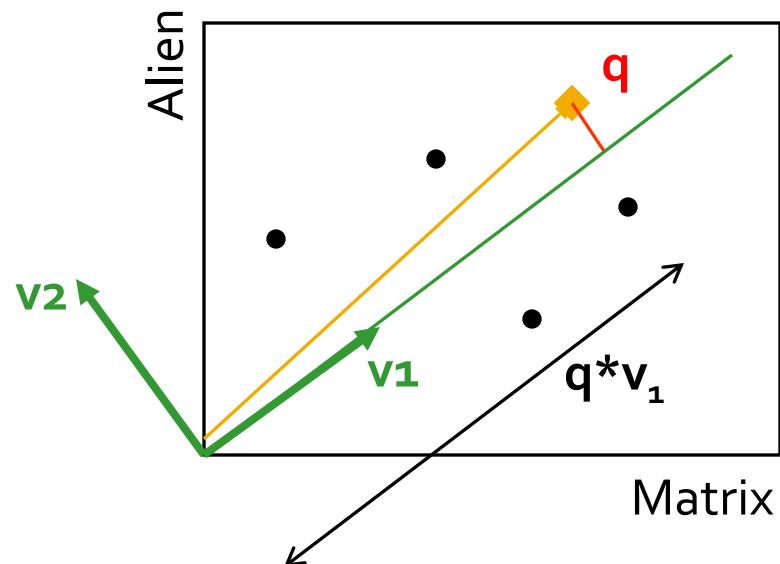


Case study: How to query?

- Q: Find users that like ‘Matrix’
- A: Map query into a ‘concept space’ – how?

$$q = \begin{bmatrix} \text{Matrix} \\ \text{Alien} \\ \text{Serenity} \\ \text{Casablanca} \\ \text{Amelie} \end{bmatrix} \quad \begin{bmatrix} 5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Project into concept space:
Inner product with each
'concept' vector v_i



Case study: How to query?

Compactly, we have:

$$q_{\text{concept}} = q \mathbf{V}$$

E.g.:

$$q = \begin{bmatrix} \text{Matrix} \\ \text{Alien} \\ \text{Serenity} \\ \text{Casablanca} \\ \text{Amelie} \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.12 \\ 0.59 & -0.02 \\ 0.56 & 0.12 \\ 0.09 & -0.69 \\ 0.09 & -0.69 \end{bmatrix} = \begin{bmatrix} 2.8 \\ 0.6 \end{bmatrix}$$

movie-to-concept
similarities (\mathbf{V})

SciFi-concept

Case study: How to query?

- How would the user d that rated ('Alien', 'Serenity') be handled?

$$d_{\text{concept}} = d V$$

E.g.:

$$d = \begin{bmatrix} \text{Matrix} \\ 0 & 4 & 5 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} \text{Alien} \\ \text{Serenity} \\ \text{Casablanca} \\ \text{Amelie} \end{bmatrix}$$

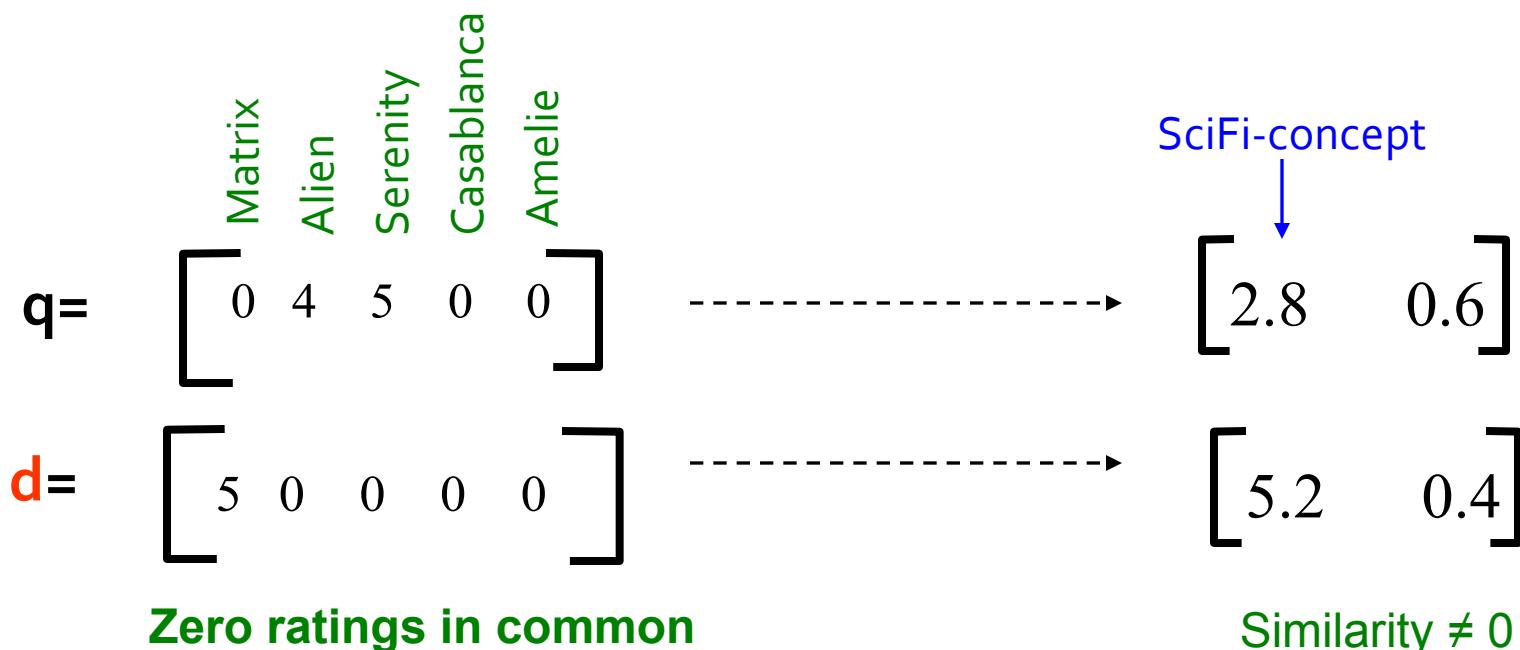
\times SciFi-concept

$$= \begin{bmatrix} 0.56 & 0.12 \\ 0.59 & -0.02 \\ 0.56 & 0.12 \\ 0.09 & -0.69 \\ 0.09 & -0.69 \end{bmatrix}$$

movie-to-concept
similarities (V)

Case study: How to query?

- **Observation:** User d that rated ('Alien', 'Serenity') will be **similar** to user q that rated ('Matrix'), although d and q have **zero ratings in common!**



Relation to Eigen-decomposition

- SVD gives us:

- $A = U \Sigma V^T$

- Eigen-decomposition:

- $A = X \Lambda X^T$

- A is symmetric

- U, V, X are orthonormal (e.g., $U^T U = I$),

- Λ , Σ are diagonal

- What is:

- $AA^T =$

- $A^T A = V \Sigma^T U^T (U \Sigma V^T) = V \Sigma \Sigma^T V^T$

Relation to Eigen-decomposition

- SVD gives us:

- $A = U \Sigma V^T$

- Eigen-decomposition:

- $A = X \Lambda X^T$

- A is symmetric

- U, V, X are orthonormal (e.g., $U^T U = I$),

- Λ , Σ are diagonal

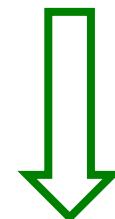
- What is:

- $AA^T = U\Sigma V^T (U\Sigma V^T)^T = U\Sigma V^T (V\Sigma^T U^T) = U\Sigma\Sigma^T U^T$

- $A^T A = V \Sigma^T U^T (U\Sigma V^T) = V \Sigma \Sigma^T V^T$

$$X \Lambda X^T$$

Shows how to compute SVD using eigenvalue decomposition!



$$X \Lambda X^T$$

$\downarrow \quad \downarrow \quad \downarrow$

$$U \Sigma \Sigma^T U^T$$

$\uparrow \quad \uparrow \quad \uparrow$

So, $\lambda_i = \sigma_i^2$

SVD: Drawbacks

- + Optimal low-rank approximation
in terms of Frobenius norm
- Interpretability problem:
 - A singular vector specifies a linear combination of all input columns or rows
- Lack of sparsity:
 - Singular vectors are **dense!**

$$\begin{matrix} \bullet & \bullet \\ \bullet & \bullet \\ \bullet & \\ \bullet & \bullet \end{matrix} = \begin{matrix} U \\ \Sigma \\ V^T \end{matrix}$$

The diagram illustrates the Singular Value Decomposition (SVD) of a 4x2 matrix. On the left, a 4x2 matrix is shown with black dots representing non-zero entries. An equals sign follows. To the right of the equals sign are three matrices: U, Sigma, and V^T. Matrix U is a 4x4 matrix where each column has exactly one black dot. Matrix Sigma is a 4x2 diagonal matrix with black dots on the diagonal. Matrix V^T is a 2x2 matrix where each row has exactly one black dot.

CUR: How it Works

- **Sampling columns (similarly for rows):**
- **ColumnSelect algorithm:**

Input: matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, sample size c

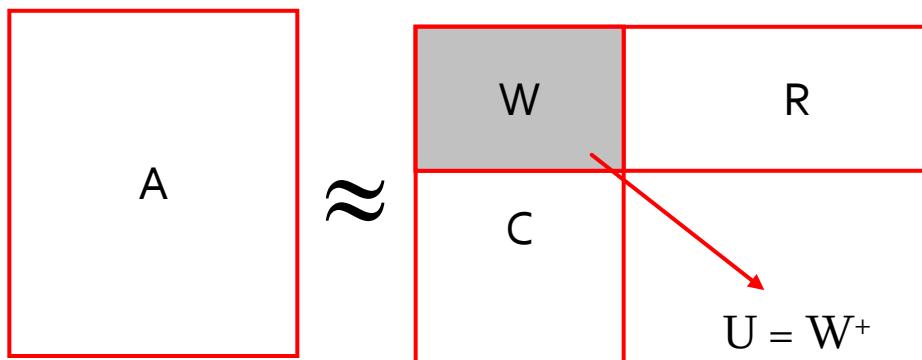
Output: $\mathbf{C}_d \in \mathbb{R}^{m \times c}$

1. for $x = 1 : n$ [column distribution]
2. $P(x) = \sum_i \mathbf{A}(i, x)^2 / \sum_{i,j} \mathbf{A}(i, j)^2$
3. for $i = 1 : c$ [sample columns]
4. Pick $j \in 1 : n$ based on distribution $P(x)$
5. Compute $\mathbf{C}_d(:, i) = \mathbf{A}(:, j) / \sqrt{cP(j)}$

Note this is a randomized algorithm, same column can be sampled more than once

Computing U

- Let \mathbf{W} be the “intersection” of sampled columns \mathbf{C} and rows \mathbf{R}
 - Let SVD of $\mathbf{W} = \mathbf{X} \mathbf{Z} \mathbf{Y}^T$
- Then: $\mathbf{U} = \mathbf{W}^+ = \mathbf{Y} \mathbf{Z}^+ \mathbf{X}^T$
 - \mathbf{Z}^+ : **reciprocals of non-zero singular values**: $Z_{ii}^+ = 1/Z_{ii}$
 - \mathbf{W}^+ is the “**pseudoinverse**”



Why pseudoinverse works?
 $\mathbf{W} = \mathbf{X} \mathbf{Z} \mathbf{Y}$ then $\mathbf{W}^{-1} = \mathbf{X}^{-1} \mathbf{Z}^{-1} \mathbf{Y}^{-1}$
Due to orthonormality
 $\mathbf{X}^{-1} = \mathbf{X}^T$ and $\mathbf{Y}^{-1} = \mathbf{Y}^T$
Since \mathbf{Z} is diagonal $\mathbf{Z}^{-1} = 1/Z_{ii}$
Thus, if \mathbf{W} is non-singular,
pseudoinverse is the true
inverse

CUR: Provably good approx. to SVD

- For example:

- Select $c = O\left(\frac{k \log k}{\epsilon^2}\right)$ columns of A using **ColumnSelect algorithm**
- Select $r = O\left(\frac{k \log k}{\epsilon^2}\right)$ rows of A using **ColumnSelect algorithm**
- Set $U = W^+$
- **Then:** $\|A - CUR\|_F \leq (2 + \epsilon) \|A - A_k\|_F$ with probability 98%

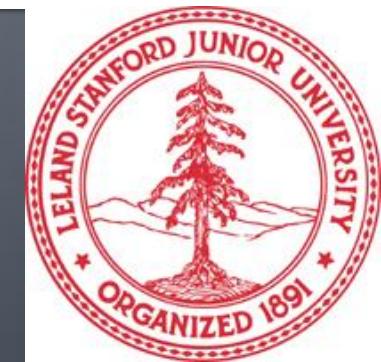
LSH Families of Hash Functions

Definition

Combining hash functions

Making steep S-Curves

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



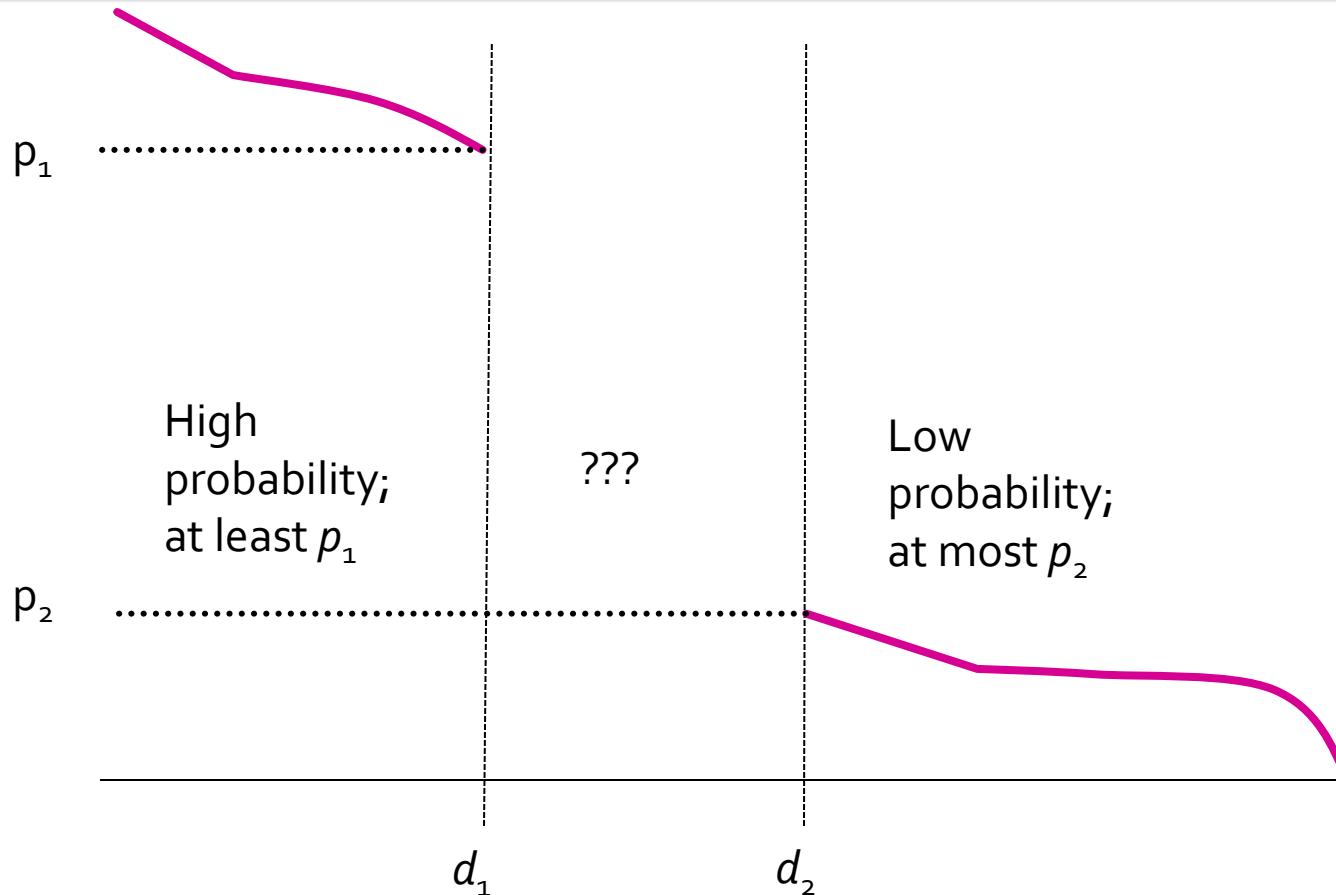
Hash Functions Decide Equality

- There is a subtlety about what a “hash function” really is in the context of LSH families.
- A hash function h really takes two elements x and y , and returns a decision whether x and y are candidates for comparison.
- **Example:** the family of minhash functions computes minhash values and says “yes” iff they are the same.
- **Shorthand:** “ $h(x) = h(y)$ ” means h says “yes” for pair of elements x and y .

LSH Families Defined

- Suppose we have a space S of points with a distance measure d .
- A family \mathbf{H} of hash functions is said to be **(d_1, d_2, p_1, p_2) -sensitive** if for any x and y in S :
 1. If $d(x, y) \leq d_1$, then the probability over all h in \mathbf{H} , that $h(x) = h(y)$ is at least p_1 .
 2. If $d(x, y) \geq d_2$, then the probability over all h in \mathbf{H} , that $h(x) = h(y)$ is at most p_2 .

LS Families: Illustration



Example: LS Family

- Let S = sets, d = Jaccard distance, \mathbf{H} is formed from the minhash functions for all permutations.
- Then $\text{Prob}[h(x)=h(y)] = 1-d(x,y)$.
 - Restates theorem about Jaccard similarity and minhashing in terms of Jaccard distance.

Example: LS Family – (2)

- **Claim:** \mathbf{H} is a $(1/3, 2/3, 2/3, 1/3)$ -sensitive family for S and d .

If distance $\leq 1/3$
(so similarity $\geq 2/3$)

Then probability
that minhash values
agree is $\geq 2/3$

For Jaccard similarity, minhashing gives us a
 $(d_1, d_2, (1-d_1), (1-d_2))$ -sensitive family for any $d_1 < d_2$.

Amplifying a LSH-Family

- The “bands” technique we learned for signature matrices carries over to this more general setting.
 - **Goal:** the “S-curve” effect seen there.
- AND construction like “rows in a band.”
- OR construction like “many bands.”

AND of Hash Functions

- Given family \mathbf{H} , construct family \mathbf{H}' whose members each consist of r functions from \mathbf{H} .
- For $h = \{h_1, \dots, h_r\}$ in \mathbf{H}' , $h(x) = h(y)$ if and only if $h_i(x) = h_i(y)$ for all i .
- **Theorem:** If \mathbf{H} is (d_1, d_2, p_1, p_2) -sensitive, then \mathbf{H}' is $(d_1, d_2, (p_1)^r, (p_2)^r)$ -sensitive.
 - **Proof:** Use fact that h_i 's are independent.

OR of Hash Functions

- Given family \mathbf{H} , construct family \mathbf{H}' whose members each consist of b functions from \mathbf{H} .
- For $h = \{h_1, \dots, h_b\}$ in \mathbf{H}' , $h(x)=h(y)$ if and only if $h_i(x)=h_i(y)$ for **some** i .
- **Theorem:** If \mathbf{H} is (d_1, d_2, p_1, p_2) -sensitive, then \mathbf{H}' is $(d_1, d_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$ -sensitive.

Effect of AND and OR Constructions

- AND makes all probabilities shrink, but by choosing r correctly, we can make the lower probability approach 0 while the higher does not.
- OR makes all probabilities grow, but by choosing b correctly, we can make the upper probability approach 1 while the lower does not.

Composing Constructions

- As for the signature matrix, we can use the AND construction followed by the OR construction.
 - Or vice-versa.
 - Or any sequence of AND's and OR's alternating.

AND-OR Composition

- Each of the two probabilities p is transformed into $1-(1-p^r)^b$.
 - The “S-curve” studied before.
- **Example:** Take \mathbf{H} and construct \mathbf{H}' by the AND construction with $r = 4$. Then, from \mathbf{H}' , construct \mathbf{H}'' by the OR construction with $b = 4$.

Table for Function $1-(1-p^4)^4$

p	$1-(1-p^4)^4$
.2	.0064
.3	.0320
.4	.0985
.5	.2275
.6	.4260
.7	.6666
.8	.8785
.9	.9860

Example: Transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.8785,.0064)-sensitive family.

OR-AND Composition

- Each of the two probabilities p is transformed into $(1-(1-p)^b)^r$.
 - The same S-curve, mirrored horizontally and vertically.
- Example: Take \mathbf{H} and construct \mathbf{H}' by the OR construction with $b = 4$. Then, from \mathbf{H}' , construct \mathbf{H}'' by the AND construction with $r = 4$.

Table for Function $(1-(1-p)^4)^4$

p	$(1-(1-p)^4)^4$
.1	.0140
.2	.1215
.3	.3334
.4	.5740
.5	.7725
.6	.9015
.7	.9680
.8	.9936

Example: Transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.9936,.1215)-sensitive family.

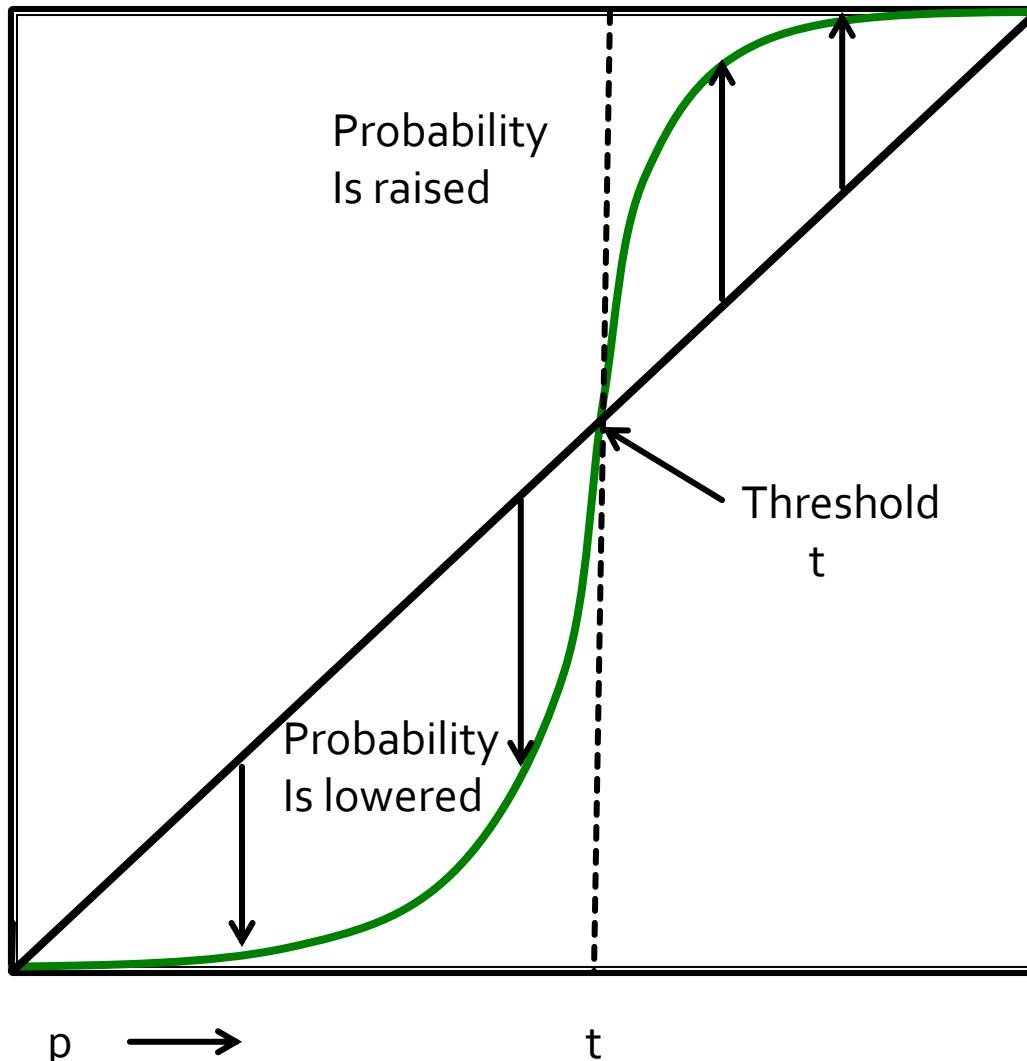
Cascading Constructions

- Example: Apply the (4,4) OR-AND construction followed by the (4,4) AND-OR construction.
- Transforms a (.2,.8,.8,.2)-sensitive family into a (.2,.8,.9999996,.0008715)-sensitive family.

General Use of S-Curves

- For each S-curve $1-(1-p^r)^b$, there is a *threshold* t , for which $1-(1-t^r)^b = t$.
- Above t , high probabilities are increased; below t , they are decreased.
- You improve the sensitivity as long as the low probability is less than t , and the high probability is greater than t .
 - Iterate as you like.

Visualization of Threshold



More LSH Families

Cosine Distance and Random
Hyperplanes
Euclidean Distance

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



An LSH Family for Cosine Distance

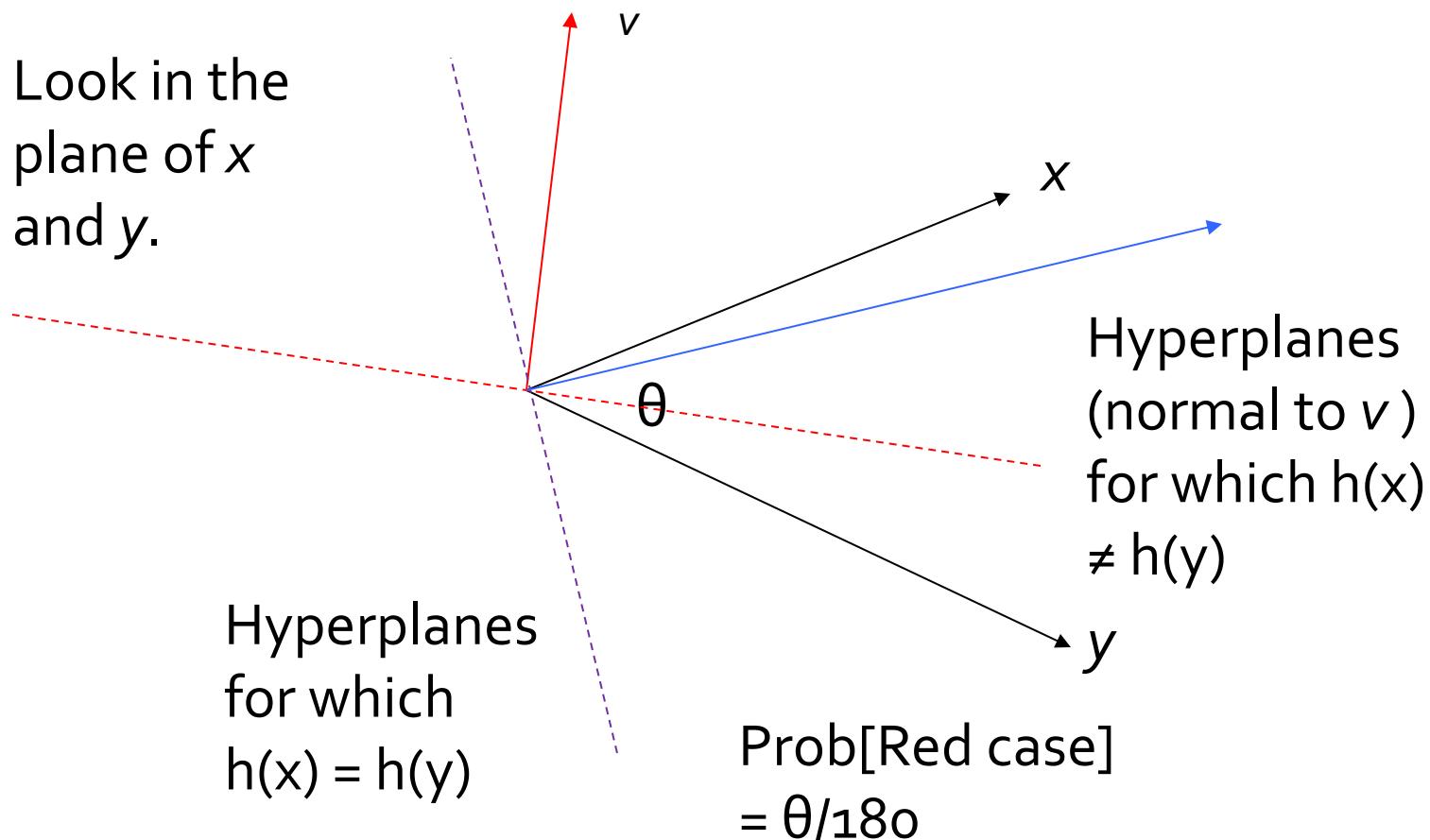
- For cosine distance, there is a technique analogous to minhashing for generating a $(d_1, d_2, (1-d_1/180), (1-d_2/180))$ -sensitive family for any d_1 and d_2 .
- Called *random hyperplanes*.

Random Hyperplanes

- Each vector v determines a hash function h_v with two buckets.
- $h_v(x) = +1$ if $v \cdot x > 0$; $= -1$ if $v \cdot x < 0$.
- LS-family \mathbf{H} = set of all functions derived from any vector.
- **Claim:** $\text{Prob}[h(x)=h(y)] = 1 - (\text{angle between } x \text{ and } y \text{ divided by } 180)$.

Proof of Claim

Look in the plane of x and y .



Signatures for Cosine Distance

- Pick some number of vectors, and hash your data for each vector.
- The result is a signature (*sketch*) of +1's and -1's that can be used for LSH like the minhash signatures for Jaccard distance.
- But you don't have to think this way.
- The existence of the LSH-family is sufficient for amplification by AND/OR.

Simplification

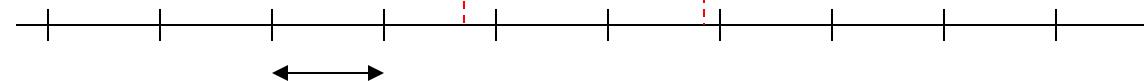
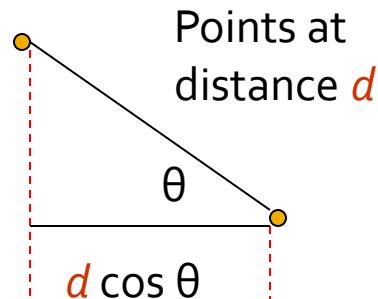
- We need not pick from among all possible vectors v to form a component of a sketch.
- It suffices to consider only vectors v consisting of +1 and -1 components.

LSH for Euclidean Distance

- Simple idea: hash functions correspond to lines.
- Partition the line into buckets of size a .
- Hash each point to the bucket containing its projection onto the line.
- Nearby points are always close; distant points are rarely in same bucket.

Projection of Points

If $d \gg a$, θ must be close to 90° for there to be any chance points go to the same bucket.



Bucket
width a

If $d \ll a$, then the chance the points are in the same bucket is at least $1 - d/a$.

Randomly
chosen
line

An LS-Family for Euclidean Distance

- If points are distance $\geq 2a$ apart, then $60 \leq \theta \leq 90$ for there to be a chance that the points go in the same bucket.
 - I.e., at most $1/3$ probability.
- If points are distance $\leq a/2$, then there is at least $\frac{1}{2}$ chance they share a bucket.
- Yields a $(a/2, 2a, 1/2, 1/3)$ -sensitive family of hash functions.

Fixup: Euclidean Distance

- For previous distance measures, we could start with a (d, e, p, q) -sensitive family for any $d < e$, and drive p and q to 1 and 0 by AND/OR constructions.
- Here, we seem to need $e \geq 4d$.

Fixup – (2)

- But as long as $d < e$, the probability of points at distance d falling in the same bucket is greater than the probability of points at distance e doing so.
- Thus, the hash family formed by projecting onto lines is a (d, e, p, q) -sensitive family for **some** $p > q$.

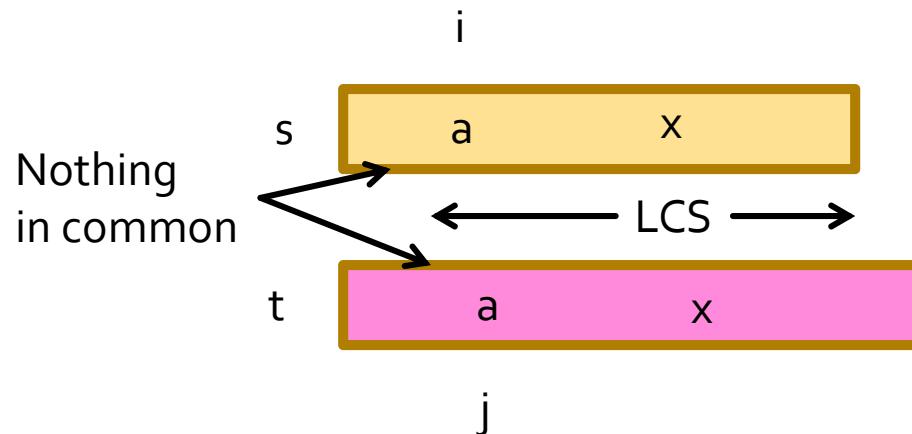
Positions Within Prefixes

Positions in the Probe and Target
Strings

Bounding the Edit Distance
Two-Dimensional Indexes

Exploiting the Position

- If position i of probe string s is the first position to match a prefix position of string t , and it matches position j , then the edit distance between s and t is at least $i + j - 2$.



- The LCS of s and t is no longer than $L-i+1$, where L is the length of s .

Positions/Prefixes – (2)

- If J is the limit on Jaccard distance, then remember $E/(E+C) \leq J$.
 - $E \geq i + j - 2$.
 - $C \leq L - i + 1$.
- Thus, $(i + j - 2)/(L + j - 1) \leq J$.
- Or, $j \leq (JL - J - i + 2)/(1 - J)$.

Positions/Prefixes – Indexing

- Create a 2-attribute index on (symbol, position).
- If string s has symbol a as the i^{th} position of its prefix, add s to the bucket (a, i) .
- A B-tree index with keys ordered first by symbol, then position is excellent.

Positions/Prefixes – (3)

- Given probe string s , we only need to find a candidate once, so we may as well:
 1. Visit positions i of s in numerical order, and
 2. Assume that there have been no matches for earlier positions.
 - That lets us use the upper bound on j when deciding what index buckets we need to look in.

Lookup

- If we want to find matches for probe string s of length L , do:

```
for (i=1; i<=J*L+1; i++) {  
    let s have a in position i;  
    for (j=1;  
        j<= (J*L-J-i+2) / (1-J) ; j++)  
        compare s with strings in  
        bucket (a, j);  
}
```

Example: Lookup

- Suppose $J = 0.2$.
- Given probe string **a_de_gj_kkmprz**, $L=10$, and the prefix is **ade**.
- For the i^{th} position of the prefix, we must look at buckets where
$$j \leq (JL - J - i + 2) / (1 - J) = (3.8 - i) / 0.8.$$
- For $i = 1$: $j \leq 3$; for $i = 2$: $j \leq 2$, and for $i = 3$: $j \leq 1$.

Example: Lookup – (2)

- Thus, for probe $s = \text{adegjkmprz}$ we look in the following buckets: $(a, 1), (a, 2), (a, 3), (d, 1), (d, 2), (e, 1)$.
- Suppose string t is in none of these buckets.
- Then the edit distance E is at least 3.
 - Why? Consider where the first common symbol between s and t could be within t .
- The LCS C cannot be longer than s , i.e., 10.
- Thus, $J \geq 3/13 > 0.2$.

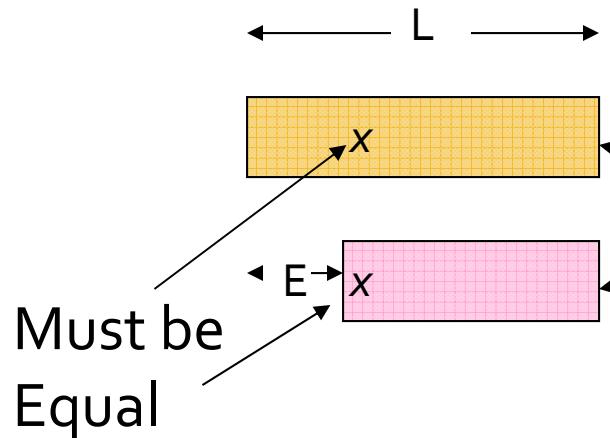
The Prefix of a String

Indexing by Symbols
Prefixes

Example: Prefix-Based Indexing

- If two strings are 90% similar, they must share some symbol in their prefixes whose length is just above 10% of the length of each string.
- Thus, we can base an index on symbols in just the first $\lfloor JL+1 \rfloor$ positions of a string of length L.
 - That's the *prefix* of the string.

Why the Limit on Prefixes?



Extreme case: second string has none of the first E symbols of the first string, but they agree thereafter.

If two strings do not share any of the first E symbols, then $J \geq E/L$.

Thus, $E = JL$ is possible, but any larger E is impossible. Index $E+1$ positions.

Indexing Prefixes

- Think of a bucket for each possible symbol.
- Each string of length L is placed in the bucket for **each** of its first $[JL+1]$ positions.
- A B-tree with symbol as key leads to the strings.

Lookup

- Given a *probe* string s of length L , with J the limit on Jaccard distance:

```
for (each symbol  $a$  among the  
first  $\lfloor JL+1 \rfloor$  positions of  $s$ )  
    look for other strings in  
    the bucket for  $a$ ;
```

Example: Indexing Prefixes

- Let $J = 0.2$.
- String **abcdef** is indexed under *a* and *b*.
- String **acdfg** is indexed under *a* and *c*.
- String **bcde** is indexed only under *b*.
- If we search for strings similar to **cdef**, we need look only in the bucket for *c*.

Methods for High Degrees of Similarity

Index-Based Methods
Exploiting Prefixes and Suffixes
Exploiting Length

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Setting: Sets as Strings

- We'll again talk about Jaccard similarity and distance of sets.
- However, now represent sets by strings (lists of symbols):
 1. Order the universal set.
 2. Represent a set by the string of its elements in sorted order.

Example: Shingles

- If the universal set is k-shingles, there is a natural lexicographic order.
- Think of each shingle as a single symbol.
- Then the 2-shingling of **abcad**, which is the set {ab, bc, ca, ad}, is represented by the list [ab, ad, bc, ca] of length 4.

Example: Words

- If we treat a document as a set of words, we could order the words lexicographically.
- **Better:** Order words lowest-frequency-first.
- **Why?** We shall index documents based on the early words in their lists.
 - Documents spread over more buckets.

Jaccard and Edit Distances

- Suppose two sets have Jaccard distance J and are represented by strings s_1 and s_2 . Let the LCS of s_1 and s_2 have length C and the (insert/delete) edit distance of s_1 and s_2 be E .
Then:
 - $1-J = \text{Jaccard similarity} = C/(C+E)$.
 - $J = E/(C+E)$.

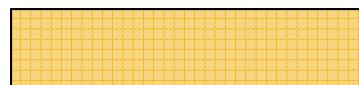
Works because these strings never repeat a symbol, and symbols appear in the same order.

Length-Based Indexes

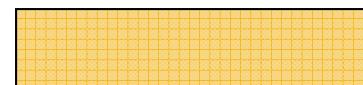
- The simplest thing to do is create an index on the length of strings.
- A set whose string has length L can be Jaccard distance J from a set whose string has length M only if $L \times (1-J) \leq M \leq L / (1-J)$.
- **Example:** if $1-J = 90\%$ (Jaccard similarity), then M is between 90% and 111% of L .

Why the Limit on Lengths?

$$\longleftrightarrow L \longrightarrow$$



$$\longleftrightarrow L \longrightarrow$$



$$\longleftrightarrow M \longrightarrow$$

$$1-J = M/L$$

$$M = L \times (1-J)$$

A shortest candidate

$$\longleftrightarrow M \longrightarrow$$

$$1-J = L/M$$

$$M = L/(1-J)$$

A longest candidate

B-Tree Indexes

- The B-tree is a perfect index structure for a length-based index.
- Given a string of length L , we can find strings with length in the range $L \times (1-J)$ to $L/(1-J)$ without looking at any candidates outside that range.
- But just because strings are similar in length, doesn't mean they are similar.

Extensions: Topic Specific (aka Personalized) PageRank

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



Topic-Specific PageRank

- Instead of generic popularity, can we measure popularity within a topic?
- Goal: Evaluate Web pages not just according to their popularity, but by how close they are to a particular topic, e.g. “sports” or “history”
- Allows search queries to be answered based on interests of the user
 - Example: Query “Trojan” wants different pages depending on whether you are interested in sports, history and computer security

Topic-Specific PageRank

- Random walker has a small probability of teleporting at any step
- **Teleport can go to:**
 - **Standard PageRank:** Any page with equal probability
 - To avoid dead-end and spider-trap problems
 - **Topic Specific PageRank:** A topic-specific set of “relevant” pages (**teleport set**)
- **Idea: Bias the random walk**
 - When walker teleports, she pick a page from a set S
 - S contains only pages that are relevant to the topic
 - E.g., Open Directory (DMOZ) pages for a given topic/query
 - For each teleport set S , we get a different vector r_s

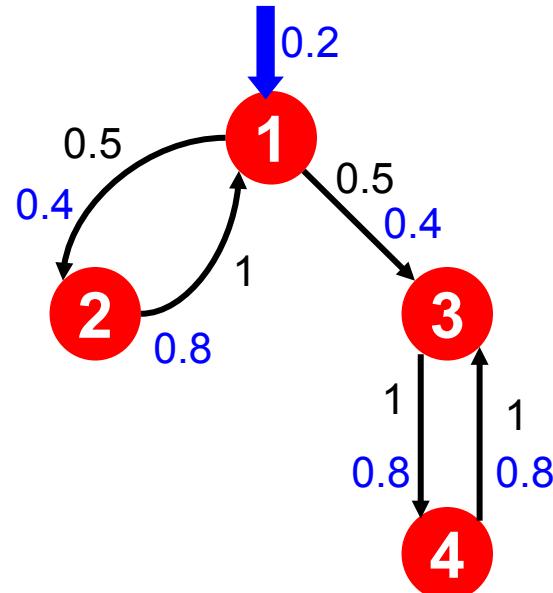
Matrix Formulation

- To make this work all we need is to update the teleportation part of the PageRank formulation:

$$A_{ij} = \begin{cases} \beta M_{ij} + (1 - \beta)/|S| & \text{if } i \in S \\ \beta M_{ij} & \text{otherwise} \end{cases}$$

- A is stochastic!
- We weighted all pages in the teleport set S equally
 - Could also assign different weights to pages!
- Random Walk with Restart: S is a single element
- Compute as for regular PageRank:
 - Multiply by M , then add a vector
 - Maintains sparseness

Example: Topic-Specific PageRank



Suppose $S = \{1\}$, $\beta = 0.8$

Node	Iteration			
	0	1	2	stable
1	0.25	0.4	0.28	0.294
2	0.25	0.1	0.16	0.118
3	0.25	0.3	0.32	0.327
4	0.25	0.2	0.24	0.261

$S=\{1,2,3,4\}$, $\beta=0.8$:

$r=[0.13, 0.10, 0.39, 0.36]$

$S=\{1,2,3\}$, $\beta=0.8$:

$r=[0.17, 0.13, 0.38, 0.30]$

$S=\{1\}$, $\beta=0.8$:

$r=[0.26, 0.20, 0.29, 0.23]$

$S=\{1\}$, $\beta=0.70$:

$r=[0.29, 0.11, 0.32, 0.26]$

$r=[0.39, 0.14, 0.27, 0.19]$

$r=[0.29, 0.11, 0.32, 0.26]$

Discovering the Topic Vector S

- **Create different PageRanks for different topics**
 - The 16 DMOZ top-level categories:
 - arts, business, sports,...
- **Which topic ranking to use?**
 - User can pick from a menu
 - Classify query into a topic
 - Can use the **context** of the query
 - E.g., query is launched from a web page talking about a known topic
 - History of queries e.g., “basketball” followed by “Jordan”
 - User context, e.g., user’s bookmarks, ...

Web Spam

Mining of Massive Datasets
Leskovec, Rajaraman, and Ullman
Stanford University



What is Web Spam?

- **Spamming:**
 - Any deliberate action to boost a web page's position in search engine results, incommensurate with page's real value
- **Spam:**
 - Web pages that are the result of spamming
- This is a very broad definition
 - SEO industry might disagree!
 - SEO = search engine optimization
- Approximately **10-15%** of web pages are spam

Web Search

- **Early search engines:**
 - Crawl the Web
 - Index pages by the words they contained
 - Respond to search queries (lists of words) with the pages containing those words
- **Early page ranking:**
 - Attempt to order pages matching a search query by “importance”
 - **First search engines considered:**
 - (1) Number of times query words appeared
 - (2) Prominence of word position, e.g. title, header

First Spammers

- As people began to use search engines to find things on the Web, those with commercial interests tried to **exploit search engines** to bring people to their own site – whether they wanted to be there or not
- **Example:**
 - Shirt-seller might pretend to be about “movies”
- **Techniques for achieving high relevance/importance for a web page**

First Spammers: Term Spam

- **How do you make your page appear to be about movies?**
 - (1) Add the word “movie” 1,000 times to your page
 - Set text color to the background color, so only search engines would see it
 - (2) Or, run the query “movie” on your target search engine
 - See what page came first in the listings
 - Copy it into your page, make it “invisible”
- **These and similar techniques are term spam**

Google's Solution to Term Spam

- Believe what people say about you, rather than what you say about yourself
 - Use words in the anchor text (words that appear underlined to represent the link) and its surrounding text
- PageRank as a tool to measure the “importance” of Web pages

Why It Works?

- **Our hypothetical shirt-seller loses**
 - Saying he is about movies doesn't help, because others don't say he is about movies
 - His page isn't very important, so it won't be ranked high for shirts or movies
- **Example:**
 - Shirt-seller creates 1,000 pages, each links to his with "movie" in the anchor text
 - These pages have no links in, so they get little PageRank
 - So the shirt-seller can't beat truly important movie pages, like IMDB

Why it does not work?



Web

Results 1 - 10 of about 969,000 for [miserable failure](#). (0.06 seconds)

[Biography of President George W. Bush](#)

Biography of the president from the official White House web site.

www.whitehouse.gov/president/gwbbio.html - 29k - [Cached](#) - [Similar pages](#)

[Past Presidents](#) - [Kids Only](#) - [Current News](#) - [President](#)

[More results from www.whitehouse.gov »](#)

[Welcome to MichaelMoore.com!](#)

Official site of the gadfly of corporations, creator of the film Roger and Me
and the television show The Awful Truth. Includes mailing list, message board, ...
www.michaelmoore.com/ - 35k - Sep 1, 2005 - [Cached](#) - [Similar pages](#)

[BBC NEWS | Americas | 'Miserable failure' links to Bush](#)

Web users manipulate a popular search engine so an unflattering description leads
to the president's page.
news.bbc.co.uk/2/hi/americas/3298443.stm - 31k - [Cached](#) - [Similar pages](#)

[Google's \(and Inktomi's\) Miserable Failure](#)

A search for **miserable failure** on Google brings up the official George W.
Bush biography from the US White House web site. Dismissed by Google as not a ...
searchenginewatch.com/sereport/article.php/3296101 - 45k - Sep 1, 2005 - [Cached](#) - [Similar pages](#)

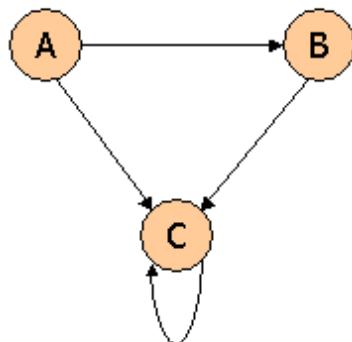
Feedback — Week 1

[Help](#)

You submitted this quiz on **Thu 2 Oct 2014 11:57 AM PDT**. You got a score of **4.00** out of **4.00**.

Question 1

Consider three Web pages with the following links:

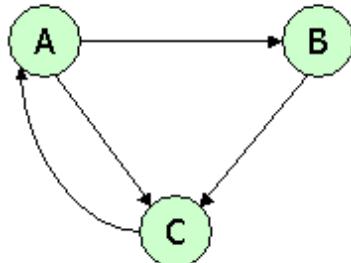


Suppose we compute PageRank with a β of 0.7, and we introduce the additional constraint that the sum of the PageRanks of the three pages must be 3, to handle the problem that otherwise any multiple of a solution will also be a solution. Compute the PageRanks a , b , and c of the three pages A, B, and C, respectively. Then, identify from the list below, the true statement.

Your Answer	Score	Explanation
<input checked="" type="radio"/> b + c = 2.7	✓ 1.00	
<input type="radio"/> a + b = 1.025		
<input type="radio"/> a + c = 2.745		
<input type="radio"/> b + c = 3.25		
Total	1.00 / 1.00	

Question 2

Consider three Web pages with the following links:

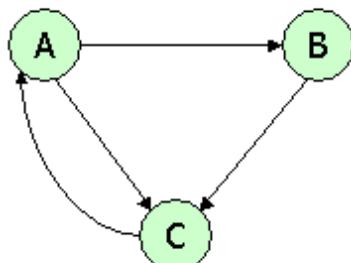


Suppose we compute PageRank with $\beta=0.85$. Write the equations for the PageRanks a , b , and c of the three pages A, B, and C, respectively. Then, identify in the list below, one of the equations.

Your Answer	Score	Explanation
<input type="radio"/> $b = .575a + .15c$		
<input type="radio"/> $.85c = b + .575a$		
<input checked="" type="radio"/> $.95c = .9b + .475a$	✓ 1.00	
<input type="radio"/> $b = .475a + .05c$		
Total	1.00 / 1.00	

Question 3

Consider three Web pages with the following links:



Assuming no "taxation," compute the PageRanks a , b , and c of the three pages A, B, and C,

using iteration, starting with the "0th" iteration where all three pages have rank $a = b = c = 1$. Compute as far as the 5th iteration, and also determine what the PageRanks are in the limit. Then, identify the true statement from the list below.

Your Answer	Score	Explanation
<input type="radio"/> In the limit, $c = 4/3$		
<input type="radio"/> After iteration 4, $b = 3/5$		
<input type="radio"/> After iteration 5, $a = 21/16$		
<input checked="" type="radio"/> After iteration 4, $c = 5/4$	✓ 1.00	
Total	1.00 / 1.00	

Question 4

Suppose our input data to a map-reduce operation consists of integer values (the keys are not important). The map function takes an integer i and produces the list of pairs (p,i) such that p is a prime divisor of i . For example, $\text{map}(12) = [(2,12), (3,12)]$.

The reduce function is addition. That is, $\text{reduce}(p, [i_1, i_2, \dots, i_k])$ is $(p, i_1 + i_2 + \dots + i_k)$.

Compute the output, if the input is the set of integers 15, 21, 24, 30, 49. Then, identify, in the list below, one of the pairs in the output.

Your Answer	Score	Explanation
<input type="radio"/> (5,49)		
<input checked="" type="radio"/> (7,70)	✓ 1.00	
<input type="radio"/> (6,54)		
<input type="radio"/> (7,119)		
Total	1.00 / 1.00	

Feedback — Week 2: Frequent Itemsets (Advanced)

[Help](#)

You submitted this quiz on **Sun 19 Oct 2014 12:35 PM PDT**. You got a score of **2.00** out of **2.00**.

Question 1

Suppose we perform the PCY algorithm to find frequent pairs, with market-basket data meeting the following specifications:

- s , the support threshold, is 10,000.
- There are one million items, which are represented by the integers 0, 1, ..., 999999.
- There are 250,000 frequent items, that is, items that occur 10,000 times or more.
- There are one million pairs that occur 10,000 times or more.
- There are P pairs that occur exactly once and consist of 2 frequent items.
- No other pairs occur at all.
- Integers are always represented by 4 bytes.
- When we hash pairs, they distribute among buckets randomly, but as evenly as possible; i.e., you may assume that each bucket gets exactly its fair share of the P pairs that occur once.

Suppose there are S bytes of main memory. In order to run the PCY algorithm successfully, the number of buckets must be sufficiently large that most buckets are not large. In addition, on the second pass, there must be enough room to count all the candidate pairs. As a function of S , what is the largest value of P for which we can successfully run the PCY algorithm on this data? Demonstrate that you have the correct formula by indicating which of the following is a value for S and a value for P that is approximately (i.e., to within 10%) the largest possible value of P for that S .

Your Answer	Score	Explanation
<input type="radio"/> S = 1,000,000,000; P = 10,000,000,000		
<input type="radio"/> S = 500,000,000; P = 10,000,000,000		
<input checked="" type="radio"/> S = 1,000,000,000; P = 20,000,000,000	✓ 1.00	
<input type="radio"/> S = 200,000,000; P = 400,000,000		
Total	1.00 / 1.00	

Question 2

During a run of Toivonen's Algorithm with set of items {A,B,C,D,E,F,G,H} a sample is found to have the following maximal frequent itemsets: {A,B}, {A,C}, {A,D}, {B,C}, {E}, {F}. Compute the negative border. Then, identify in the list below the set that is NOT in the negative border.

Your Answer Score Explanation



{C,D}



{B,D}



{H}

 {D}

1.00 Correct! This set is not in the negative border because it is itself frequent. We know it is frequent because it is a subset of maximal frequent itemset {A,D}.

Total

1.00 /
1.00

Feedback — Week 2: Frequent Itemsets (Advanced)

[Help](#)

You submitted this quiz on **Sun 19 Oct 2014 11:47 AM PDT**. You got a score of **1.00** out of **2.00**. You can [attempt again](#), if you'd like.

Question 1

Suppose we perform the PCY algorithm to find frequent pairs, with market-basket data meeting the following specifications:

- s , the support threshold, is 10,000.
- There are one million items, which are represented by the integers 0,1,...,999999.
- There are 250,000 frequent items, that is, items that occur 10,000 times or more.
- There are one million pairs that occur 10,000 times or more.
- There are P pairs that occur exactly once and consist of 2 frequent items.
- No other pairs occur at all.
- Integers are always represented by 4 bytes.
- When we hash pairs, they distribute among buckets randomly, but as evenly as possible; i.e., you may assume that each bucket gets exactly its fair share of the P pairs that occur once.

Suppose there are S bytes of main memory. In order to run the PCY algorithm successfully, the number of buckets must be sufficiently large that most buckets are not large. In addition, on the second pass, there must be enough room to count all the candidate pairs. As a function of S , what is the largest value of P for which we can successfully run the PCY algorithm on this data? Demonstrate that you have the correct formula by indicating which of the following is a value for S and a value for P that is approximately (i.e., to within 10%) the largest possible value of P for that S .

Your Answer**Score****Explanation**

S =
200,000,000; P
=
1,600,000,000

S = ✗ 0.00
1,000,000,000;
P =
10,000,000,000

Here are some hints:

1. The number of infrequent pairs per bucket on the first pass will be about P divided by the number of buckets.
2. A pair can only be a candidate pair for the second pass if it is in a frequent bucket. For the values of P and S found in this

question, that can only occur if the bucket contains one of the 1,000,000 frequent pairs.

3. You must use a hash table to count candidate pairs on the second pass of PCY. This hash table takes 12 bytes per candidate pair.

S =
300,000,000; P
=
1,800,000,000

S =
300,000,000; P
=
3,500,000,000

Total 0.00 /
 1.00

Question 2

During a run of Toivonen's Algorithm with set of items {A,B,C,D,E,F,G,H} a sample is found to have the following maximal frequent itemsets: {A,B}, {A,C}, {A,D}, {B,C}, {E}, {F}. Compute the negative border. Then, identify in the list below the set that is NOT in the negative border.

Your Answer	Score	Explanation
-------------	-------	-------------

{A,B,C}

{C,D}

{D,F}

{G,H}  1.00 Correct! This set is not in the negative border because immediate proper subset {G} is not frequent.

Total 1.00 /
 1.00

Feedback — Week 2: Frequent Itemsets (Advanced)

[Help](#)

You submitted this quiz on **Sun 19 Oct 2014 11:57 AM PDT**. You got a score of **0.00** out of **2.00**. You can [attempt again](#), if you'd like.

Question 1

Suppose we perform the PCY algorithm to find frequent pairs, with market-basket data meeting the following specifications:

- s , the support threshold, is 10,000.
- There are one million items, which are represented by the integers 0, 1, ..., 999999.
- There are 250,000 frequent items, that is, items that occur 10,000 times or more.
- There are one million pairs that occur 10,000 times or more.
- There are P pairs that occur exactly once and consist of 2 frequent items.
- No other pairs occur at all.
- Integers are always represented by 4 bytes.
- When we hash pairs, they distribute among buckets randomly, but as evenly as possible; i.e., you may assume that each bucket gets exactly its fair share of the P pairs that occur once.

Suppose there are S bytes of main memory. In order to run the PCY algorithm successfully, the number of buckets must be sufficiently large that most buckets are not large. In addition, on the second pass, there must be enough room to count all the candidate pairs. As a function of S , what is the largest value of P for which we can successfully run the PCY algorithm on this data? Demonstrate that you have the correct formula by indicating which of the following is a value for S and a value for P that is approximately (i.e., to within 10%) the largest possible value of P for that S .

Your Answer**Score****Explanation**

$S = 200,000,000$; \times 0.00
 $P = 1,600,000,000$

Here are a few simplifying assumptions that hold for the choices of S and P actually occurring in this question:

1. The space needed on the first pass to count items is negligible.
2. On the first pass, buckets without a frequent pair will not have a large enough count to be frequent.
3. Almost all the candidate pairs will be infrequent; i.e., there are much more than 1,000,000 such pairs.

S =

1,000,000,000; P =

35,000,000,000

 S = 300,000,000;

P = 1,800,000,000

 S = 300,000,000;

P = 3,500,000,000

Total 0.00 /
 1.00

Question 2

During a run of Toivonen's Algorithm with set of items {A,B,C,D,E,F,G,H} a sample is found to have the following maximal frequent itemsets: {A,B}, {A,C}, {A,D}, {B,C}, {E}, {F}. Compute the negative border. Then, identify in the list below the set that is NOT in the negative border.

Your Answer	Score	Explanation
-------------	-------	-------------

 {B,E} {D,F} {D}

<input checked="" type="radio"/> {G}	✗ 0.00	This set is in the negative border because it is not frequent, yet each of its immediate proper subsets, i.e., the empty set only, is frequent. Note that a subset of a maximal frequent itemset, such as the empty set, must itself be frequent.
--------------------------------------	---	---

Total 0.00 /
 1.00

Feedback — Week 2: Frequent Itemsets (Basic)

[Help](#)

You submitted this quiz on **Sun 5 Oct 2014 7:51 AM PDT**. You got a score of **3.00** out of **3.00**.

Question 1

Suppose we have transactions that satisfy the following assumptions:

- s , the support threshold, is 10,000.
- There are one million items, which are represented by the integers 0,1,...,999999.
- There are N frequent items, that is, items that occur 10,000 times or more.
- There are one million pairs that occur 10,000 times or more.
- There are $2M$ pairs that occur exactly once. M of these pairs consist of two frequent items, the other M each have at least one nonfrequent item.
- No other pairs occur at all.
- Integers are always represented by 4 bytes.

Suppose we run the a-priori algorithm to find frequent pairs and can choose on the second pass between the triangular-matrix method for counting candidate pairs (a triangular array $\text{count}[i][j]$ that holds an integer count for each pair of items (i, j) where $i < j$). As a function of N and M , what is the minimum number of bytes of main memory needed to execute the a-priori algorithm on this data? Demonstrate that you have the correct formula by selecting, from the choices below, the triple consisting of values for N , M , and the (approximate, i.e., to within 10%) minimum number of bytes of main memory, S , needed for the a-priori algorithm to execute with this data.

Your Answer	Score	Explanation
<input type="radio"/> N = 100,000; M = 40,000,000; S = 800,000,000		
<input checked="" type="radio"/> N = 30,000; M = 200,000,000; S = 1,800,000,000	✓ 1.00	
<input type="radio"/> N = 10,000; M = 50,000,000; S = 600,000,000		
<input type="radio"/> N = 20,000; M = 60,000,000; S = 1,000,000,000		
Total	1.00 / 1.00	

Question Explanation

On the first pass, we need 4,000,000 bytes to count the 1,000,000 items. This number is tiny compared with the amount needed on the second pass in all choices appearing in this question, so we shall ignore the first pass. On the second pass, we need $4N$ bytes to store the ID's of the N frequent items. This amount is also tiny compared to the space needed to count pairs in all

choices for this question, so we shall neglect it.

If we use a triangular table to store the counts of pairs of frequent items, we need $4(N \text{ choose } 2)$ or about $2N^2$ bytes. If we use a hash table to count only the frequent pairs that occur, we need 12 bytes per occurring pair. The number of pairs that occur is 1,000,000 frequent pairs, plus M pairs that are not frequent, but consist of two frequent items. Thus, the form of correct answers will be:

$(N, M, \min(2N^2, 12(1,000,000 + M)))$

For instance, with $N = 100,000$ and $M = 100,000,000$, S is approximately $\min(2*100,000*100,000, 12(1,000,000 + 100,000,000)) = \min(20,000,000,000, 1,212,000,000)$ or approximately 1.2 billion. Note that in this case the hash table is far better than the triangular array.

Question 2

Imagine there are 100 baskets, numbered 1,2,...,100, and 100 items, similarly numbered. Item i is in basket j if and only if i divides j evenly. For example, basket 24 is the set of items $\{1,2,3,4,6,8,12,24\}$. Describe all the association rules that have 100% confidence. Which of the following rules has 100% confidence?

Your Answer	Score	Explanation
<input type="radio"/> $\{2,3,5\} \rightarrow 45$		
<input type="radio"/> $\{8\} \rightarrow 16$		
<input checked="" type="radio"/> $\{12,18\} \rightarrow 36$	✓ 1.00	
<input type="radio"/> $\{3,4,5\} \rightarrow 42$		
Total	1.00 / 1.00	

Question Explanation

In order for the confidence to be 100%, every basket b that contains all the items on the left must contain the item on the right. Since membership in baskets is defined by divisibility, what we're really looking for is that every integer b that is divisible by all the numbers on the left is also divisible by the number on the right. For example, the rule $\{4,6\} \rightarrow 12$ has 100% confidence, because if b is divisible by 4 and 6, it has at least two factors 2 and at least one factor 3. That means it is divisible by $2*2*3 = 12$.

Question 3

Suppose ABC is a frequent itemset and BCDE is NOT a frequent itemset. Given this information, we can be sure that certain other itemsets are frequent and sure that certain itemsets are NOT frequent. Other itemsets may be either frequent or not. Which of the following is a correct

classification of an itemset?

Your Answer	Score	Explanation
<input checked="" type="radio"/> BCD can be either frequent or not frequent.	✓	1.00
<input type="radio"/> AB can be either frequent or not frequent.		
<input type="radio"/> ABCDEF can be either frequent or not frequent.		
<input type="radio"/> ABCD is frequent.		
Total	1.00 / 1.00	

Question Explanation

All subsets of ABC are frequent and all supersets of BCDE are not frequent. Any other itemset can be either frequent or not.

Feedback — Week 2: LSH (Basic)

[Help](#)

You submitted this quiz on **Sun 5 Oct 2014 6:33 AM PDT**. You got a score of **6.00** out of **6.00**.

Question 1

The edit distance is the minimum number of character insertions and character deletions required to turn one string into another. Compute the edit distance between each pair of the strings he, she, his, and hers. Then, identify which of the following is a true statement about the number of pairs at a certain edit distance.

Your Answer	Score	Explanation
<input checked="" type="radio"/> There is 1 pair at distance 2.	✓ 1.00	
<input type="radio"/> There are 2 pairs at distance 4.		
<input type="radio"/> There is 1 pair at distance 3.		
<input type="radio"/> There are 2 pairs at distance 1.		
Total	1.00 / 1.00	

Question Explanation

We need to calculate the edit distance between each of the six pairs of words. Consider $d(\text{he}, \text{she})$, an easy case. You can convert "he" into "she" by one edit: insert "s" at the beginning. Alternatively, convert "she" into "he" by the single edit of deleting the first character. Thus, $d(\text{he}, \text{she}) = 1$. For a harder case, consider $d(\text{she}, \text{his})$. There are two ways to convert "she" into "his" but both take four edits. We could delete "he" from "she", leaving only the "s", and then insert "hi" in front of the "s". Or we could delete "s" and "e" from "she" and then follow the remaining "h" by "is". Either way, 4 edits are needed. Thus, $d(\text{she}, \text{his}) = 4$.

In a similar manner, we can discover $d(\text{he}, \text{his}) = 3$, $d(\text{he}, \text{hers}) = 2$, $d(\text{she}, \text{hers}) = 3$, and $d(\text{his}, \text{hers}) = 3$. A useful rule is that the edit distance is the sum of the lengths of the words minus twice the length of the longest common subsequence. For instance, the longest common subsequence of "his" and "hers" is "hs", so their edit distance is $|\text{his}| + |\text{hers}| - 2|\text{hs}| = 3 + 4 - 2 \cdot 2 = 3$.

Question 2

Consider the following matrix:



	C1	C2	C3	C4
R1	0	1	1	0
R2	1	0	1	1
R3	0	1	0	1
R4	0	0	1	0
R5	1	0	1	0
R6	0	1	0	0

Perform a minhashing of the data, with the order of rows: R4, R6, R1, R3, R5, R2. Which of the following is the correct minhash value of the stated column? **Note:** we give the minhash value in terms of the original name of the row, rather than the order of the row in the permutation. These two schemes are equivalent, since we only care whether hash values for two columns are equal, not what their actual values are.

Your Answer	Score	Explanation
<input type="radio"/> The minhash value for C2 is R1		
<input checked="" type="radio"/> The minhash value for C2 is R6	✓ 1.00	
<input type="radio"/> The minhash value for C4 is R5		
<input type="radio"/> The minhash value for C4 is R2		
Total	1.00 / 1.00	

Question Explanation

Look at the rows in the stated order R4, R6, R1, R3, R5, R2, and for each row, make that row be the minhash value of a column if the column has not yet been assigned a minhash value. We start with R4, which only has 1 in column C3, so the minhash value for C3 is R4.

Next, we consider R6, which has 1 in C2 only. Since C2 does not yet have a minhash value, R6 becomes its value.

Next is R1, with 1's in C2 and C3. However, both these columns already have minhash values, so we do nothing.

Next, consider R3. It has 1's in C2 and C4. C2 already has a minhash value, but C4 does not. Thus, the minhash value of C4 is R3.

When we consider R5 next, we see it has 1's in C1 and C3. The latter already has a minhash value, but R5 becomes the minhash value for C1. Since all columns now have minhash values, we are done.

Question 3

Here is a matrix representing the signatures of seven columns, C1 through C7.

C1	C2	C3	C4	C5	C6	C7
1	2	1	1	2	5	4
2	3	4	2	3	2	2
3	1	2	3	1	3	2
4	1	3	1	2	4	4
5	2	5	1	1	5	1
6	1	6	4	1	1	4

Suppose we use locality-sensitive hashing with three bands of two rows each. Assume there are enough buckets available that the hash function for each band can be the identity function (i.e., columns hash to the same bucket if and only if they are identical in the band). Find all the candidate pairs, and then identify one of them in the list below.

Your Answer	Score	Explanation
<input checked="" type="radio"/> C1 and C4	✓ 1.00	
<input type="radio"/> C1 and C2		
<input type="radio"/> C2 and C6		
<input type="radio"/> C5 and C6		
Total	1.00 / 1.00	

Question Explanation

In the first band (first two rows) C1 and C4 both have (1,2), so they form a candidate pair. Also, C2 and C5 both have (2,3), so that is another candidate pair.

In the second band (rows 3 and 4) we find only C1 and C6 agree, and in the third band we find C1-C3 agree and C4-C7 agree. Thus, the five candidate pairs are C1-C4, C2-C5, C1-C6, C1-C3, and C4-C7.

Question 4

Find the set of 2-shingles for the "document":

ABRACADABRA

and also for the "document":

BRICABRAC

Answer the following questions:

1. How many 2-shingles does ABRACADABRA have?
2. How many 2-shingles does BRICABRAC have?
3. How many 2-shingles do they have in common?
4. What is the Jaccard similarity between the two documents"?

Then, find the true statement in the list below.

Your Answer	Score	Explanation
<input type="radio"/> The Jaccard similarity is 5/7.		
<input type="radio"/> BRICABRAC has 4 2-shingles.		
<input type="radio"/> The Jaccard similarity is 4/7.		
<input checked="" type="radio"/> BRICABRAC has 7 2-shingles.	✓ 1.00	
Total	1.00 / 1.00	

Question Explanation

The 2-shingles for ABRACADABRA: AB, BR, RA, AC, CA, AD, DA.

The 2-shingles for BRICABRAC: BR, RI, IC, CA, AB, RA, AC.

There are 5 shingles in common: AB, BR, RA, AC, CA.

As there are 9 different shingles in all, the Jaccard similarity is 5/9.

Question 5

Here are eight strings that represent sets:

$s_1 = \{a, b, c, e, f\}$

$s_2 = \{a, c, d, e, g\}$

$s_3 = \{b, c, d, e, f, g\}$

$s_4 = \{a, d, f, g\}$

$s_5 = \{b, c, d, f, g, h\}$

$s_6 = \{b, c, e, g\}$

$s_7 = \{c, d, f, g\}$

$s_8 = \{a, b, c, d\}$

Suppose our upper limit on Jaccard distance is 0.2, and we use the indexing scheme of

Section 3.9.4 based on symbols appearing in the prefix (no position or length information).

For each of s_1 , s_3 , and s_6 , determine how many *other* strings that string will be compared with, if it is used as the probe string. Then, identify the true count from the list below.

Your Answer	Score	Explanation
<input type="radio"/> s_3 is compared with 4 other strings.		
<input checked="" type="radio"/> s_3 is compared with 5 other strings.	✓ 1.00	
<input type="radio"/> s_1 is compared with 7 other strings.		
<input type="radio"/> s_3 is compared with 6 other strings.		
Total	1.00 / 1.00	

Question Explanation

First, we index a string of length L on the symbols appearing in its prefix of length $\text{floor}(0.2L+1)$. Thus, strings of length 5 and 6 are indexed on their first two symbols, while strings of length 4 are indexed on their first symbol only. Thus, the index for a consists of $\{s_1, s_2, s_4, s_8\}$; the index for b consists of $\{s_1, s_3, s_5, s_6\}$, the index for c consists of $\{s_2, s_3, s_5, s_7\}$, and no other symbol is indexed at all.

For s_1 , we examine the indexes for a and b , which contains all strings but s_7 . Thus, s_1 is compared with 6 other strings.

For s_3 , we examine the indexes for b and c , which together contain s_1, s_2, s_3, s_5, s_6 , and s_7 . Thus, s_3 is compared with five other strings.

For s_6 , we examine only the index for b . Thus, s_6 is compared only with the three other strings s_1 , s_3 , and s_5 .

Question 6

Suppose we want to assign points to whichever of the points $(0,0)$ or $(100,40)$ is nearer. Depending on whether we use the L_1 or L_2 norm, a point (x,y) could be clustered with a different one of these two points. For this problem, you should work out the conditions under which a point will be assigned to $(0,0)$ when the L_1 norm is used, but assigned to $(100,40)$ when the L_2 norm is used.

Identify one of those points from the list below.

Your Answer	Score	Explanation
<input type="radio"/> $(63,8)$		
<input checked="" type="radio"/> $(59,10)$	✓ 1.00	
<input type="radio"/> $(51,15)$		

(54,8)

Total	1.00 / 1.00
-------	-------------

Question Explanation

The L_1 distance from (x,y) to $(0,0)$ is $x+y$. The L_1 distance from (x,y) to $(100,40)$ is $140-x-y$. Thus, (x,y) is assigned to $(0,0)$ using the L_1 norm if $x < 70-y$.

When comparing L_2 distances, it is often better to use the squares of the distances. The square of the L_2 distance from (x,y) to $(0,0)$ is x^2+y^2 , and the square of the L_2 distance from (x,y) to $(100,40)$ is $(100-x^2)+(40-y^2) = 11600-200x-80y+x^2+y^2$. Thus, for (x,y) to be clustered with $(100,40)$ according to the L_2 norm, we must have $200x+80y > 11600$, or $x > 58-2y/5$. Thus, each of the correct answers is an (x,y) pair with $58-2y/5 < x < 70-y$. For example, if $y=10$, we must have $54 < x < 60$.

Feedback — Week 2: LSH (Basic)

[Help](#)

You submitted this quiz on **Sun 5 Oct 2014 5:58 AM PDT**. You got a score of **4.00** out of **6.00**. You can [attempt again](#), if you'd like.

Question 1

The edit distance is the minimum number of character insertions and character deletions required to turn one string into another. Compute the edit distance between each pair of the strings he, she, his, and hers. Then, identify which of the following is a true statement about the number of pairs at a certain edit distance.

Your Answer	Score	Explanation
<input checked="" type="radio"/> There is 1 pair at distance 2.	✓ 1.00	
<input type="radio"/> There are 3 pairs at distance 4.		
<input type="radio"/> There are 4 pairs at distance 3.		
<input type="radio"/> There are 2 pairs at distance 2.		
Total	1.00 / 1.00	

Question Explanation

We need to calculate the edit distance between each of the six pairs of words. Consider $d(\text{he}, \text{she})$, an easy case. You can convert "he" into "she" by one edit: insert "s" at the beginning. Alternatively, convert "she" into "he" by the single edit of deleting the first character. Thus, $d(\text{he}, \text{she}) = 1$. For a harder case, consider $d(\text{she}, \text{his})$. There are two ways to convert "she" into "his" but both take four edits. We could delete "he" from "she", leaving only the "s", and then insert "hi" in front of the "s". Or we could delete "s" and "e" from "she" and then follow the remaining "h" by "is". Either way, 4 edits are needed. Thus, $d(\text{she}, \text{his}) = 4$.

In a similar manner, we can discover $d(\text{he}, \text{his}) = 3$, $d(\text{he}, \text{hers}) = 2$, $d(\text{she}, \text{hers}) = 3$, and $d(\text{his}, \text{hers}) = 3$. A useful rule is that the edit distance is the sum of the lengths of the words minus twice the length of the longest common subsequence. For instance, the longest common subsequence of "his" and "hers" is "hs", so their edit distance is $|\text{his}| + |\text{hers}| - 2|\text{hs}| = 3 + 4 - 2 \cdot 2 = 3$.

Question 2

Consider the following matrix:



	C1	C2	C3	C4
R1	0	1	1	0
R2	1	0	1	1
R3	0	1	0	1
R4	0	0	1	0
R5	1	0	1	0
R6	0	1	0	0

Perform a minhashing of the data, with the order of rows: R4, R6, R1, R3, R5, R2. Which of the following is the correct minhash value of the stated column? **Note:** we give the minhash value in terms of the original name of the row, rather than the order of the row in the permutation. These two schemes are equivalent, since we only care whether hash values for two columns are equal, not what their actual values are.

Your Answer**Score****Explanation**

- The minhash value for C2 is R3 ✗ 0.00 This answer would only be correct if none of the rows that precede R3 in the list R4, R6, R1, R3, R5, R2 had 1 in column C2.

- The minhash value for C3 is R5

- The minhash value for C2 is R4

- The minhash value for C2 is R6

Total 0.00 / 1.00

Question Explanation

Look at the rows in the stated order R4, R6, R1, R3, R5, R2, and for each row, make that row be the minhash value of a column if the column has not yet been assigned a minhash value. We start with R4, which only has 1 in column C3, so the minhash value for C3 is R4.

Next, we consider R6, which has 1 in C2 only. Since C2 does not yet have a minhash value, R6 becomes its value.

Next is R1, with 1's in C2 and C3. However, both these columns already have minhash values, so we do nothing.

Next, consider R3. It has 1's in C2 and C4. C2 already has a minhash value, but C4 does not. Thus, the minhash value of C4 is R3.

When we consider R5 next, we see it has 1's in C1 and C3. The latter already has a minhash value, but R5 becomes the minhash value for C1. Since all columns now have minhash values, we are done.

Question 3

Here is a matrix representing the signatures of seven columns, C1 through C7.

C1	C2	C3	C4	C5	C6	C7
1	2	1	1	2	5	4
2	3	4	2	3	2	2
3	1	2	3	1	3	2
4	1	3	1	2	4	4
5	2	5	1	1	5	1
6	1	6	4	1	1	4

Suppose we use locality-sensitive hashing with three bands of two rows each. Assume there are enough buckets available that the hash function for each band can be the identity function (i.e., columns hash to the same bucket if and only if they are identical in the band). Find all the candidate pairs, and then identify one of them in the list below.

Your Answer	Score	Explanation
<input type="radio"/> C3 and C5		
<input checked="" type="radio"/> C4 and C7	✓ 1.00	
<input type="radio"/> C2 and C6		
<input type="radio"/> C2 and C3		
Total	1.00 / 1.00	

Question Explanation

In the first band (first two rows) C1 and C4 both have (1,2), so they form a candidate pair. Also, C2 and C5 both have (2,3), so that is another candidate pair.

In the second band (rows 3 and 4) we find only C1 and C6 agree, and in the third band we find C1-C3 agree and C4-C7 agree. Thus, the five candidate pairs are C1-C4, C2-C5, C1-C6, C1-C3, and C4-C7.

Question 4

Find the set of 2-shingles for the "document":

ABRACADABRA

and also for the "document":

BRICABRAC

Answer the following questions:

1. How many 2-shingles does ABRACADABRA have?
2. How many 2-shingles does BRICABRAC have?
3. How many 2-shingles do they have in common?
4. What is the Jaccard similarity between the two documents"?

Then, find the true statement in the list below.

Your Answer	Score	Explanation
<input type="radio"/> BRICABRAC has 6 2-shingles.		
<input type="radio"/> BRICABRAC has 8 2-shingles.		
<input type="radio"/> There are 4 shingles in common.		
<input checked="" type="radio"/> BRICABRAC has 7 2-shingles.	✓ 1.00	
Total	1.00 / 1.00	

Question Explanation

The 2-shingles for ABRACADABRA: AB, BR, RA, AC, CA, AD, DA.

The 2-shingles for BRICABRAC: BR, RI, IC, CA, AB, RA, AC.

There are 5 shingles in common: AB, BR, RA, AC, CA.

As there are 9 different shingles in all, the Jaccard similarity is 5/9.

Question 5

Here are eight strings that represent sets:

$s_1 = abcef$

$s_2 = acdeg$

$s_3 = bcdefg$

$s_4 = adfg$

$s_5 = bcd fgh$

$s_6 = bceg$

$s_7 = cdf g$

$s_8 = abcd$

Suppose our upper limit on Jaccard distance is 0.2, and we use the indexing scheme of Section 3.9.4 based on symbols appearing in the prefix (no position or length information). For each of s_1 , s_3 , and s_6 , determine how many *other* strings that string will be compared with, if it is used as the probe string. Then, identify the true count from the list below.

Your Answer	Score	Explanation
<input type="radio"/> s_6 is compared with 3 other strings.		
<input type="radio"/> s_1 is compared with 7 other strings.		
<input checked="" type="radio"/> s_1 is compared with 5 other strings.	✗ 0.00	
<input type="radio"/> s_6 is compared with 2 other strings.		
Total	0.00 / 1.00	

Question Explanation

First, we index a string of length L on the symbols appearing in its prefix of length $\text{floor}(0.2L+1)$. Thus, strings of length 5 and 6 are indexed on their first two symbols, while strings of length 4 are indexed on their first symbol only. Thus, the index for a consists of $\{s_1, s_2, s_4, s_8\}$; the index for b consists of $\{s_1, s_3, s_5, s_6\}$, the index for c consists of $\{s_2, s_3, s_5, s_7\}$, and no other symbol is indexed at all.

For s_1 , we examine the indexes for a and b , which contains all strings but s_7 . Thus, s_1 is compared with 6 other strings.

For s_3 , we examine the indexes for b and c , which together contain s_1, s_2, s_3, s_5, s_6 , and s_7 . Thus, s_3 is compared with five other strings.

For s_6 , we examine only the index for b . Thus, s_6 is compared only with the three other strings s_1 , s_3 , and s_5 .

Question 6

Suppose we want to assign points to whichever of the points $(0,0)$ or $(100,40)$ is nearer. Depending on whether we use the L_1 or L_2 norm, a point (x,y) could be clustered with a different one of these two points. For this problem, you should work out the conditions under which a point will be

assigned to (0,0) when the L₁ norm is used, but assigned to (100,40) when the L₂ norm is used.

Identify one of those points from the list below.

Your Answer	Score	Explanation
<input type="radio"/> (63,8)		
<input type="radio"/> (56,15)		
<input checked="" type="radio"/> (61,8)		1.00
<input type="radio"/> (53,10)		
Total	1.00 / 1.00	

Question Explanation

The L₁ distance from (x,y) to (0,0) is x+y. The L₁ distance from (x,y) to (100,40) is 140-x-y. Thus, (x,y) is assigned to (0,0) using the L₁ norm if x < 70-y.

When comparing L₂ distances, it is often better to use the squares of the distances. The square of the L₂ distance from (x,y) to (0,0) is x²+y², and the square of the L₂ distance from (x,y) to (100,40) is (100-x)²+(40-y)² = 11600-200x-80y+x²+y². Thus, for (x,y) to be clustered with (100,40) according to the L₂ norm, we must have 200x+80y > 11600, or x > 58-2y/5. Thus, each of the correct answers is an (x,y) pair with 58-2y/5 < x < 70-y. For example, if y=10, we must have 54 < x < 60.

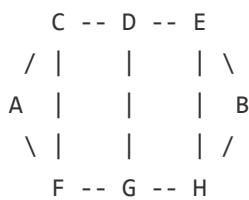
Feedback — Week3A (Advanced)

[Help](#)

You submitted this quiz on **Sat 25 Oct 2014 1:47 PM PDT**. You got a score of **5.00** out of **5.00**.

Question 1

For the following graph:



Write the adjacency matrix A, the degree matrix D, and the Laplacian matrix L. For each, find the sum of all entries and the number of nonzero entries. Then identify the true statement from the list below.

Your Answer**Score****Explanation**

- The sum of the entries of L is 44.
- The sum of the entries of L is 22.
- D has 8 nonzero entries.
- The sum of the entries of A is 8.

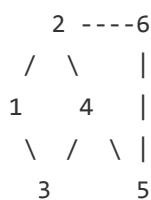
1.00

Total

1.00 / 1.00

Question 2

You are given the following graph.



The goal is to find two clusters in this graph using Spectral Clustering on the Laplacian matrix. Compute the Laplacian of this graph. Then compute the second eigen vector of the Laplacian (the one corresponding to the second smallest eigenvalue).

To cluster the points, we decide to split at the mean value. We say that a node is a tie if its value in the eigen-vector is exactly equal to the mean value. Let's assume that if a point is a tie, we choose its cluster at random. Identify the true statement from the list below.

Your Answer	Score	Explanation
<input checked="" type="radio"/> 4 and 6 can either be in the same cluster or in different clusters (depending on randomness)	✓ 1.00	
<input type="radio"/> 5 and 6 can either be in the same cluster or in different clusters (depending on randomness)		
<input type="radio"/> 3 and 5 can either be in the same cluster or in different clusters (depending on randomness)		
<input type="radio"/> 3 is a tie		
Total	1.00 / 1.00	

Question 3

We wish to estimate the surprise number (2nd moment) of a data stream, using the method of AMS. It happens that our stream consists of ten different values, which we'll call 1, 2,..., 10, that cycle repeatedly. That is, at timestamps 1 through 10, the element of the stream equals the timestamp, at timestamps 11 through 20, the element is the timestamp minus 10, and so on. It is now timestamp 75, and a 5 has just been read from the stream. As a start, you should calculate the surprise number for this time.

For our estimate of the surprise number, we shall choose three timestamps at random, and estimate the surprise number from each, using the AMS approach (length of the stream times $2m-1$, where m is the number of occurrences of the element of the stream at that timestamp, considering all times from that timestamp on, to the current time). Then, our estimate will be the median of the three resulting values.

You should discover the simple rules that determine the estimate derived from any given timestamp and from any set of three timestamps. Then, identify from the list below the set of three "random" timestamps that give the closest estimate.

Your Answer	Score	Explanation
<input type="radio"/> {37, 46, 55}		
<input type="radio"/> {30, 47, 62}		
<input type="radio"/> {31, 32, 44}		
<input checked="" type="radio"/> {22, 42, 62}	✓ 1.00	
Total	1.00 / 1.00	

Question 4

We wish to use the Flajolet-Martin algorithm of Section 4.4 to count the number of distinct elements in a stream. Suppose that there are ten possible elements, 1, 2, ..., 10, that could appear in the stream, but only four of them have actually appeared. To make our estimate of the count of distinct elements, we hash each element to a 4-bit binary number. The element x is hashed to $3x + 7$ (modulo 11). For example, element 8 hashes to $3*8+7 = 31$, which is 9 modulo 11 (i.e., the remainder of 31/11 is 9). Thus, the 4-bit string for element 8 is 1001.

A set of four of the elements 1 through 10 could give an estimate that is exact (if the estimate is 4), or too high, or too low. You should figure out under what circumstances a set of four elements falls into each of those categories. Then, identify in the list below the set of four elements that gives the exactly correct estimate.

Your Answer	Score	Explanation
<input type="radio"/> {2, 3, 6, 9}		
<input checked="" type="radio"/> {1, 6, 7, 10}	✓ 1.00	
<input type="radio"/> {1, 4, 7, 9}		
<input type="radio"/> {2, 5, 7, 10}		
Total	1.00 / 1.00	

Question 5

Suppose we are using the DGIM algorithm of Section 4.6.2 to estimate the number of 1's in suffixes of a sliding window of length 40. The current timestamp is 100, and we have the following buckets stored:

End Time	100	98	95	92	87	80	65
Size	1	1	2	2	4	8	8

Note: we are showing timestamps as absolute values, rather than modulo the window size, as DGIM would do.

Suppose that at times 101 through 105, 1's appear in the stream. Compute the set of buckets that would exist in the system at time 105. Then identify one such bucket from the list below. Buckets are represented by pairs (end-time, size).

Your Answer	Score	Explanation
<input type="radio"/> (103,2)		
<input type="radio"/> (104,1)		
<input checked="" type="radio"/> (104,2)	✓	1.00
<input type="radio"/> (103,1)		
Total	1.00 / 1.00	

Feedback — Week3B (Basic)

[Help](#)

You submitted this quiz on **Sat 25 Oct 2014 12:07 PM PDT**. You got a score of **2.00 out of 2.00**.

Question 1

Suppose we hash the elements of a set S having 20 members, to a bit array of length 99. The array is initially all-0's, and we set a bit to 1 whenever a member of S hashes to it. The hash function is random and uniform in its distribution. What is the expected fraction of 0's in the array after hashing? What is the expected fraction of 1's? You may assume that 99 is large enough that asymptotic limits are reached.

Your Answer	Score	Explanation
<input checked="" type="radio"/> The fraction of 1's is $1-e^{-20/99}$.	✓	1.00
<input type="radio"/> The fraction of 1's is 79/99.		
<input type="radio"/> The fraction of 0's is 20/99.		
<input type="radio"/> The fraction of 1's is 20/99.		
Total	1.00 / 1.00	

Question 2

A certain Web mail service (like gmail, e.g.) has 10^8 users, and wishes to create a sample of data about these users, occupying 10^{10} bytes. Activity at the service can be viewed as a stream of elements, each of which is an email. The element contains the ID of the sender, which must be one of the 10^8 users of the service, and other information, e.g., the recipient(s), and contents of the message. The plan is to pick a subset of the users and collect in the 10^{10} bytes records of length 100 bytes about every email sent by the users in the selected set (and nothing about other users).

The method of Section 4.2.4 will be used. User ID's will be hashed to a bucket number, from 0 to 999,999. At all times, there will be a threshold t such that the 100-byte records for all the users whose ID's hash to t or less will be retained, and other users' records will not be

retained. You may assume that each user generates emails at exactly the same rate as other users. As a function of n , the number of emails in the stream so far, what should the threshold t be in order that the selected records will not exceed the 10^{10} bytes available to store records? From the list below, identify the true statement about a value of n and its value of t .

Your Answer	Score	Explanation
<input type="radio"/> $n = 10^{12}; t = 100$		
<input checked="" type="radio"/> $n = 10^{13}; t = 9$	✓	1.00
<input type="radio"/> $n = 10^{11}; t = 1000$		
<input type="radio"/> $n = 10^{10}; t = 10,000$		
Total	1.00 / 1.00	

Feedback — Week4A (Basic)

Help

You submitted this quiz on **Tue 21 Oct 2014 11:40 AM PDT**. You got a score of **2.00** out of **2.00**.

Question 1

Here is a table of 1-5 star ratings for five movies (M, N, P, Q, R) by three raters (A, B, C).

	M	N	P	Q	R
A	1	2	3	4	5
B	2	3	2	5	3
C	5	5	5	3	2

Normalize the ratings by subtracting the average for each row and then subtracting the average for each column in the resulting table. Then, identify the true statement about the normalized table.

Your Answer	Score	Explanation
<input checked="" type="radio"/> The largest element is (A,R)	✓	1.00
<input type="radio"/> The entry (B,N) is -1/3.		
<input type="radio"/> The entry (C,M) is -5/3.		
<input type="radio"/> The smallest element is (B,P).		
Total	1.00 / 1.00	

Question 2

Below is a table giving the profile of three items.

A	1	0	1	0	1	2
B	1	1	0	0	1	6
C	0	1	0	1	0	2

The first five attributes are Boolean, and the last is an integer "rating." Assume that the scale factor for the rating is α . Compute, as a function of α , the cosine distances between each pair of profiles. For each of $\alpha = 0, 0.5, 1$, and 2 , determine the cosine of the angle between each pair of vectors. Which of the following is FALSE?

Your Answer	Score	Explanation
<input type="radio"/> For $\alpha = 1$, B is closer to C than A is.		
<input type="radio"/> For $\alpha = 2$, C is closer to B than A is.		
<input type="radio"/> For $\alpha = 2$, B is closer to A than C is.		
<input checked="" type="radio"/> For $\alpha = 2$, A is closer to C than B is.	✓ 1.00	
Total	1.00 / 1.00	

Feedback — Week4B (Basic)

[Help](#)

You submitted this quiz on **Thu 23 Oct 2014 11:44 PM PDT**. You got a score of **4.00** out of **4.00**.

Question 1

Note: In this question, all columns will be written in their transposed form, as rows, to make the typography simpler. Matrix M has three rows and two columns, and the columns form an orthonormal basis. One of the columns is $[2/7, 3/7, 6/7]$. There are many options for the second column $[x, y, z]$. Write down those constraints on x, y, and z. Then, identify in the list below the one column that could be $[x, y, z]$. All components are computed to three decimal places, so the constraints may be satisfied only to a close approximation.

Your Answer	Score	Explanation
<input checked="" type="radio"/> [-.937, .312, .156]	✓ 1.00	
<input type="radio"/> [-.857, .286, .429]		
<input type="radio"/> [.890, -.346, -.297]		
<input type="radio"/> [.429, .857, .286]		
Total	1.00 / 1.00	

Question 2

Note: In this question, all columns will be written in their transposed form, as rows, to make the typography simpler. Matrix M has three rows and three columns, and the columns form an orthonormal basis. One of the columns is $[2/7, 3/7, 6/7]$, and another is $[6/7, 2/7, -3/7]$. Let the third column be $[x, y, z]$. Since the length of the vector $[x, y, z]$ must be 1, there is a constraint that $x^2 + y^2 + z^2 = 1$. However, there are other constraints, and these other constraints can be used to deduce facts about the ratios among x, y, and z. Compute these ratios, and then identify one of them in the list below.

Your Answer	Score	Explanation

$2z = -3x$ $y = -2x$ 

1.00

 $y = 3z$ $2x = -3z$

Total

1.00 / 1.00

Question 3

Suppose we have three points in a two dimensional space: (1,1), (2,2), and (3,4). We want to perform PCA on these points, so we construct a 2-by-2 matrix whose eigenvectors are the directions that best represent these three points. Construct this matrix and identify, in the list below, one of its elements.

Your Answer**Score****Explanation** 11 14

1.00

 24 15

Total

1.00 / 1.00

Question 4

Find, in the list below, the vector that is orthogonal to the vector [1,2,3]. Note: the interesting concept regarding eigenvectors is "orthonormal," that is unit vectors that are orthogonal. However, this question avoids using unit vectors to make the calculations simpler.

Your Answer**Score****Explanation** [-1, -2, -3] [-3, -2, 5] [1, -2, 1]

1.00

[-3, 4, -2]

Total

1.00 / 1.00

Feedback — Week5A (Advanced)

[Help](#)

You submitted this quiz on **Fri 31 Oct 2014 2:24 PM PDT**. You got a score of **3.00** out of **3.00**.

Question 1

Consider the diagonal matrix $M =$

1	0	0
0	2	0
0	0	0

. Compute its Moore-Penrose pseudoinverse, and then identify, in the list below, the true statement about the elements of the pseudoinverse.

Your Answer**Score****Explanation**

- There is one element with value 2.
- There is one element with value -2.
- There is one element with value -1.
- There are seven elements with value 0.

 1.00**Total**

1.00 / 1.00

Question 2

An ad publisher selects three ads to place on each page, in order from the top. Click-through rates (CTR's) at each position differ for each advertiser, and each advertiser has a different CTR for each position. Each advertiser bids for click-throughs, and each advertiser has a daily budget, which may not be exceeded. When a click-through occurs, the advertiser pays the amount they bid. In one day, there are 101 click-throughs to be auctioned.

Here is a table of the bids, CTR's for positions 1, 2, and 3, and budget for each advertiser.

Advertiser	Bid	CTR1	CTR2	CTR3	Budget
A	\$.10	.015	.010	.005	\$ 1

B	\$.09	.016	.012	.006	\$2
C	\$.08	.017	.014	.007	\$3
D	\$.07	.018	.015	.008	\$4
E	\$.06	.019	.016	.010	\$5

The publisher uses the following strategy to allocate the three ad slots:

1. Any advertiser whose budget is spent is ignored in what follows.
2. The first slot goes to the advertiser whose expected yield for the first slot (product of the bid and the CTR for the first slot) is the greatest. This advertiser is ignored in what follows.
3. The second slot goes to the advertiser whose expected yield for the second slot (product of the bid and the CTR for the second slot) is the greatest. This advertiser is ignored in what follows.
4. The third slot goes to the advertiser whose expected yield for the third slot (product of the bid and the CTR for the third slot) is the greatest.

The same three advertisers get the three ad positions until one of two things happens:

1. An advertiser runs out of budget, or
2. All 101 click-throughs have been obtained.

Either of these events ends one *phase* of the allocation. If a phase ends because an advertiser ran out of budget, then they are assumed to get all the clicks their budget buys. During the same phase, we calculate the number of click-throughs received by the other two advertisers by assuming that all three received click-throughs in proportion to their respective CTR's for their positions (round to the nearest integer). If click-throughs remain, the publisher reallocates all three slots and starts a new phase.

If the phase ends because all click-throughs have been allocated, assume that the three advertisers received click-throughs in proportion to their respective CTR's (again, rounding if necessary).

Your task is to simulate the allocation of slots and to determine how many click-throughs each of the five advertisers get.

Your Answer	Score	Explanation
<input type="radio"/> B gets 30 click-throughs.		
<input type="radio"/> A gets 57 click-throughs.		
<input type="radio"/> A gets 5 click-throughs.		
<input checked="" type="radio"/> B gets 22 click-throughs.	✓ 1.00	
Total	1.00 / 1.00	

Question 3

In certain clustering algorithms, such as CURE, we need to pick a representative set of points in a supposed cluster, and these points should be as far away from each other as possible. That is, begin with the two furthest points, and at each step add the point whose minimum distance to any of the previously selected points is maximum.

Suppose you are given the following points in two-dimensional Euclidean space: $x = (0,0)$; $y = (10,10)$, $a = (1,6)$; $b = (3,7)$; $c = (4,3)$; $d = (7,7)$, $e = (8,2)$; $f = (9,5)$. Obviously, x and y are furthest apart, so start with these. You must add five more points, which we shall refer to as the first, second,..., fifth points in what follows. The distance measure is the normal Euclidean L_2 -norm. Which of the following is true about the order in which the five points are added?

Your Answer	Score	Explanation
<input checked="" type="radio"/> f is added fifth	✓ 1.00	
<input type="radio"/> a is added third		
<input type="radio"/> d is added second		
<input type="radio"/> c is added first		
Total	1.00 / 1.00	

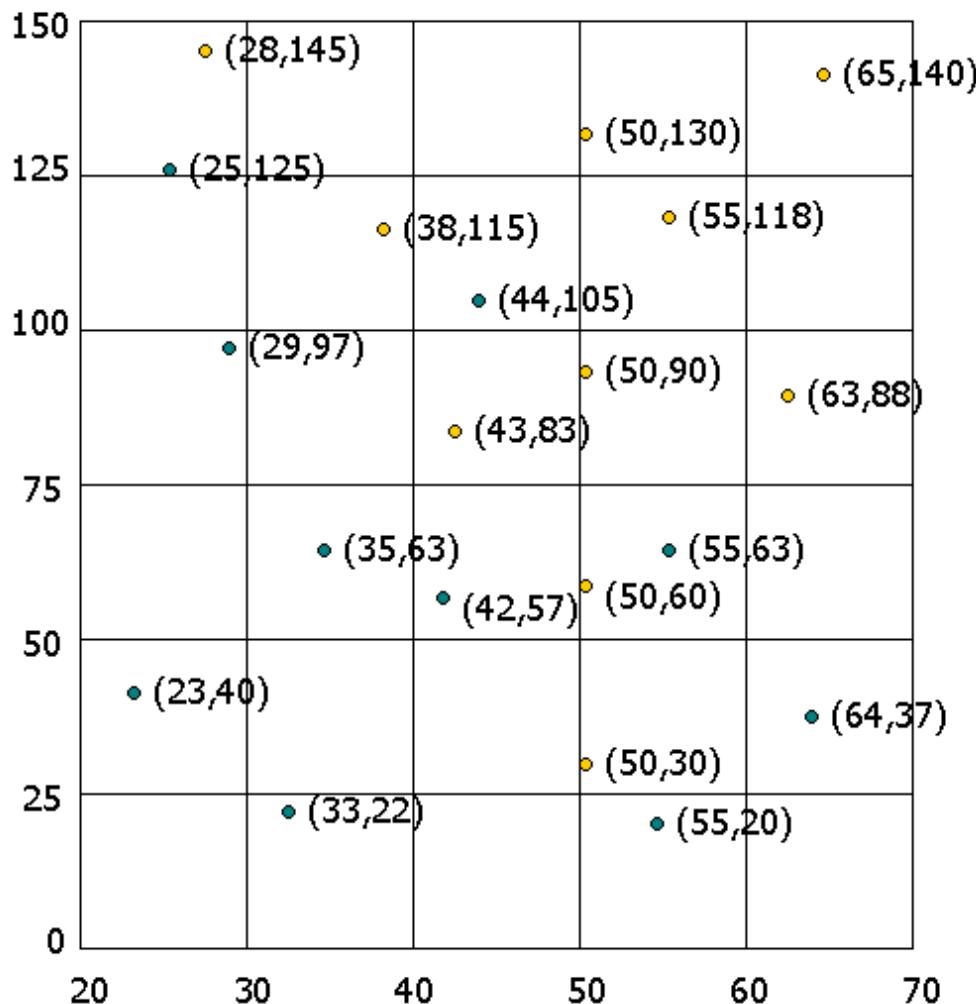
Feedback — Week5B (Basic)

[Help](#)

You submitted this quiz on **Fri 31 Oct 2014 7:45 AM PDT**. You got a score of **5.00** out of **5.00**.

Question 1

We wish to cluster the following set of points:

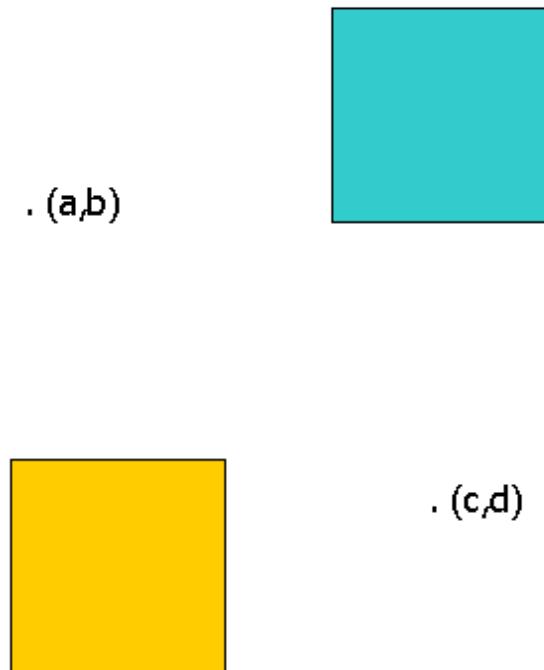


into 10 clusters. We initially choose each of the green points as a centroid. Assign each of the gold points to their nearest centroid. (Note: the scales of the horizontal and vertical axes differ, so you really need to apply the formula for distance of points; you can't just "eyeball" it.) Then, recompute the centroids of each of the clusters. Do any of the points then get reassigned to a new cluster on the next round? Identify the true statement in the list below. Each statement refers either to a centroid AFTER recomputation of centroids (precise to one decimal place) or to a point that gets reclassified.

Your Answer	Score	Explanation
<input checked="" type="radio"/> The point (65,140) is reassigned after computation of centroids.	✓	1.00
<input type="radio"/> There is a centroid after recomputation at (56,70.3)		
<input type="radio"/> The point (38,115) is reassigned after computation of centroids.		
<input type="radio"/> There is a centroid after recomputation at (46,58.5)		
Total	1.00 /	
	1.00	

Question 2

When performing a k-means clustering, success depends very much on the initially chosen points. Suppose that we choose two centroids $(a,b) = (5,10)$ and $(c,d) = (20,5)$, and the data truly belongs to two rectangular clusters, as suggested by the following diagram:



Under what circumstances will the initial clustering be successful? That is, under what conditions will all the yellow points be assigned to the centroid $(5,10)$, while all of the blue points are assigned to cluster $(20,5)$? Identify in the list below, a pair of rectangles (described by their upper left corner, UL, and their lower-right corner LR) that are successfully clustered.

Your Answer	Score	Explanation
<input type="radio"/> Yellow: UL=(3,3) and LR=(10,1); Blue: UL=(13,10) and LR=(16,4)		
<input checked="" type="radio"/> Yellow: UL=(3,3) and LR=(10,1); Blue: UL=(15,14) and LR=(20,10)	1.00	
<input type="radio"/> Yellow: UL=(6,15) and LR=(13,7); Blue: UL=(16,19) and LR=(25,12)		
<input type="radio"/> Yellow: UL=(7,8) and LR=(12,5); Blue: UL=(13,10) and LR=(16,4)		
Total	1.00 / 1.00	

Question 3

Suppose we apply the BALANCE algorithm with bids of 0 or 1 only, to a situation where advertiser A bids on query words x and y, while advertiser B bids on query words x and z. Both have a budget of \$2. Identify in the list below a sequence of four queries that will certainly be handled optimally by the algorithm.

Your Answer	Score	Explanation
<input type="radio"/> xyxz		
<input type="radio"/> xyyx		
<input checked="" type="radio"/> zzzz	1.00	Note that the optimum only yields \$3.
<input type="radio"/> xxxz		
Total	1.00 / 1.00	

Question 4

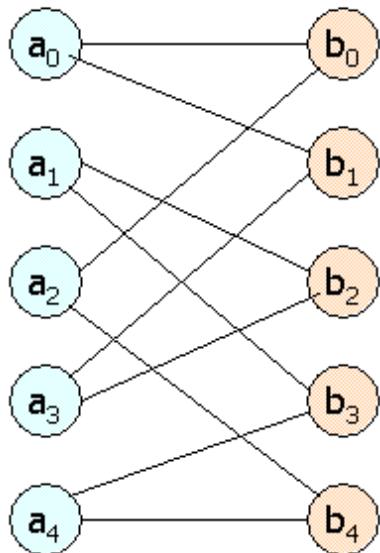
The set cover problem is: given a list of sets, find a smallest collection of these sets such that every element in any of the sets is in at least one set of the collection. As we form a collection, we say an element is covered if it is in at least one set of the collection. Note: In this problem, we shall represent sets by concatenating their elements, without brackets or commas. For example,

{A,B} will be represented simply as AB. There are many greedy algorithms that could be used to pick a collection of sets that is close to as small as possible. Here are some that you will consider in this problem. Dumb: Select sets for the collection in the order in which they appear on the list. Stop when all elements are covered. Simple: Consider sets in the order in which they appear on the list. When it is considered, select a set if it has at least one element that is not already covered. Stop when all elements are covered. Largest-First: Consider sets in order of their size. If there are ties, break the tie in favor of the one that appears first on the list. When it is considered, select a set if it has at least one element that is not already covered. Stop when all elements are covered. Most-Help: Consider sets in order of the number of elements they contain that are not already covered. If there are ties, break the tie in favor of the one that appears first on the list. Stop when all elements are covered. Here is a list of sets: AB, BC, CD, DE, EF, FG, GH, AH, ADG, ADF First, determine the optimum solution, that is, the fewest sets that can be selected for a collection that covers all eight elements A,B,...,H. Then, determine the sizes of the collections that will be constructed by each of the four algorithms mentioned above. Compute the ratio of the size returned by the algorithm to the optimum size, and identify one of these ratios in the list below, correct to two decimal places.

Your Answer	Score	Explanation
<input type="radio"/> The ratio for Most-Help is 1.33		
<input type="radio"/> The ratio for Simple is 2.33		
<input type="radio"/> The ratio for Largest-First is 1.25		
<input checked="" type="radio"/> The ratio for Dumb is 1.75	✓ 1.00	
Total	1.00 / 1.00	

Question 5

This bipartite graph:



Has several perfect matchings. Find all the perfect matchings and then identify, in the list below, a pair of edges that can appear together in a perfect matching.

Your Answer**Score****Explanation**

- a₄-b₃ and a₂-b₀
- a₁-b₂ and a₂-b₀
- a₂-b₄ and a₃-b₂
- a₂-b₄ and a₁-b₂



1.00

Total

1.00 / 1.00

Feedback — Final (Basic)

[Help](#)

You submitted this exam on **Fri 28 Nov 2014 1:45 PM PST**. You got a score of **19.60** out of **24.00**.

This is the Basic Final Exam for the MMDS Course. All students are expected to take this exam. You must submit your work within 3 hours of opening. The exam is open-book; any inanimate source may be used. There is no penalty for a wrong answer (compared with no answer), so feel free to guess.

Question 1

How many distinct 3-shingles are there in the string "hello world"? (Note: the quotes are not part of the string.)

Your Answer	Score	Explanation
<input checked="" type="radio"/> 9	✓	1.00
<input type="radio"/> 4		
<input type="radio"/> 10		
<input type="radio"/> 6		
Total	1.00 / 1.00	

Question Explanation

each position except the last two start a 3-shingle, so there are 9 shingles.

Question 2

Here is a column representing a set, whose minhash value we wish to compute. The hash function we shall use to determine the order of rows is $h(x) = (3x+2) \text{ modulo } 11$.

Row Number	Column Value
1	0

2	1
3	0
4	1
5	0

What is the minhash value for this column? Note: take the minhash value to be the row number, NOT the hash value of the row number.

Your Answer	Score	Explanation
<input type="radio"/> 1		
<input type="radio"/> 2		
<input type="radio"/> 3		
<input checked="" type="radio"/> 4	✓ 1.00	
<input type="radio"/> 5		
Total	1.00 / 1.00	

Question Explanation

Only 2 and 4 are candidates. But $(3*4+2) \bmod 11 = 3$, and $(3*2+2) \bmod 11 = 8$. so 4 is the lowest.

Question 3

This question involves three different Bloom-filter-like scenarios. Each scenario involves setting to 1 certain bits of a 10-bit array, each bit of which is initially 0.

Scenario A: we use one hash function that randomly, and with equal probability, selects one of the ten bits of the array. We apply this hash function to four different inputs and set to 1 each of the selected bits.

Scenario B: We use two hash functions, each of which randomly, with equal probability, and independently of the other hash function selects one of the 10 bits of the array. We apply both hash functions to each of two inputs and set to 1 each of the selected bits.

Scenario C: We use one hash function that randomly and with equal probability selects two **different** bits among the ten in the array. We apply this hash function to two inputs and set to 1 each of the selected bits.

Let a, b, and c be the expected number of bits set to 1 under scenarios A, B, and C, respectively. Which of the following correctly describes the relationships among a, b, and c?

Your Answer	Score	Explanation
<input type="radio"/> a < b = c		
<input type="radio"/> a = b < c		
<input type="radio"/> a = b = c		
<input checked="" type="radio"/> a < b < c	✗ 0.00	
Total	0.00 / 1.00	

Question Explanation

In Scenarios A and B, each of the 10 bits will not be selected $(9/10)^4 = .6561$ fraction of the time. Therefore, $a = b = .3439$.

In Scenario C, the first input definitely sets two of the ten bits to 1. We may as well assume they are the first two bits. When the second input is hashed, two bits are also selected. There are $(10 \text{ choose } 2) = 45$ pairs. One of these pairs will be the first two, and in that case, only 2 bits are set to 1. There are 16 pairs consisting of one of the first two and one of the last 8, and in this case three bits are set to 1. In the other $(8 \text{ choose } 2) = 28$ pairs, both were not previously set to 1, so four bits wind up set. The expected number of 1's is therefore $2*(1/45) + 3*(16/45) + 4*(28/45) = 3.6$. Thus $c = 3.6/10 = 0.36$, which is greater than a and b.

Question 4

In this market-basket problem, there are 99 items, numbered 2 to 100. There is a basket for each prime number between 2 and 100. The basket for p contains all and only the items whose numbers are a multiple of p. For example, the basket for 17 contains the following items: {17, 34, 51, 68, 85}. What is the support of the pair of items {12, 30}?

Your Answer	Score	Explanation
<input type="radio"/> 2		
<input checked="" type="radio"/> 3	✗ 0.00	
<input type="radio"/> 4		
<input type="radio"/> 5		
Total	0.00 / 1.00	

Question Explanation

These two items appear in each basket that corresponds to a prime dividing both of them.
These primes are only 2 and 3.

Question 5

To two decimal places, what is the cosine of the angle between the vectors [2,1,1] and [10,-7,1]?

Your Answer	Score	Explanation
<input type="radio"/> 0.84		
<input type="radio"/> 0.65		
<input checked="" type="radio"/> 0.47	✓ 1.00	
<input type="radio"/> -0.38		
Total	1.00 / 1.00	

Question Explanation

The dot product is $2*10 - 1*7 + 1*1 = 14$. The lengths of the two vectors are $\sqrt{4+1+1}$ and $\sqrt{100+49+1}$, so the product of the lengths is $\sqrt{900} = 30$. Therefore, the cosine of the angle is $14/30 = 0.47$.

Question 6

In this question we use six minhash functions, organized as three bands of two rows each, to identify sets of high Jaccard similarity. If two sets have Jaccard similarity 0.6, what is the probability (to two decimal places) that this pair will become a candidate pair?

Your Answer	Score	Explanation
<input checked="" type="radio"/> 0.74	✓ 1.00	
<input type="radio"/> 0.36		
<input type="radio"/> 0.64		
<input type="radio"/> 0.26		
Total	1.00 / 1.00	

Question Explanation

The probability of agreeing in both rows of one band is $(0.6)^2 = 0.36$. Thus, the probability of NOT agreeing in a given band is $1 - 0.36 = 0.64$. The probability of not agreeing in any of the three bands is $(0.64)^3 = 0.26$. Thus, the probability the two sets will agree in at least one band is $1 - 0.26 = 0.74$.

Question 7

Suppose we have a (.4, .6, .9, .1)-sensitive family of functions. If we apply a 3-way OR construction to this family, we get a new family of functions whose sensitivity is:

Your Answer	Score	Explanation
<input type="radio"/> (.4, .6, .973, .271)		
<input type="radio"/> (.4, .6, .999, .729)		
<input type="radio"/> (.4, .6, .973, .729)		
<input checked="" type="radio"/> (.4, .6, .999, .271)	✓ 1.00	
Total	1.00 / 1.00	

Question Explanation

The 3-way OR replaces a probability p by $1 - (1-p)^3$. Thus, .9 is transformed to $1 - (1 - .9)^3 = .999$, and .1 is transformed to $1 - (1 - .1)^3 = .271$. Thus, the new family is (.4, .6, .999, .271)-sensitive.

Question 8

Suppose we have a database of (Class, Student, Grade) facts, each giving the grade the student got in the class. We want to estimate the fraction of students who have gotten A's in at least 10 classes, but we do not want to examine the entire relation, just a sample of 10% of the tuples. We shall hash tuples to 10 buckets, and take only those tuples in the first bucket. But to get a valid estimate of the fraction of students with at least 10 A's, we need to pick our hash key judiciously. To which Attribute(s) of the relation should we apply the hash function?

Your Answer	Score	Explanation
<input type="radio"/> Student only		
<input type="radio"/> Class only		
<input type="radio"/> Class and Grade		

<input checked="" type="radio"/> Student and Class	✗	0.00
Total		0.00 / 1.00

Question Explanation

We need to get all or none of the tuples for each student, since having only a fraction of the tuples for a given student in our sample will not allow us to determine their number of A's. Thus, the hash key is only Student.

Question 9

Suppose the Web consists of four pages A, B, C, and D, that form a chain

A-->B-->C-->D

We wish to compute the PageRank of each of these pages, but since D is a "dead end," we will "teleport" from D with probability 1 to one of the four pages, each with equal probability. We do not teleport from pages A, B, or C. Assuming the sum of the PageRanks of the four pages is 1, what is the PageRank of page B, correct to two decimal places?

Your Answer

Score

Explanation

0.33

0.40

0.25

0.20



1.00

Total

1.00 / 1.00

Question Explanation

The equations are $A = D/4$, $B = A + D/4$, $C = B + D/4$, and $D = C + D/4$. If we substitute $D/4$ for A in the last three equations we get $B = D/2$, $C = 3D/4$, and $D = D$. That is, the four PageRanks are in the ratio 1:2:3:4. Since their sum is 1, the PageRank of B must be 0.2.

Question 10

Suppose in the AGM model we have four individuals (A,B,C,D) and two communities. Community 1 consists of {A,B,C} and Community 2 consists of {B,C,D}. For Community 1 there is a 30%

chance it will cause an edge between any two of its members. For Community 2 there is a 40% chance it will cause an edge between any two of its members. To the nearest two decimal places, what is the probability that there is an edge between B and C?

Your Answer	Score	Explanation
<input checked="" type="radio"/> 0.58	✓	1.00
<input type="radio"/> 0.40		
<input type="radio"/> 0.70		
<input type="radio"/> 0.42		
Total	1.00 / 1.00	

Question Explanation

The probability that Community 1 will NOT cause an edge between B and C is 0.7, and the probability that Community 2 will not cause an edge is 0.6. Thus the probability that neither will cause an edge is $0.7 \times 0.6 = 0.42$. The probability that there WILL be an edge B-C is thus 0.58.

Question 11

X is a dataset of n columns for which we train a supervised Machine Learning algorithm. e is the error of the model measured against a validation dataset. Unfortunately, e is too high because model has overfitted on the training data X and it doesn't generalize well. We now decide to reduce the model variance by reducing the dimensionality of X, using a Singular Value Decomposition, and using the resulting dataset to train our model. If i is the number of singular values used in the SVD reduction, how does e change as a function of i, for $i \in \{1, 2, \dots, n\}$?

Your Answer	Score	Explanation
<input type="radio"/> e starts high, then decreases.		
<input type="radio"/> e starts low, then increases, then decreases.		
<input checked="" type="radio"/> e starts high, then decreases, then increases.	✓	1.00
<input type="radio"/> e starts low, then decreases		
<input type="radio"/> e doesn't change.		
Total	1.00 / 1.00	

Question Explanation

If $i=1$, the model will be biased and error will be high. If $i=n$ the resulting model is the same one we started with and the error will be high (overfitting). There is a sweet spot between 1 and n where the error is minimized.

Question 12

A is a users times movie-ratings matrix like the one seen in class. Each column in A represents a movie, and there are 5 movies in total. Recall that a Singular Value Decomposition of a matrix is a multiplication of three matrices: U , Σ and V . The following is such a decomposition for matrix A :

$$\begin{bmatrix} -0.25 & -0.05 \\ -0.5 & -0.1 \\ -0.76 & -0.15 \\ -0.29 & 0.2 \\ -0.03 & 0.26 \\ -0.07 & 0.51 \\ -0.1 & 0.77 \end{bmatrix} \begin{bmatrix} 6.74 & 0 \\ 0 & 5.44 \end{bmatrix} \begin{bmatrix} -0.57 & -0.11 & -0.57 & -0.11 & -0.57 \\ -0.09 & 0.7 & -0.09 & 0.7 & -0.09 \end{bmatrix}$$

What is the cosine similarity between a user with ratings [5,0,0,0,0] and a user with ratings [0,2,0,0,4] using their concept space vectors? (round your answer to two decimals. For example, if your answer is 0.345 the rounded answer is 0.35. If your answer is 0.344, the rounded answer is 0.34.)

You entered:

-4.41

Your Answer

-4.41

Score

✗

0.00

Explanation

Total

0.00 / 1.00

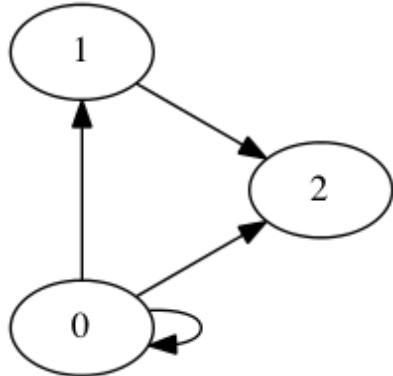
Question Explanation

The concept vector for user with ratings [5,0,0,0,0] is [-2.8508671 , -0.45375114] and the concept vector for user with ratings [0,2,0,0,4] is [-2.50298544, 1.03363303]. The cosine similarity of these two vectors is 0.8528.

Question 13

Recall that the power iteration does $r=X \cdot r$ until converging, where X is a $n \times n$ matrix and n is the number of nodes in the graph.

Using the power iteration notation above, what is matrix X value when solving topic sensitive Pagerank with teleport set $\{0, 1\}$ for the following graph? Use $\beta=0.8$. (Recall that the teleport set contains the destination nodes used when teleporting).



Your Answer**Score****Explanation**



1.00

11/30	1/10	1/10
11/30	1/10	1/10
4/15	8/10	0



1/3	0	0
1/3	0	0
1/3	1	0



8/30	0	0
8/30	0	0
8/30	8/10	0



1/6	1/2	1/2
-----	-----	-----

1/6	1/2	1/2
1/3	1	0

Total 1.00 / 1.00

Question Explanation

Let's call matrix A the graph's adjacency matrix and matrix B the graph representing random jumps to the teleport set adjacency matrix.

A=

1/3	0	0
1/3	0	0
1/3	1	0

B=

1/2	1/2	1/2
1/2	1/2	1/2
0	0	0

$$X = 0.8*M + 0.2*B$$

Question 14

Here are two sets of integers $S = \{1, 2, 3, 4\}$ and $T = \{1, 2, 5, 6, x\}$, where x stands for some integer.

For how many different integer values of x are the Jaccard similarity and the Jaccard distance of S and T the same? (Note: x can be one of 1, 2, 5, or 6, but in that case T , being a set, will contain x only once and thus have four members, not five.)

Your Answer

Score

Explanation

6



1.00

2

4

An infinite number

Total

1.00 / 1.00

Question Explanation

If x is 3 or 4, then the intersection of S and T has size 3 and the union has size 6. Thus, both the similarity and distance of S and T are 1/2. If x is 1, 2, 5, or 6, then the intersection and

union are of sizes 2 and 6 respectively, so the similarity is 1/3 and the distance 2/3. If S is any integer other than 1,...,6, the intersection and union are of sizes 2 and 7 respectively, so again the similarity (2/7) and distance (5/7) are different.

Question 15

Which of the following are advantages of using decision trees? (check all correct options)

Your Answer	Score	Explanation
<input type="checkbox"/> It avoids overfitting	✓ 0.20	
<input type="checkbox"/> It can handle multiple output easily	✗ 0.00	
<input checked="" type="checkbox"/> It can handle categorical input data without any special preprocessing	✓ 0.20	
<input checked="" type="checkbox"/> The resulting model is easy to interpret	✓ 0.20	
<input type="checkbox"/> The training is easy to parallelize	✗ 0.00	
Total	0.60 / 1.00	

Question Explanation

The main warning when using decision trees is to be careful with overfitting. Pruning helps.

Question 16

The hard margin SVM optimization problem is:

$$\begin{aligned} & \text{minimize } \frac{1}{2} \|w\|^2 \\ & \text{s.t. } y_i \cdot (x_i \cdot w + b) \geq 1, \quad \forall i = 1, \dots, n \end{aligned}$$

and the soft margin SVM optimization problem is:

$$\begin{aligned} & \text{minimize } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \\ & \text{s.t. } y_i \cdot (x_i \cdot w + b) \geq 1 - \xi_i, \quad \forall i = 1, \dots, n \\ & \xi_i \geq 0, \quad \forall i = 1, \dots, n \end{aligned}$$

Consider a dataset of points x_1, \dots, x_n with labels $y_1, \dots, y_n \in \{-1, 1\}$, such that the data is separable. We run a soft-margin SVM and a hard-margin SVM, and in each case we obtain parameters w and b . Check the option that is true:

Your Answer	Score	Explanation
<input checked="" type="radio"/> The resulting w and b can be different, and the boundaries can be different.	✓	1.00
<input type="radio"/> The resulting w and b are the same in the two cases, hence boundaries are the same.		
<input type="radio"/> The resulting w and b can be different in the two cases, but the boundaries are the same.		
<input type="radio"/> None of the above.		

Total	1.00 /	
	1.00	

Question Explanation

Think about this example: You are classifying gender based on height. All males are 6' and all females but one are 5' tall. The outlier female is 5.9' tall. The data is linearly separable. A hard margin SVM would set the separation at 5.95'. A soft margin SVM would set it north of 5.5' (how north depends on the number of males and females). The slack variable ξ will have a nonzero value for the outlier female datapoint. Finally, in this particular example the boundary is different.

Question 17

Consider the following MapReduce algorithm. The input is a collection of positive integers. Given integer X, the Map function produces a tuple with key Y and value X for each prime divisor Y of X. For example, if X = 20, there are two key-value pairs: (2,20) and (5,20). The Reduce function, given a key K and list L of values, produces a tuple with key K and value sum(L) i.e., the sum of the values in the list. Given the input 9, 15, 16, 23, 25, 27, 28, 56 which of the following tuples appears in the final output?

Your Answer	Score	Explanation
<input type="radio"/> (7, 51)		
<input type="radio"/> (5, 51)		
<input type="radio"/> (4, 51)		
<input checked="" type="radio"/> (3, 51)	✓	1.00
Total	1.00 / 1.00	

Question 18

Suppose we run K-means clustering over the following set of points in 2-d space using the L_1 distance metric: (1,1), (2,1) (2,2), (3,3), (4,2), (2,4), (4,4). We pick k=2 and the initial centroids are (1,1) and (4,4). Which of these is the centroid of the cluster containing the point (3,3) when the algorithm terminates?

Recall that the L_1 distance between two points is the sum of their distances along each dimension, e.g. the L_1 distance between (1, 2) and (-1, 3) is 3.

Your Answer	Score	Explanation
<input type="radio"/> (5/3, 4/3)		
<input checked="" type="radio"/> (13/4, 13/4)	✓	1.00
<input type="radio"/> (3, 3)		
<input type="radio"/> (4, 4)		
Total	1.00 / 1.00	

Question 19

In an implementation of the Bradley-Fayyad-Reina (BFR) algorithm over a 3-dimensional data set, the discard set for a cluster is summarized by the following parameters:

$$N = 1000$$

$$\text{SUM} = (-323, 1066, 1776)$$

$$\text{SUMSQ} = (412, 1500, 3500)$$

Which of the following choices is closest to the Mahalanobis distance of the point (0,0,0) from the centroid of this cluster?

Your Answer	Score	Explanation
<input checked="" type="radio"/> 3.55	✓	1.00
<input type="radio"/> 1.55		

2.55 4.55

Total 1.00 / 1.00

Question 20

Consider an execution of the BALANCE algorithm with 4 advertisers, A1, A2, A3, A4, and 4 kinds of queries, Q1, Q2, Q3, Q4. Advertiser A1 bids on queries Q1 and Q2; A2 bids on queries Q2 and Q3; A3 on queries Q3 and Q4; and A4 on queries Q1 and Q4. All bids are equal to 1, and all clickthrough rates are equal. All advertisers have a budget of 3, and ties are broken in favor of the advertiser with the lower index (e.g., A1 beats A2). Queries appear in the following order:

Q1, Q2, Q3, Q3, Q1, Q2, Q3, Q1, Q4, Q1

Which advertiser's budget is exhausted first?

Your Answer**Score****Explanation** A4 A1 A2 A3

1.00

Total

1.00 / 1.00

Question 21

Consider the bipartite graph with the following edges (you might want to draw a picture):

(a,1), (a,3), (b,1), (b,2), (b,4), (c,2), (d,1), (d,4)

Which of the following edges appears in NO perfect matching?

Your Answer**Score****Explanation**

(d, 4) (c, 2) (b, 4) (a, 1)

1.00

Total

1.00 / 1.00

Question 22

The Utility Matrix below captures the ratings of 5 users (A,B,C,D,E) for 5 movies (P,Q,R,S,T).

Each known rating is a number between 1 and 5, and blanks represent unknown ratings. What is the Pearson Correlation (also known as the Centered Cosine) between users B and D?

	P	Q	R	S	T
A	2		4		
B		3	1	2	
C	5			5	
D		4	3		2
E	4			5	1

Your Answer

Score

Explanation 0.5

1.00

 0.23

 0.96

 0.74

Total

1.00 / 1.00

Question 23

The Utility Matrix below captures the ratings of 5 users (A,B,C,D,E) for 5 movies (P,Q,R,S,T).

Each known rating is a number between 1 and 5, and blanks represent unknown ratings. Let

(U,M) denote the rating of movie M by user U. We evaluate a Recommender System by withholding the ratings (A,P), (B,Q), and (C,S). The recommender system estimates (A,P)=1, (B,Q)=4, and (C,S)=5. What is the RMSE of the Recommender System, rounded to 2 decimal places?

	P	Q	R	S	T
A	2		4		
B		3	1	2	
C	5			5	
D		4	3		2
E	4			5	1

Your Answer

Score

Explanation

2.36

0.82



1.00

1.44

0.0

Total

1.00 / 1.00

Question 24

We are going to perform a hierarchical (agglomerative) clustering on the four strings {he, she, her, their}, using edit distance (just insertions and deletions; no mutations of characters). Initially, each string is in a cluster by itself. The distance between two clusters is the **minimum** edit distance between two strings, one chosen from each of the two clusters. When we complete the hierarchical clustering, there is one cluster containing all four strings, and we performed three mergers of clusters to get to that point. For each of the three mergers there was a distance between the merged clusters. What is the sum of those three distances?

Your Answer

Score

Explanation

3

4

1.00

5

- It depends on how we break ties when there are two pairs of clusters at the same distance.

Total	1.00 /
	1.00

Question Explanation

In the first merger we merge "he" with either "she" or "her", since both are at distance 1. In the second merger, we merge whichever of "she" or "her" remains, with the cluster of size 2. Again, the distance is 1. Then, we merge "their" with the cluster of size 3. Since "their" is distance 2 from "her" and at higher distance from "he" and "she", the distance between the last two clusters is 2. The answer is thus $1+1+2 = 4$.

Feedback — Final (Advanced)

[Help](#)

You submitted this exam on **Sat 29 Nov 2014 1:26 PM PST**. You got a score of **11.00** out of **15.00**.

This is the Advanced Final Exam for the MMDS Course. Your score on this exam counts only for a SoA "With Distinction.". You must submit your work within 2 hours of opening. The exam is open-book; any inanimate source may be used. There is no penalty for a wrong answer (compared with no answer), so feel free to guess.

Question 1

Suppose ABCD is **not** a frequent itemset, while ABC and ACD are frequent itemsets. Which of the following is **definitely** true?

Your Answer	Score	Explanation
-------------	-------	-------------

- ABD is a frequent itemset.
- BC is a frequent itemset. ✓ 1.00
- ABCD is in the negative border.
- ABCE is not a frequent itemset.

Total	1.00 / 1.00
-------	-------------

Question 2

Suppose we are representing sets by strings and indexing the strings according to both the symbol and its position within the prefix. We want to find strings within Jaccard distance at most 0.2 (i.e., similarity at least 0.8), and we are given a probe string of length 24. Into how many buckets must we look?

Your Answer	Score	Explanation
-------------	-------	-------------

- 18

15 21

1.00

 5

Total

1.00 / 1.00

Question Explanation

Here is the code:

```
for (i=1; i
```

Since $J = 0.2$ and $L = 24$, the loop on i ranges from 1 to 5, and the loop on j ranges from 1 to $(33-5i)/4$. Thus, for $i = 1, 2, 3, 4$, and 5, the numbers of values of j are 7, 5, 4, 3, and 2, respectively, which sums to 21.

Question 3

In the following question we consider an example of the implementation of the PCY algorithm. All numbers should be treated as decimal; e.g., "one million" is 1,000,000, NOT $2^{20} = 1,048,576$. All integers (item counts and bucket counts) require 4 bytes.

We have one billion bytes of main memory available for the first pass. There are 100,000,000 items, and also 100,000,000 baskets, each of which contains exactly 10 items. Say that PCY is **effective** if the average count of a bucket is at most half the support value. For the given data, what is the smallest support value for which PCY will be effective?

Your Answer

Score

Explanation

 6 60

1.00

 600 6000

Total

1.00 / 1.00

Question Explanation

The item counts take up 4×10^8 bytes, leaving 6×10^8 bytes for buckets. We therefore can use 1.5×10^8 buckets. Each basket generates 45 pairs, so we have 4.5×10^9 pairs to divide among

the buckets. An average bucket therefore gets a count of 30. The minimum support for which PCY will be "effective" is twice that, or 60.

Question 4

Suppose we want to represent the multiplication of two 10-by-10 matrices as a "problem" in the sense used for our discussion of the theory of MapReduce algorithms. How many pairs are in the input-output mapping?

Your Answer	Score	Explanation
<input type="radio"/> 1000		
<input type="radio"/> 20		
<input checked="" type="radio"/> 2000	✓ 1.00	
<input type="radio"/> 100		
Total	1.00 / 1.00	

Question Explanation

There are 100 outputs, each of which depends on 20 inputs (10 in the appropriate row of the first matrix and 10 in the appropriate column of the second matrix). Thus, there are 2000 input-output pairs.

Question 5

The "all-triples" problem is described by n inputs, $(n \text{ choose } 3)$ outputs, and an input-output mapping where each output is connected to a different set of three inputs. Suppose q is the reducer size. Which of the following functions of n and q approximates, to within a constant factor, the lowest possible replication rate for a mapping schema that solves this problem?

Your Answer	Score	Explanation
<input type="radio"/> $r = n^2/q^2$		
<input checked="" type="radio"/> $r = n^3/q$	✗ 0.00	
<input type="radio"/> $r = n^2/q$		
<input type="radio"/> $r = n/q^2$		
Total	0.00 / 1.00	

Question Explanation

As we did in the lectures, we'll assume without proof that the minimum replication rate is obtained when you use the fewest possible number of reducers, each of which gets q inputs. There are approximately $n^3/6$ outputs, and a reducer with q inputs can only cover $q^3/6$ of them, so we need approximately n^3/q^3 reducers. Each reducer gets q inputs, so the total communication from mappers to reducers is n^3/q^2 . The replication rate is this function divided by n , or $r \geq n^2/q^2$.

We can design a MapReduce algorithm that matches this lower bound to within a constant factor. Divide the n inputs into $3n/q$ groups of $q/3$ inputs each. Each reducer gets a different set of three groups, so the number of reducers we need is $(3n/q)$ choose 3, or approximately $(9/2)n^3/q^3$. Since each reducer gets 3 of the $3n/q$ groups, the fraction of reducers that any one group is sent to is q/n . Thus, the members of any group are sent to approximately $(9/2)n^2/q^2$ reducers, and this replication rate is about 4.5 times the lower bound. Thus, $r = n^2/q^2$ is, to within a constant factor, the best possible replication rate.

Question 6

Suppose we are running the DGIM algorithm (approximate counting of 1's in a window. At time t , the list of bucket sizes being maintained is 8,4,4,2,1,1. At times $t+1$, $t+2$, and $t+3$, 1's arrive on the input. Assuming no buckets are deleted because they fall outside the window, what are the numbers of buckets after each of the times $t+1$, $t+2$, and $t+3$?

Your Answer	Score	Explanation
<input checked="" type="radio"/> time $t+1$: 6; time $t+2$: 7; time $t+3$: 5.	✓ 1.00	
<input type="radio"/> time $t+1$: 5; time $t+2$: 6; time $t+3$: 5		
<input type="radio"/> time $t+1$: 7; time $t+2$: 8; time $t+3$: 9		
<input type="radio"/> time $t+1$: 7; time $t+2$: 6; time $t+3$: 7.		
Total	1.00 / 1.00	

Question Explanation

At time $t+1$, we receive a third 1, so the first two 1's are combined into a 2, leaving 8,4,4,2,2,1 (six buckets). At time $t+2$, the 1 that arrives forms a bucket of its own, so we have 8,4,4,2,2,1,1 (seven buckets). When the third 1 arrives at time $t+3$, we must combine the first two 1's into a 2. We now have three 2's, so the first two of these are combined into a 4. We have three 4's so the first two are combined into an 8. The resulting list of buckets is 8,8,4,2,1 (five buckets).

Question 7

Apply the HITS algorithm to a network with four pages (nodes) A, B, C, and D, arranged in a chain:

A-->B-->C-->D

Compute the hubbiness and authority of each of these pages (scale doesn't matter, because you only have to identify pages with the same hubbiness or the same authority). Which of the following is FALSE.

Your Answer	Score	Explanation
<input type="radio"/> A and B have the same hubbiness.		
<input type="radio"/> B and C have the same authority.		
<input type="radio"/> B and C have the same hubbiness.		
<input checked="" type="radio"/> A and B have the same authority.	✓ 1.00	
Total	1.00 / 1.00	

Question Explanation

Suppose we start with each page having authority 1. Then the hubbiness of A, B, and C will be 1 and the hubbiness of D will be 0 at the next iteration. Then, from these hubbiness values, we can compute new authorities: B, C, and D have authority 1 and A has authority 0. From these new authorities, we again compute the hubbiness and get the same answer as at the first step: A, B, and C have hubbiness 1 and D has hubbiness 0. We have thus converged, and conclude that A, B, and C have the same hubbiness, while B,C, and D have the same authority.

Question 8

Let G be the complete graph on five nodes (i.e., there is an edge in G between every pair of distinct nodes). What is the sum of the **squares** of the elements of the Laplacian matrix for G?

Your Answer	Score	Explanation
<input type="radio"/> 40		
<input type="radio"/> 0		
<input checked="" type="radio"/> 100	✓ 1.00	
<input type="radio"/> 20		
Total	1.00 / 1.00	

Question Explanation

The Laplacian matrix has 4's in the five diagonal entries and -1's in the other 20 positions. Thus, the sum of the squares of the elements is $20*(-1)^2 + 5*(4)^2 = 20*1 + 5*16 = 100$.

Question 9

Note: This problem is similar to one on the Basic Final, but involves a combiner.

Consider the following MapReduce algorithm. The input is a collection of positive integers.

Given integer X, the Map function produces a tuple with key Y and value X for each prime divisor Y of X. For example, if X = 20, there are two key-value pairs: (2,20) and (5,20). The Reduce function, given a key K and list L of values, produces a tuple with key K and value $\text{sum}(L)$ i.e., the sum of the values in the list.

Suppose we process the input 9, 15, 16, 23, 25, 27, 28, 56, using a Combiner. There are 4 Map tasks and 1 Reduce task. The first Map task processes the first two inputs, the second the next two, and so on. How many input tuples does the Reduce task receive?

Your Answer	Score	Explanation
<input type="radio"/> 6		
<input checked="" type="radio"/> 8	✓	1.00
<input type="radio"/> 3		
<input type="radio"/> 11		
Total	1.00 / 1.00	

Question Explanation

When we use a Combiner, each Map task sends to the Reduce task a single tuple for each distinct key it produces. Each map task produces 2 distinct keys. Task 1 {3,5}; Task 2 {2,23}; Task 3 {3,5}; Task 4 {2,7}. So the number of tuples sent to the Reduce task is 8.

Question 10

Consider an AdWords scenario with 4 advertisers competing for the same query Q, all with the same budget of \$100 and the same clickthrough rate. The table below shows the bid and the dollars spent by each advertiser until this point. Suppose we use Generalized BALANCE, and

show one ad for each query. Which advertiser do we pick the next time query Q comes up?

Advertiser	Bid	Spend
A	\$1	\$20
B	\$2	\$40
C	\$3	\$60
D	\$4	\$80

Your Answer**Score****Explanation** C

1.00

 B A D

Total

1.00 / 1.00

Question 11

Suppose we wish to estimate the rating of movie M by user U using item-Item Collaborative Filtering, but there are no movies really similar to movie M. The average of all ratings is 3.5, user U's average rating is 3.1, and movie M's average rating is 4.3. What is our best guess for the rating of movie M by user U using a global baseline estimate?

Your Answer**Score****Explanation** 4.7 3.9 3.5 4.3

0.00

Total

0.00 / 1.00

Question Explanation

Global Baseline Estimate = Global Average + Movie Bias + User Bias = $3.5 + (4.3 - 3.5) + (3.1 - 3.5) = 3.9$

Question 12

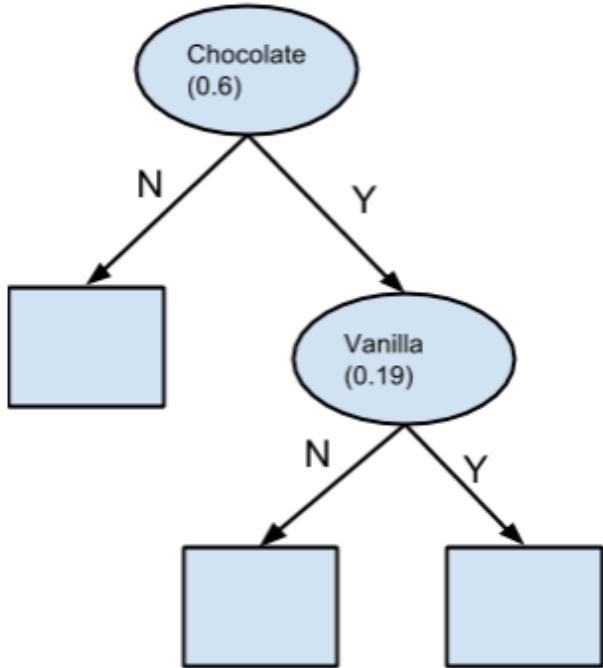
The table below shows data from ten people showing whether they like four different ice cream flavors.

Chocolate	Vanilla	Strawberry	Peanut
Y	N	Y	Y
N	Y	Y	N
N	N	N	N
Y	Y	Y	Y
Y	Y	N	Y
N	N	N	N
Y	Y	Y	Y
N	Y	N	N
Y	N	Y	N
Y	N	Y	Y

Fit a decision tree that predicts whether somebody would like Peanut ice cream based on whether she liked the other three flavors. Use Information gain as the measure to make the splits. What is the order of splits?

Your Answer	Score	Explanation
<input checked="" type="radio"/> Only Chocolate	✗	0.00
<input type="radio"/> Vanilla->Chocolate		
<input type="radio"/> Chocolate->Vanilla		
<input type="radio"/> Strawberry->Chocolate->Vanilla		
<input type="radio"/> Chocolate->Strawberry->Vanilla		
Total	0.00 / 1.00	

Question Explanation



Question 13

For an unknown graph with 3 nodes, r_1 is the topic sensitive PageRank using teleport set $\{0, 1\}$ and r_2 is the topic sensitive PageRank using teleport set $\{1\}$. What's the value of the topic sensitive PageRank vector when using teleport set $\{0\}$?

Your Answer

Score

Explanation

$2r_1 - r_2$

$r_2 - r_1$

$r_1 - 2r_2$

$r_1 - r_2$



0.00

Total

0.00 / 1.00

Question 14

A is a users times movie-ratings matrix like the one seen in class. Each column in A represents a movie, and there are 5 movies in total. Recall that a Singular Value Decomposition of a matrix is a multiplication of three matrices: U , Σ and V . Is the following such a decomposition for matrix A :

A:

$$\begin{bmatrix} -0.25 & -0.05 \\ -0.5 & -0.1 \\ -0.76 & -0.15 \\ -0.29 & 0.2 \\ -0.03 & 0.26 \\ -0.07 & 0.51 \\ -0.1 & 0.77 \end{bmatrix} \begin{bmatrix} 6.74 & 0 \\ 0 & 5.44 \end{bmatrix} \begin{bmatrix} -0.57 & -0.11 & -0.57 & -0.11 & -0.57 \\ -0.09 & 0.7 & -0.09 & 0.7 & -0.09 \end{bmatrix}$$

If we get three new users with the following rating vectors: User 1: [5,0,0,0,0] User 2: [0,5,0,0,0] User 3: [0,0,0,0,4] If for advertising purposes we want to cluster these three customers into two clusters using the movie concepts as features. How would you cluster them? (use cosine distance).

Your Answer	Score	Explanation
<input type="radio"/> [1,2] and [3]		
<input checked="" type="radio"/> [1,3] and [2]	✓ 1.00	
<input type="radio"/> [1] and [2,3]		
<input type="radio"/> [1,2,3]		
<input type="radio"/> [1] and [2] and [3]		
Total	1.00 / 1.00	

Question Explanation

The projected vectors for each user are:

User 1: [-2.8508671 -0.45375114]
User 2: [-0.55572939 3.49158486]
User 3: [-2.28069368 -0.36300092]

The cosine distance between User 1 and User 2 is: 1.0
The cosine distance between User 1 and User 3 is: 1.11022302463e-16
The cosine distance between User 2 and User 3 is: 1.0

Clearly user 1 and user 3 belong to the same cluster.

Question 15

The soft margin SVM optimization problem is:

$$\begin{aligned} & \text{minimize}_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi_i \\ & \text{s.t. } y_i \cdot (x_i \cdot w + b) \geq 1 - \xi_i, \quad \forall i = 1, \dots, n \\ & \quad \xi_i \geq 0, \quad \forall i = 1, \dots, n \end{aligned}$$

If for some i we have $\xi_i = 0$, this indicates that the point x_i is (check the true option):

Your Answer	Score	Explanation
<input checked="" type="radio"/> Correctly classified		1.00
<input type="radio"/> Exactly in the decision boundary		
<input type="radio"/> A support vector		
<input type="radio"/> Incorrectly classified		
Total	1.00 / 1.00	

Feedback — Week6A (Advanced)

[Help](#)

You submitted this quiz on **Sat 1 Nov 2014 4:01 AM PDT**. You got a score of **4.00** out of **4.00**.

Question 1

Using the matrix-vector multiplication described in Section 2.3.1, applied to the matrix and vector:

1	2	3	4	1
5	6	7	8	2
9	10	11	12	3
13	14	15	16	4

apply the Map function to this matrix and vector. Then, identify in the list below, one of the key-value pairs that are output of Map.

Your Answer	Score	Explanation
<input checked="" type="radio"/> (4,28)	✓ 1.00	
<input type="radio"/> (2,20)		
<input type="radio"/> (3,110)		
<input type="radio"/> (2,70)		
Total	1.00 / 1.00	

Question 2

Suppose we use the algorithm of Section 2.3.10 to compute the product of matrices M and N. Let M have x rows and y columns, while N has y rows and z columns. As a function of x, y, and z, express the answers to the following questions:

1. The output of the Map function has how many different keys? How many key-value pairs are there with each key? How many key-value pairs are there in all?
2. The input to the Reduce function has how many keys? What is the length of the value (a list)

associated with each key?

Then, identify the true statement in the list below.

Your Answer	Score	Explanation
<input type="radio"/> The output of the Map function has xyz pairs.		
<input type="radio"/> The input to the Reduce function has pairs with lists of length xz.		
<input type="radio"/> The input to the Reduce function has pairs with lists of length 2xz.		
<input checked="" type="radio"/> The output of the Map function has xz different keys.	✓ 1.00	
Total	1.00 / 1.00	

Question 3

Suppose we use the two-stage algorithm of Section 2.3.9 to compute the product of matrices M and N. Let M have x rows and y columns, while N has y rows and z columns. As a function of x, y, and z, express the answers to the following questions:

1. The output of the first Map function has how many different keys? How many key-value pairs are there with each key? How many key-value pairs are there in all?
2. The output of the first Reduce function has how many keys? What is the length of the value (a list) associated with each key?
3. The output of the second Map function has how many different keys? How many key-value pairs are there with each key? How many key-value pairs are there in all?

Then, identify the true statement in the list below.

Your Answer	Score	Explanation
<input checked="" type="radio"/> The output of the second Map function has xyz pairs.	✓ 1.00	
<input type="radio"/> The output of the second Map function has xy different keys.		
<input type="radio"/> The output of the first Map function has xz pairs with each key.		
<input type="radio"/> The output of the first Map function has z pairs with each		

key.

Total	1.00 / 1.00
-------	----------------

Question 4

Suppose we have the following relations:

R		S	
A	B	B	C
0	1	0	1
1	2	1	2
2	3	2	3

and we take their natural join by the algorithm of Section 2.3.7. Apply the Map function to the tuples of these relations. Then, construct the elements that are input to the Reduce function. Identify one of these elements in the list below.

Your Answer	Score	Explanation
<input checked="" type="radio"/> (3, [(R,2)])	✓ 1.00	
<input type="radio"/> (0,(S,1))		
<input type="radio"/> (1,(S,2))		
<input type="radio"/> (1, [(R,0)])		
Total	1.00 / 1.00	

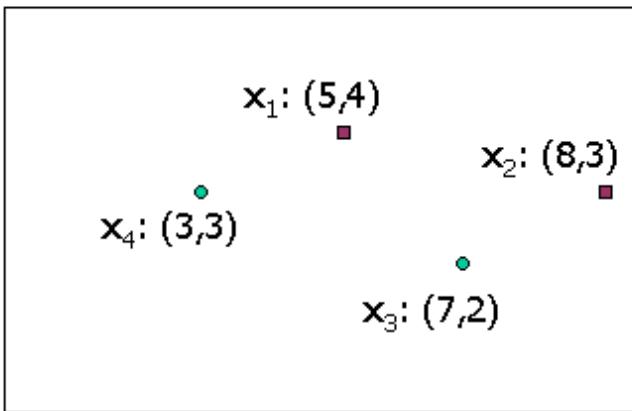
Feedback — Week6B (Basic)

[Help](#)

You submitted this quiz on **Sat 1 Nov 2014 2:38 AM PDT**. You got a score of **3.00** out of **3.00**.

Question 1

The figure below shows two positive points (purple squares) and two negative points (green circles):



That is, the training data set consists of:

$$(x_1, y_1) = ((5,4), +1)$$

$$(x_2, y_2) = ((8,3), +1)$$

$$(x_3, y_3) = ((7,2), -1)$$

$$(x_4, y_4) = ((3,3), -1)$$

Our goal is to find the maximum-margin linear classifier for this data. In easy cases, the shortest line between a positive and negative point has a perpendicular bisector that separates the points. If so, the perpendicular bisector is surely the maximum-margin separator. Alas, in this case, the closest pair of positive and negative points, x_2 and x_3 , have a perpendicular bisector that misclassifies x_1 as negative, so that won't work.

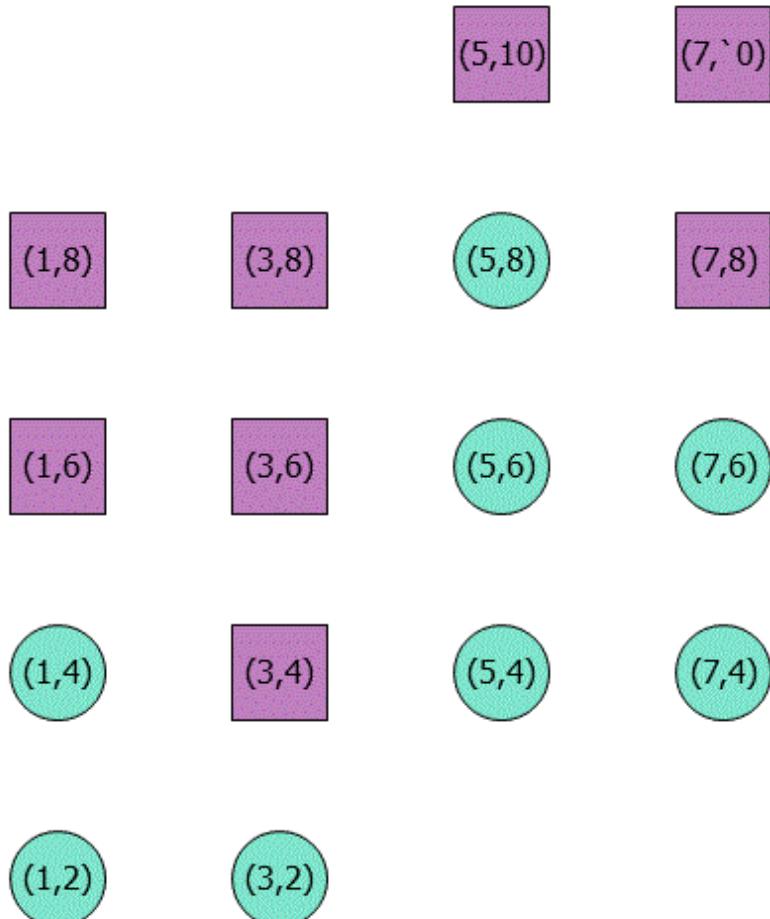
The next-best possibility is that we can find a pair of points on one side (i.e., either two positive or two negative points) such that a line parallel to the line through these points is the maximum-margin separator. In these cases, the limit to how far from the two points the parallel line can get is determined by the closest (to the line between the two points) of the points on the other side. For our simple data set, this situation holds.

Consider all possibilities for boundaries of this type, and express the boundary as $\mathbf{w} \cdot \mathbf{x} + b = 0$, such that $\mathbf{w} \cdot \mathbf{x} + b \geq 1$ for positive points \mathbf{x} and $\mathbf{w} \cdot \mathbf{x} + b \leq -1$ for negative points \mathbf{x} . Assuming that $\mathbf{w} = (w_1, w_2)$, identify in the list below the true statement about one of w_1 , w_2 , and b .

Your Answer	Score	Explanation
<input type="radio"/> $w_1 = 2/5$		
<input type="radio"/> $w_1 = 1$		
<input type="radio"/> $b = -5$		
<input checked="" type="radio"/> $b = -15/2$	✓ 1.00	
Total	1.00 / 1.00	

Question 2

Consider the following training set of 16 points. The eight purple squares are positive examples, and the eight green circles are negative examples.



We propose to use the diagonal line with slope +1 and intercept +2 as a decision boundary,

with positive examples above and negative examples below. However, like any linear boundary for this training set, some examples are misclassified. We can measure the goodness of the boundary by computing all the slack variables that exceed 0, and then using them in one of several objective functions. In this problem, we shall only concern ourselves with computing the slack variables, not an objective function.

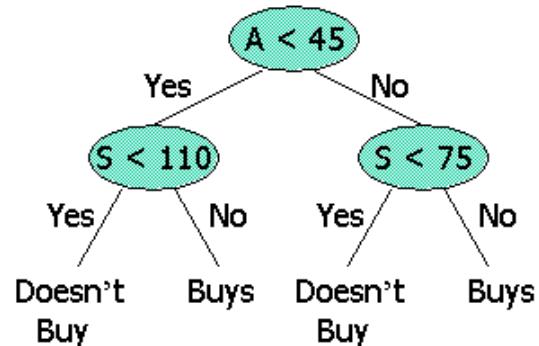
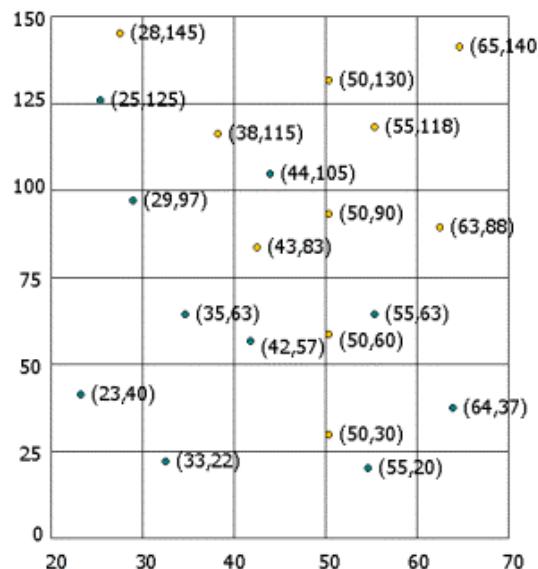
To be specific, suppose the boundary is written in the form $\mathbf{w} \cdot \mathbf{x} + b = 0$, where $\mathbf{w} = (-1, 1)$ and $b = -2$. Note that we can scale the three numbers involved as we wish, and so doing changes the margin around the boundary. However, we want to consider this specific boundary and margin.

Determine the slack for each of the 16 points. Then, identify the correct statement in the list below.

Your Answer	Score	Explanation
<input type="radio"/> The slack for (7,10) is 2.		
<input type="radio"/> The slack for (1,4) is 0.		
<input checked="" type="radio"/> The slack for (1,4) is 2.	✓ 1.00	
<input type="radio"/> The slack for (5,8) is 0.		
Total	1.00 / 1.00	

Question 3

Below we see a set of 20 points and a decision tree for classifying the points.



To be precise, the 20 points represent (Age,Salary) pairs of people who do or do not buy gold jewelry. Age (abbreviated A in the decision tree) is the x-axis, and Salary (S in the tree) is the y-axis. Those that do are represented by gold points, and those that do not by green points.

The 10 points of gold-jewelry buyers are:

(28,145), (38,115), (43,83), (50,130), (50,90), (50,60), (50,30), (55,118), (63,88), and (65,140).

The 10 points of those that do not buy gold jewelry are:

(23,40), (25,125), (29,97), (33,22), (35,63), (42,57), (44, 105), (55,63), (55,20), and (64,37).

Some of these points are correctly classified by the decision tree and some are not. Determine the classification of each point, and then indicate in the list below the point that is misclassified.

Your Answer	Score	Explanation
<input type="radio"/> (63,88)		
<input type="radio"/> (29,97)		
<input checked="" type="radio"/> (50,30)	✓	1.00
<input type="radio"/> (42,57)		
Total	1.00 / 1.00	

Feedback — Week7A Advanced

[Help](#)

You submitted this quiz on **Sat 22 Nov 2014 3:32 AM PST**. You got a score of **4.00** out of **4.00**.

Question 1

Suppose we have an LSH family h of $(d_1, d_2, .6, .4)$ hash functions. We can use three functions from h and the AND-construction to form a (d_1, d_2, w, x) family, and we can use two functions from h and the OR-construction to form a (d_1, d_2, y, z) family. Calculate w , x , y , and z , and then identify the correct value of one of these in the list below.

Your Answer	Score	Explanation
<input type="radio"/> w=.36		
<input type="radio"/> w=.064		
<input checked="" type="radio"/> w=.216	✓ 1.00	
<input type="radio"/> w=.936		
Total	1.00 / 1.00	

Question Explanation

When we use the AND-construction with three hash functions, we cube the probabilities associated with h . Thus, $w=.216$ and $x=.064$. To get the probabilities associated with the OR-construction on two hash functions, we take each probability associated with h , subtract it from 1, square the result, and subtract that from 1. Thus, $.6$ becomes $1-(1-.6)^2 = .84$, and $.4$ becomes $1-(1-.4)^2 = .64$.

Question 2

Here are eight strings that represent sets:

$s_1 = abcef$

$s_2 = acdeg$

$s_3 = bcdefg$

$s_4 = adfg$

$s_5 = bcdfgh$

$s_6 = bceg$

$s_7 = cdfg$

$s_8 = abcd$

Suppose our upper limit on Jaccard distance is 0.2, and we use the indexing scheme of Section 3.9.4 based on symbols appearing in the prefix (no position or length information). For each of s_1 , s_3 , and s_6 , determine how many *other* strings that string will be compared with, if it is used as the probe string. Then, identify the true count from the list below.

Your Answer	Score	Explanation
<input checked="" type="radio"/> s_3 is compared with 5 other strings.	✓ 1.00	
<input type="radio"/> s_1 is compared with 5 other strings.		
<input type="radio"/> s_1 is compared with 7 other strings.		
<input type="radio"/> s_3 is compared with 6 other strings.		
Total	1.00 / 1.00	

Question Explanation

First, we index a string of length L on the symbols appearing in its prefix of length $\text{floor}(0.2L+1)$. Thus, strings of length 5 and 6 are indexed on their first two symbols, while strings of length 4 are indexed on their first symbol only. Thus, the index for a consists of $\{s_1, s_2, s_4, s_8\}$; the index for b consists of $\{s_1, s_3, s_5, s_6\}$, the index for c consists of $\{s_2, s_3, s_5, s_7\}$, and no other symbol is indexed at all.

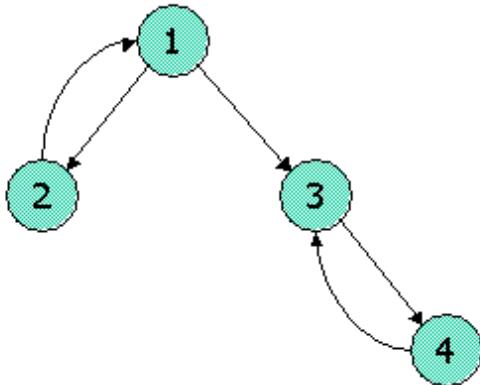
For s_1 , we examine the indexes for a and b , which contains all strings but s_7 . Thus, s_1 is compared with 6 other strings.

For s_3 , we examine the indexes for b and c , which together contain s_1, s_2, s_3, s_5, s_6 , and s_7 . Thus, s_3 is compared with five other strings.

For s_6 , we examine only the index for b . Thus, s_6 is compared only with the three other strings s_1, s_3 , and s_5 .

Question 3

Consider the link graph



First, construct the L , the link matrix, as discussed in Section 5.5 on the HITS algorithm. Then do the following:

1. Start by assuming the hubbiness of each node is 1; that is, the vector h is (the transpose of) $[1, 1, 1, 1]$.
2. Compute an estimate of the authority vector $a = L^T h$.
3. Normalize a by dividing all values so the largest value is 1.
4. Compute an estimate of the hubbiness vector $h = La$.
5. Normalize h by dividing all values so the largest value is 1.
6. Repeat steps 2-5.

Now, identify in the list below the true statement about the final estimates.

Your Answer	Score	Explanation
<input type="radio"/> The final estimate of the hubbiness of 3 is 1.		
<input type="radio"/> The final estimate of the authority of 2 is $1/8$.		
<input checked="" type="radio"/> The final estimate of the hubbiness of 1 is 1.	✓ 1.00	
<input type="radio"/> The final estimate of the authority of 4 is $3/5$.		
Total	1.00 / 1.00	

Question Explanation

Here is the matrix L :

$$\begin{matrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{matrix}$$

In what follows, all vectors will be written as rows, i.e., in transposed form. We start with $h = [1, 1, 1, 1]$ and compute $L^T h = [1, 1, 2, 1]$. Since the largest value is 2, we divide all values by 2, giving us the first estimate $a = [1/2, 1/2, 1, 1/2]$.

Next, we compute $La = [3/2, 1/2, 1/2, 1]$ and normalize by multiplying by $2/3$ to get $h = [1, 1/3, 1/3, 2/3]$.

The next calculation of \mathbf{a} from the estimate of \mathbf{h} gives $L^T \mathbf{h} = [1/3, 1/5, 3/3, 1/3]$, and normalizing gives $\mathbf{a} = [1/5, 3/5, 1, 1/5]$.

For the final estimate of \mathbf{h} we compute $L\mathbf{a} = [8/5, 1/5, 1/5, 1]$, which after normalizing gives $\mathbf{h} = [1, 1/8, 1/8, 5/8]$.

Question 4

Consider an implementation of the Block-Stripe Algorithm discussed in Section 5.2 to compute page rank on a graph of N nodes (i.e., Web pages). Suppose each page has, on average, 20 links, and we divide the new rank vector into k blocks (and correspondingly, the matrix M into k stripes). Each stripe of M has one line per "source" web page, in the format:

[source_id, degree, m, dest_1, ..., dest_m]

Notice that we had to add an additional entry, m , to denote the number of destination nodes in this stripe, which of course is no more than the degree of the node. Assume that all entries (scores, degrees, identifiers,...) are encoded using 4 bytes.

There is an additional detail we need to account for, namely, **locality** of links. As a very simple model, assume that we divide web pages into two disjoint sets:

1. **Introvert** pages, which link only to other pages within the same host as themselves.
2. **Extrovert** pages, which have links to pages across several hosts.

Assume a fraction x of pages (0 < x < 1). Construct a formula that counts the amount of I/O per page rank iteration in terms of N , x , and k . The 4-tuples below list combinations of N , k , x , and I/O (in bytes). Pick the correct combination.

Note. There are some additional optimizations one can think of, such as striping the old score vector, encoding introvert and extrovert pages using different schemes, etc. For the purposes of working this problem, assume we don't do any optimizations beyond the block-stripe algorithm discussed in class.

Your Answer	Score	Explanation
<input type="radio"/> N = 1 billion, k = 2, x = 0.75, 112GB		
<input checked="" type="radio"/> N = 1 billion, k = 2, x = 0.75, 107GB	✓ 1.00	
<input type="radio"/> N = 1 billion, k = 2, x = 0.75, 116GB		
<input type="radio"/> N = 1 billion, k = 2, x = 0.5, 112GB		
Total	1.00 / 1.00	

Question Explanation

The number of bytes involved in reading the old pagerank vector and writing the new pagerank vector to disk = $4(k+1)N$ For the M matrix: - The introvert pages will appear xN times and each row will have on average 23 entries (3 metadata and 20 destination links). Total number of bytes read = $4 \cdot 23 \cdot xN$ - The extrovert pages will appear $(1-x)kN$ times and each row will have 3 (metadata) + $20/k$ (destination links) entries on average. Total number of bytes read = $4 \cdot (3+20/k) \cdot (1-x)kN$ Total I/O per pagerank iteration (in GB, where $1GB \sim 10^9 = N$ bytes) = $4 \cdot [(k+1)N + 23xN + (3k+20)(1-x)N] / N = 4[(k+1) + 23x + (3k+20)(1-x)] = 4[21 + k + 3(x + (1-x)k)]$

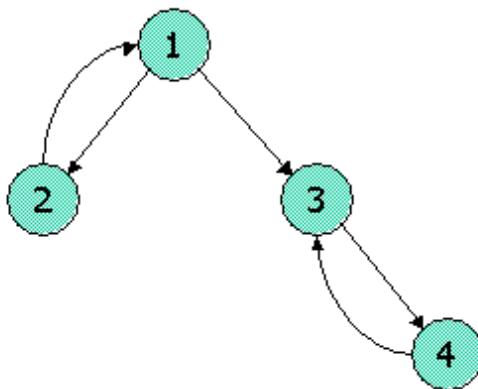
Feedback — Week7B Basic

[Help](#)

You submitted this quiz on **Sat 8 Nov 2014 10:09 PM PST**. You got a score of **2.00 out of 2.00**.

Question 1

Compute the Topic-Specific PageRank for the following link topology. Assume that pages selected for the teleport set are nodes 1 and 2 and that in the teleport set, the weight assigned for node 1 is twice that of node 2. Assume further that the teleport probability, $(1 - \beta)$, is 0.3. Which of the following statements is correct?



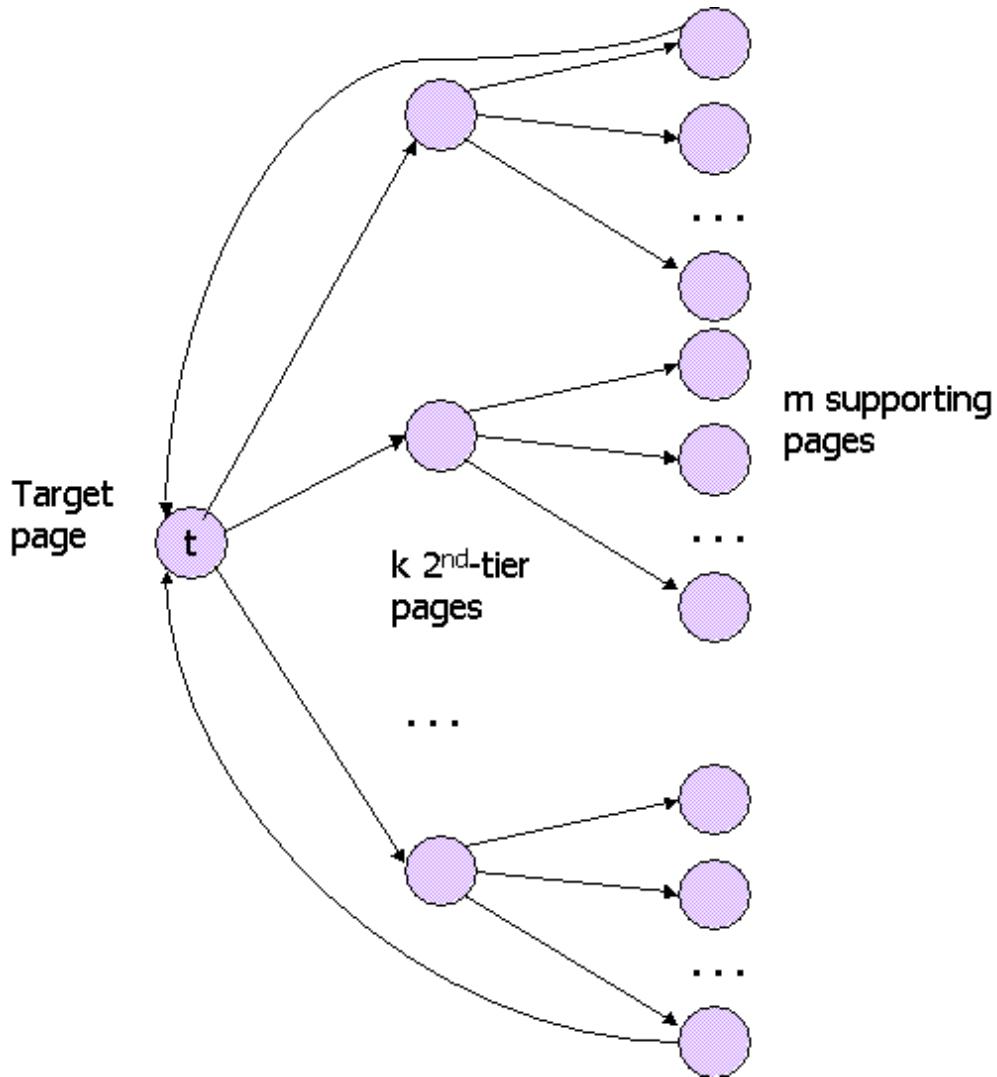
Your Answer	Score	Explanation
<input type="radio"/> TSPR(1) = .4236		
<input checked="" type="radio"/> TSPR(3) = .2454	✓ 1.00	
<input type="radio"/> TSPR(4) = .6680		
<input type="radio"/> TSPR(4) = .4787		
Total	1.00 / 1.00	

Question Explanation

"the weight assigned for node 1 is twice that of node 2" means that given a random walker and its current position, its teleport probability to node 1 is twice that to node 2.

Question 2

The spam-farm architecture described in Section 5.4.1 suffers from the problem that the target page has many links --- one to each supporting page. To avoid that problem, the spammer could use the architecture shown below:



There, k "second-tier" nodes act as intermediaries. The target page t has only to link to the k second-tier pages, and each of those pages links to m/k of the m supporting pages. Each of the supporting pages links only to t (although most of these links are not shown). Suppose the taxation parameter is $\beta = 0.85$, and x is the amount of PageRank supplied from outside to the target page. Let n be the total number of pages in the Web. Finally, let y be the PageRank of target page t . If we compute the formula for y in terms of k , m , and n , we get a formula with the form

$$y = ax + \frac{bm}{n} + \frac{ck}{n}$$

Note: To arrive at this form, it is necessary at the last step to drop a low-order term that is a fraction of $1/n$. Determine coefficients a , b , and c , remembering that β is fixed at 0.85. Then, identify the value, correct to two decimal places, for one of these coefficients.

Your Answer	Score	Explanation
<input checked="" type="radio"/> c = 0.28	✓ 1.00	
<input type="radio"/> a = 1.98		
<input type="radio"/> c = 0.33		
<input type="radio"/> a = 3.60		
Total	1.00 / 1.00	

Question Explanation

Let w be the PageRank of each of the second-tier pages, and let z be the PageRank of each of the supporting pages. Then the equations relating y , w , and z are:

$$y = x + \beta z m + (1-\beta)/n$$

$$w = \beta y/k + (1-\beta)/n$$

$$z = \beta k w/m + (1-\beta)/n$$

The first equation says that the PageRank of t is the external contribution x , plus βz (the amount of PageRank not taxed) times the number of supporting pages, plus $(1-\beta)/n$, which is the share of "tax" that every page gets. The second equation says that each second-tier page gets $1/k$ -th of the untaxed PageRank of t , plus its share of the tax. The third equation says each supporting page gets 1 part in m/k of the untaxed PageRank of the second-tier page that reaches that supporting page, plus its share of the tax.

Begin by substituting for z in the first equation:

$$y = x + \beta^2 k w + \beta(1-\beta)m/n + (1-\beta)/n$$

Now, substitute for w in the above:

$$y = x + \beta^3 y + \beta(1-\beta)m/n + \beta^2(1-\beta)k/n + (1-\beta)/n$$

Neglect the last term $(1-\beta)/n$, per the directions in the statement of the problem. If we move the term $\beta^3 y$ to the left, and note that $\beta^3 = (1-\beta)(1+\beta+\beta^2)$, we get

$$y = x/(1-\beta^3) + (\beta/(1+\beta+\beta^2))(m/n) + (\beta/(1+\beta+\beta^2))(k/n)$$

For $\beta = 0.85$, these coefficients evaluate to:

$$y = 2.59x + 0.33(m/n) + 0.28(k/n)$$