Mining Massive Datasets: Review

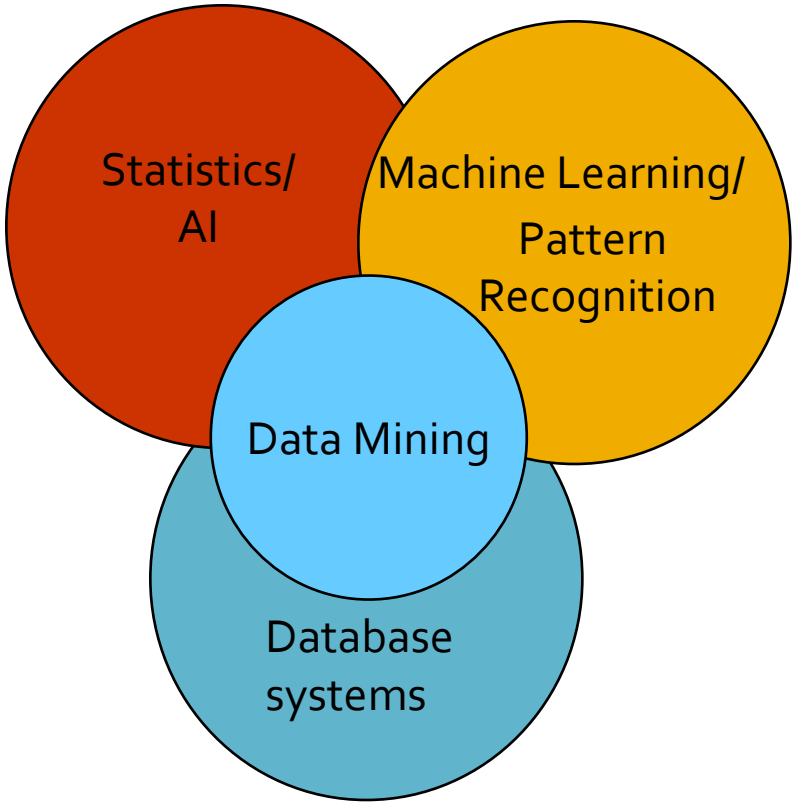
CS246: Mining Massive Datasets

Jure Leskovec, Stanford University

http://cs246.stanford.edu

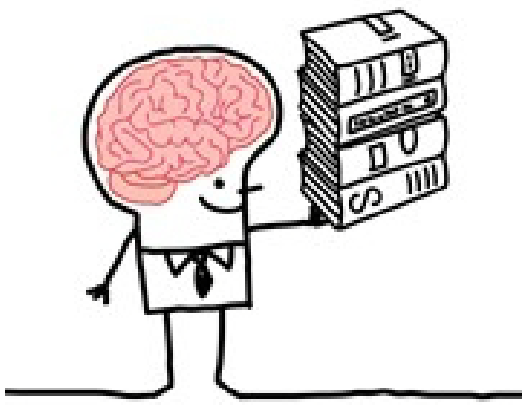
Mining Massive Datasets

- Overlaps with machine learning, statistics, artificial intelligence, databases, visualization but more stress on
  - scalability of number of features and instances
  - stress on algorithms and architectures
  - automation for handling large data

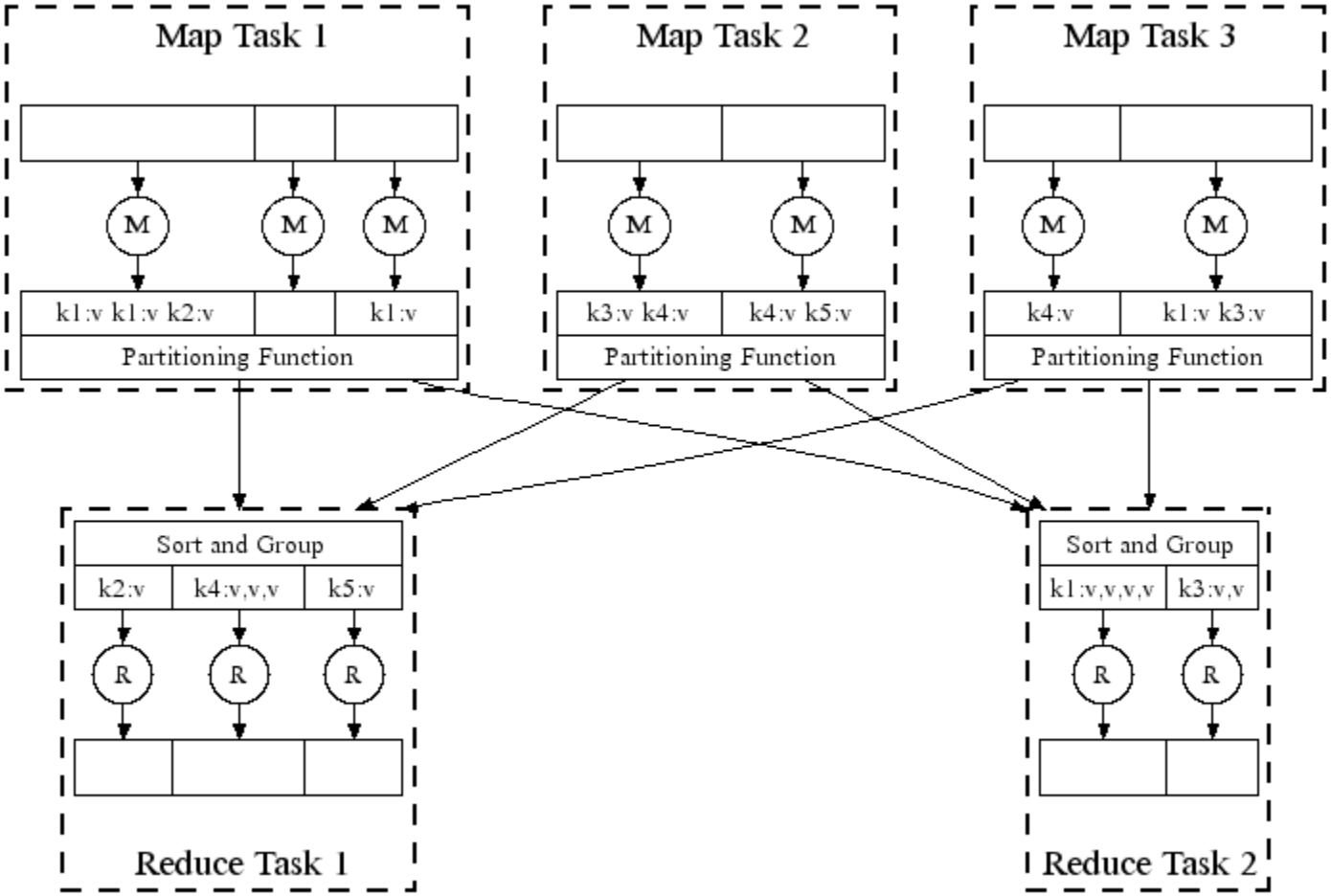


What we've covered

- MapReduce
- Association Rules
- Finding Similar Items
- Locality Sensitive Hashing
- Dim. Reduction (SVD, CUR))
- Clustering
- Recommender systems
- PageRank and TrustRank
- Machine Learning: kNN, SVM, Decision Trees
- Mining data streams
- Advertising on the Web



MapReduce


Map Task 1
M
M
M
k1:v k1:v k2:v
k1:v
Partitioning Function
Map Task 2
M
M
k3:v k4:v
k4:v k5:v
Partitioning Function
Map Task 3
M
M
k4:v
k1:v k3:v
Partitioning Function
Sort and Group
k2:v
k4:v,v,v
k5:v
R
R
R
Reduce Task 1
Sort and Group
k1:v,v,v,v
k3:v,v
R
R
Reduce Task 2

MapReduce: Word counting

Provided by the programmer

MAP:
reads input and produces a set of key value pairs

Group by key:
Collect all pairs with same key

Provided by the programmer

Reduce:
Collect all values belonging to the key and output

The crew of the space shuttle Endeavor recently returned to Earth as ambassadors, harbingers of a new era of space exploration. Scientists at NASA are saying that the recent assembly of the Dextre but is the first step in a long-term space-based man/machine partnership. '"The work we're doing now -- the robotics we're doing -- is what we're going to need to do to build any work station or habitat structure on the moon or Mars," said Allard Beutel.

Big document

(the, 1)
(crew, 1)
(of, 1)
(the, 1)
(space, 1)
(shuttle, 1)
(Endeavor, 1)
(recently, 1)
....

(key, value)

(crew, 1)
(crew, 1)
(space, 1)
(the, 1)
(the, 1)
(the, 1)
(shuttle, 1)
(recently, 1)
...

(key, value)

(crew, 2)
(space, 1)
(the, 3)
(shuttle, 1)
(recently, 1)
...

(key, value)

Only sequential reads

How it all fits together?

High-dimensional data:

- Locality Sensitive Hashing
- Dimensionality reduction
- Clustering

The data is a graph:

- Link Analysis: PageRank, TrustRank, Hubs & Authorities

Machine Learning:

- kNN, Perceptron, SVM, Decision Trees

Data is infinite:

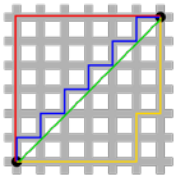
- Mining data streams
- Advertising on the Web

Applications:

- Association Rules
- Recommender systems

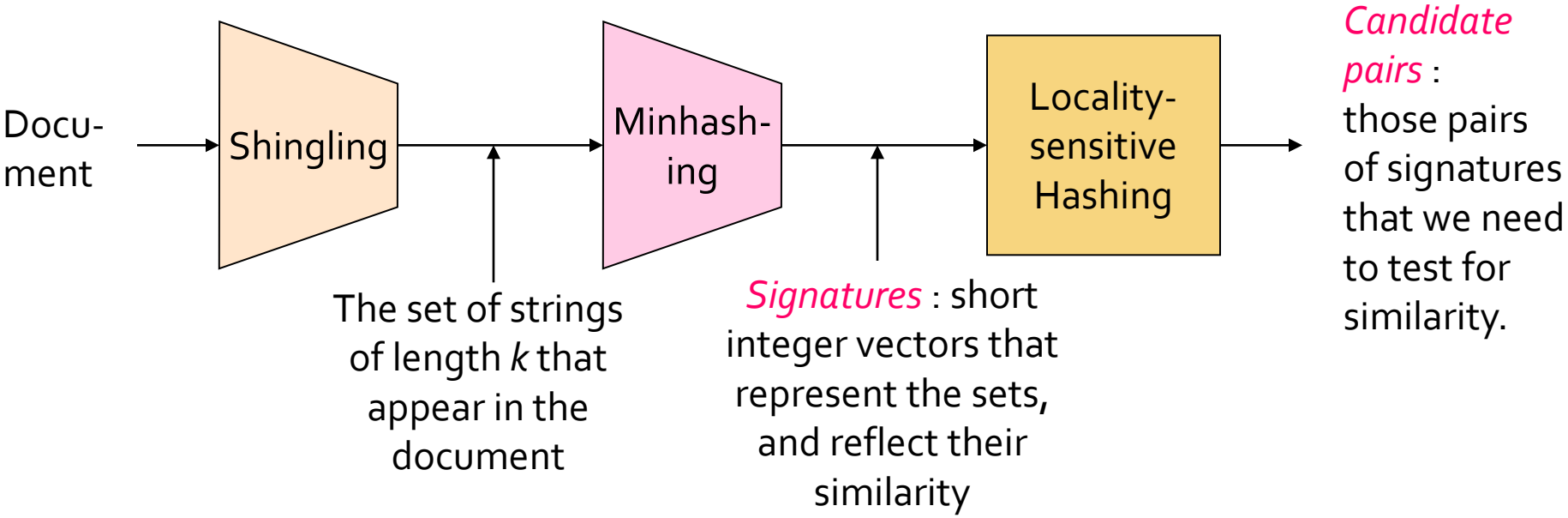
(1) Finding similar items

- Many problems can be expressed as finding "similar" sets:
  - Find near-neighbors in high-D space
- Distance metrics:
  - Points in $\Re^n$: L1, L2, Manhattan distance
  - Vectors: Cosine similarity
  - Sets of items: Jaccard similarity, Hamming distance



- Problem:
  - **Find near-duplicate documents**

3 Essential Steps:



1. *Shingling:* convert docs 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

Locality Sensitive Hashing

- **Shingling:** convert docs to sets of items
  - Shingle: sequence of *k* tokens that appear in doc
  - Example: k=2; D_1= abcab, 2-shingles: $S(D_1)$={ab, bc, ca}
  - Represent a doc by the set of hashes of its shingles

- **MinHashing:** convert large sets to short signatures, while preserving similarity
  - Similarity preserving hash func. *h*() s.t.:

    $$\mathbf{Pr}[h_\pi(S(D_1)) = h_\pi(S(D_2))] = Sim(S(D_1), S(D_2))$$

    - For Jaccard use permutation of columns and index of first 1.

Min Hashing – Example

Input matrix

1	4	3	1	0	1	0
3	2	4	1	0	0	1
7	1	7	0	1	0	1
6	3	6	0	1	0	1
2	6	1	0	1	0	1
5	7	2	1	0	1	0
4	5	5	1	0	1	0

Signature matrix *M*

2	1	2	1
2	1	4	1
1	2	1	2

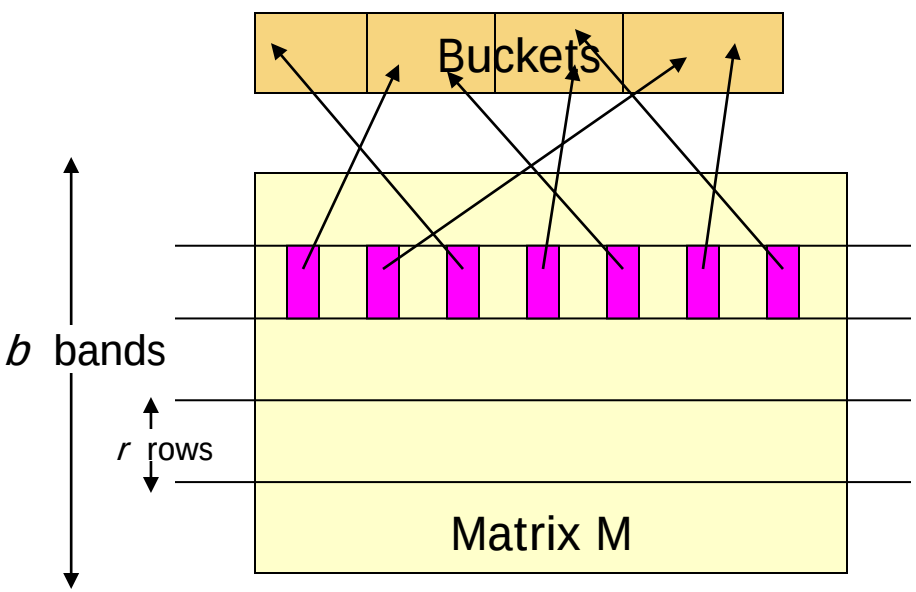
Similarities:

	1-3	2-4	1-2	3-4
Col/Col	0.75	0.75	0	0
Sig/Sig	0.67	1.00	0	0

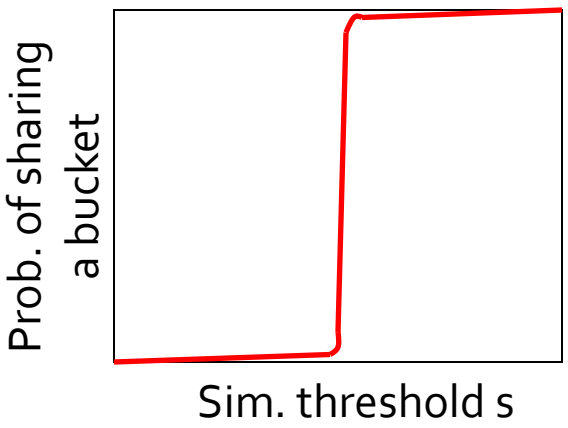
LSH

2	1	4	1
1	2	1	2
2	1	2	1

- Hash cols of signature matrix *M:* Similar columns likely hash to same bucket
  - Cols. *x* and *y* are a candidate pair if $M(i, x) = M(i, y)$ for at least frac. *s* values of *i*
  - Divide matrix *M* into *b* bands of *r* rows



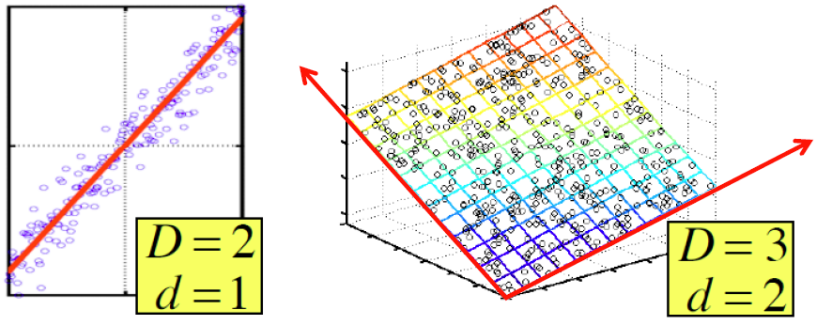
- $\text{Sim}(C_1, C_2) = s$
- Prob. that at least 1 band is identical = $1 - (1 - s^r)^b$
- Given *s*, tune *r* and *b* to get almost all pairs with similar signatures, but eliminate most pairs that do not have similar signatures



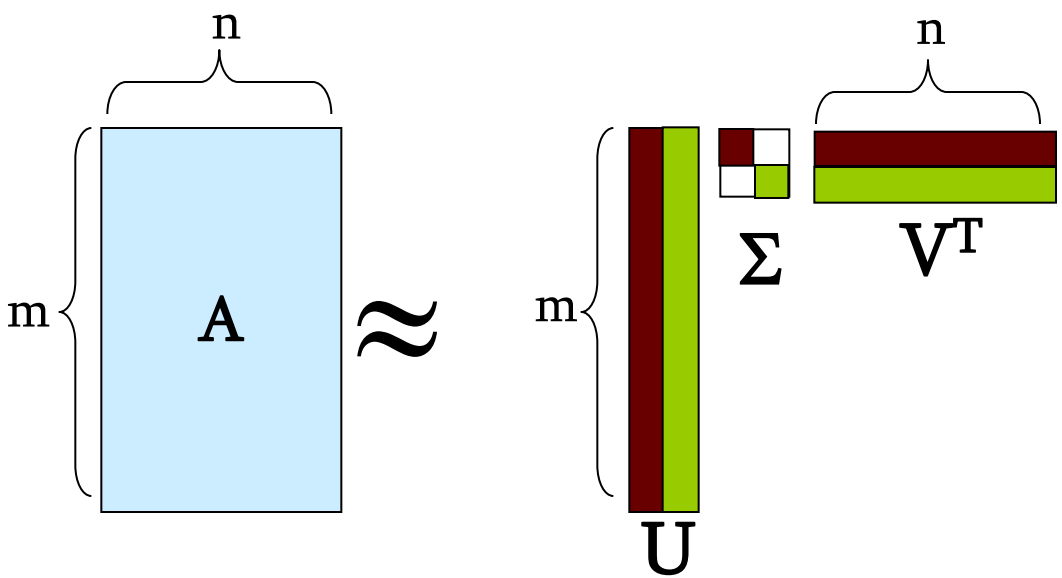
b=20, r=5

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

(2) Dimensionality Reduction



$$\mathbf{A} \approx \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i$$



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

user-to-concept similarity matrix

SciFi-concept

Romance-concept

$$
\begin{array}{c}
\\ \uparrow \\ \text{SciFi} \\ \downarrow \\ \uparrow \\ \text{Romnce} \\ \downarrow
\end{array}
\begin{array}{c}
\begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27
\end{bmatrix}
\times
\begin{bmatrix}
9.64 & 0 \\
0 & 5.29
\end{bmatrix}
\times
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71
\end{bmatrix}
$$

- **A = U Σ V^T** - example:

$$
\begin{array}{c}
\\ \uparrow \\ \text{SciFi} \\ \downarrow \\ \uparrow \\ \text{Romnce} \\ \downarrow
\end{array}
\begin{array}{c}
\begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27
\end{bmatrix}
\times
\begin{array}{c}
\text{'strength' of SciFi-concept} \\
\begin{bmatrix}
9.64 & 0 \\
0 & 5.29
\end{bmatrix}
\end{array}
\times
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71
\end{bmatrix}
$$

- **A = U Σ V^T** - example:

movie-to-concept similarity matrix

SciFi-concept

$$
\begin{array}{c}
\\ \uparrow \\ \text{SciFi} \\ \downarrow \\ \uparrow \\ \text{Romnce} \\ \downarrow
\end{array}
\begin{array}{c}
\begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27
\end{bmatrix}
\times
\begin{bmatrix}
9.64 & 0 \\
0 & 5.29
\end{bmatrix}
\times
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71
\end{bmatrix}
$$

SVD

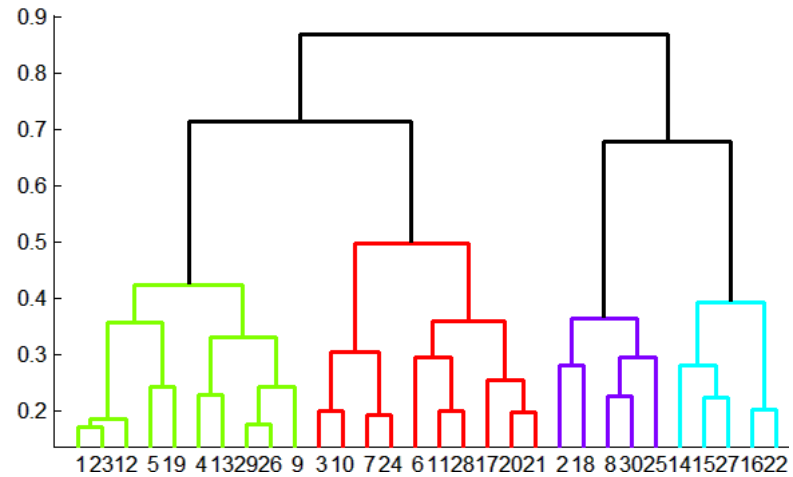
- How to do dimensionality reduction:
  - Set small singular values to zero
- How to query?
  - Map query vector into "concept space" –
    - How? Compute q·V

Even though d and q do not share a movie, they are still similar

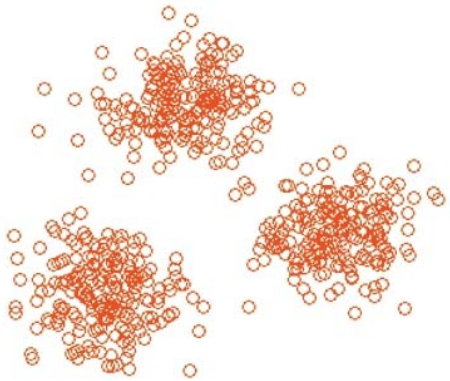
	Matrix	Alien	Serenity	Casablanca	Amelie		SciFi-concept	
d=	0	4	5	0	0	→	1.16	0
q=	5	0	0	0	0	→	0.58	0

(3) Clustering

- Hierarchical:
  - Agglomerative (bottom up):
    - Initially, each point is a cluster
    - Repeatedly combine the two "nearest" clusters into one
    - Represent a cluster by its centroid or clustroid

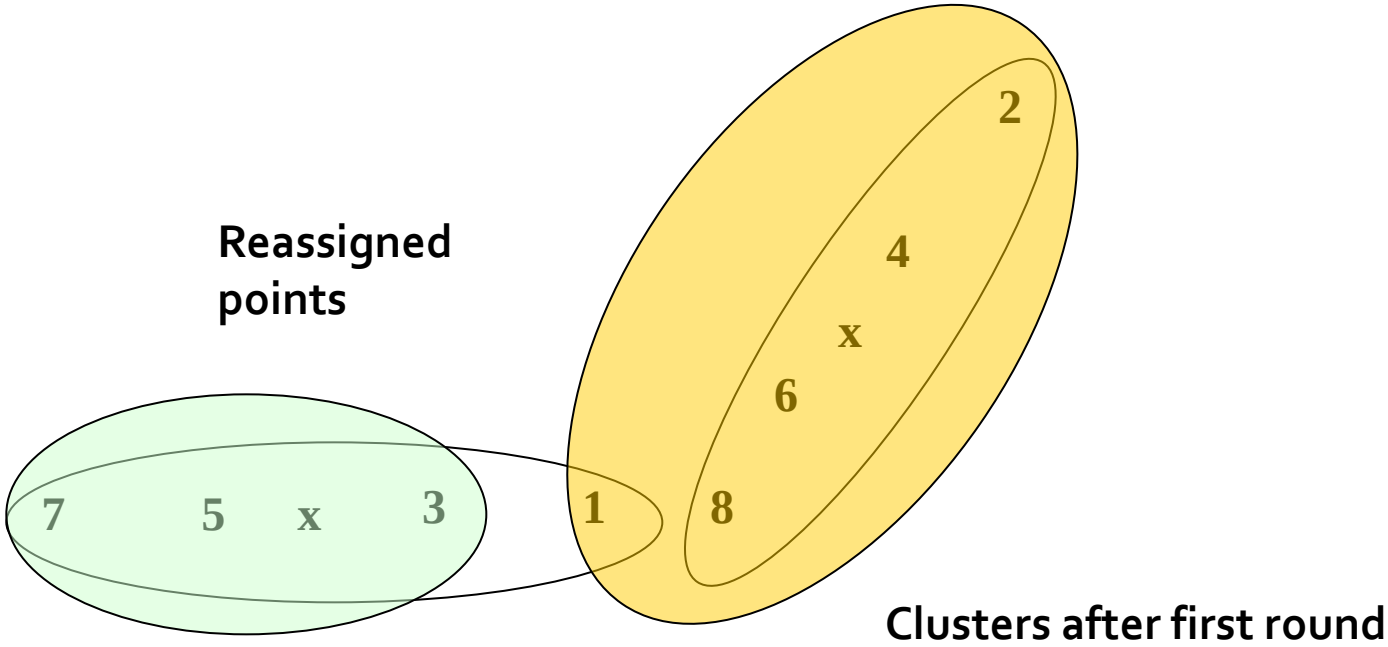


- Point Assignment:
  - Maintain a set of clusters
  - Points belong to "nearest" cluster



k-Means

- **k-means**: initialize cluster centroids
  - Iterate:
    - For each point, place it in the cluster whose current centroid it is nearest
    - Update the cluster centroids based on memberships



High-dim data methods: Comparison

- LSH:
  - Find somewhat similar pairs of items while avoiding $O(N^2)$ comparisons
- Clustering:
  - Assign points into a prespecified number of clusters
    - Each point belongs to a single cluster
    - Summarize the cluster by a centroid (e.g., topic vector)
- SVD (dimensionality reduction):
  - Want to explore correlations in the data
  - Some dimensions may be irrelevant
  - Useful for visualization, removing noise from the data, detecting anomalies

How it all fits together?

High-dimensional data:

- Locality Sensitive Hashing
- Dimensionality reduction
- Clustering

The data is a graph:

- Link Analysis: PageRank, TrustRank, Hubs & Authorities

Machine Learning:

- kNN, Perceptron, SVM, Decision Trees

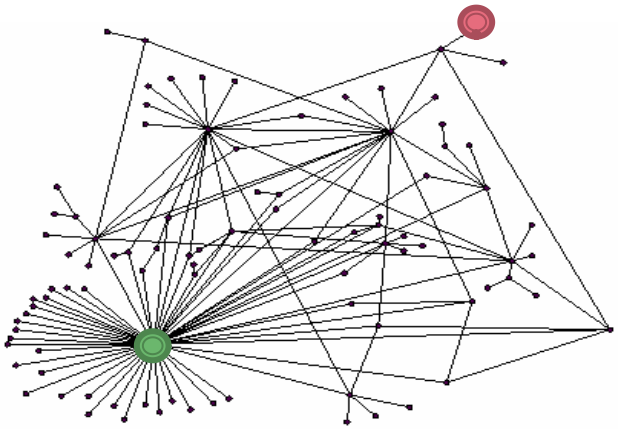
Data is infinite:

- Mining data streams
- Advertising on the Web

Applications:

- Association Rules
- Recommender systems

Link Analysis: PageRank



- Rank nodes using link structure
- **PageRank:**
  - Link voting:
    - P with importance x has n out-links, each link gets x/n votes
    - Page R's importance is the sum of the votes on its in-links
  - Complications: Spider traps, Dead-ends
  - At each step, random surfer has two options:
    - With probability β, follow a link at random
    - With prob. 1-β, jump to some page uniformly at random

TrustRank & SimRank

- **TrustRank**: topic-specific PageRank with a teleport set of "trusted" pages
  - Spam mass of page p:
    - Fraction of pagerank score r(p) coming from spam pages: $|r(p) - r^{+}(p)| / r(p)$
- **SimRank**: measure similarity between items
  - a k-partite graph with k types of nodes
    - Example: picture nodes and tag nodes
  - Perform a random-walk with restarts from node N
    - i.e., teleport set = {N}.
  - Resulting prob. distribution measures similarity to N

Hubs & Authorities

- HITS (Hypertext-Induced Topic Selection) is a measure of importance of pages or documents, similar to PageRank:
  - Authorities are pages containing useful information
    - E.g., course home pages
  - Hubs are pages that link to authorities
    - On-line list of links to CS courses.
- Mutually recursive definition:
  - A good hub links to many good authorities
  - A good authority is linked from many good hubs
  - Model using **two** scores for each node:
    - **Hub** score *h* and **Authority** score *a*

PageRank vs. HITS

- PageRank and HITS are two solutions to the same problem:
  - What is the value of an in-link from *u* to *v*?
  - In the PageRank model, the value of the link depends on the links **into** *u*
  - In the HITS model, it depends on the value of the other links **out of** *u*
  - PageRank gives flexibility with teleportation

How it all fits together?

High-dimensional data:

- Locality Sensitive Hashing
- Dimensionality reduction
- Clustering

The data is a graph:

- Link Analysis: PageRank, TrustRank, Hubs & Authorities

Machine Learning:

- kNN, Perceptron, SVM, Decision Trees

Data is infinite:

- Mining data streams
- Advertising on the Web

Applications:

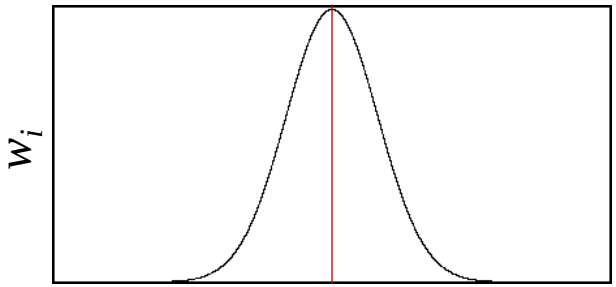
- Association Rules
- Recommender systems

Large Scale Machine Learning

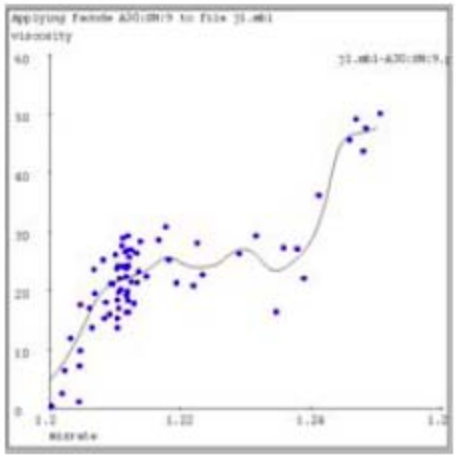
- Would like to do prediction: **estimate** a function f(x) so that y = f(x)
- Where *y* can be:
  - Real number: Regression
  - Categorical: Classification
- Data is labeled: have many pairs {(x, y)}
  - x … vector of real valued features
  - y … class ({+1, -1}, or a real number)
- Methods:
  - k-Nearest Neighbor
  - Support Vector Machines
  - Decision trees

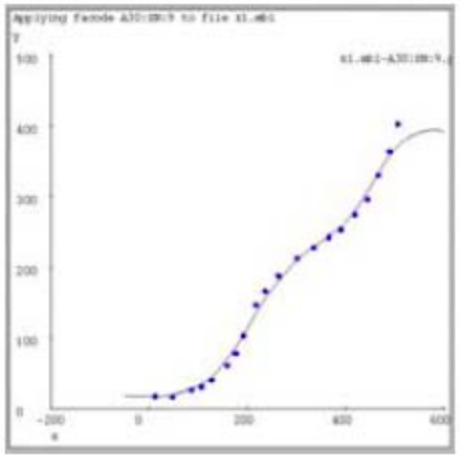
k-Nearest Neighbors

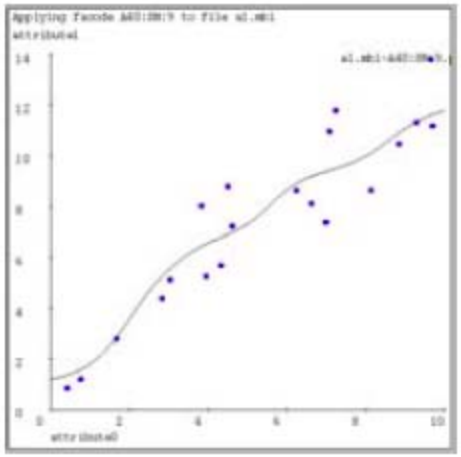
- Distance metric:
  - Euclidean
- How many neighbors to look at?
  - All of them (!)
- Weighting function:
  - $w_i = \exp(-d(x_i, q)^2/K_w)$
    - Nearby points to query q are weighted more strongly. K_w…kernel width.
- How to fit with the local points?
  - Predict weighted average: $\Sigma w_i y_i / \Sigma w_i$



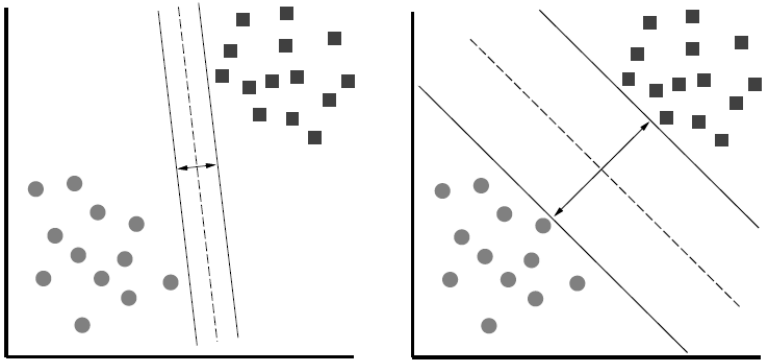
$d(x_i, q) = 0$







Support Vector Machines

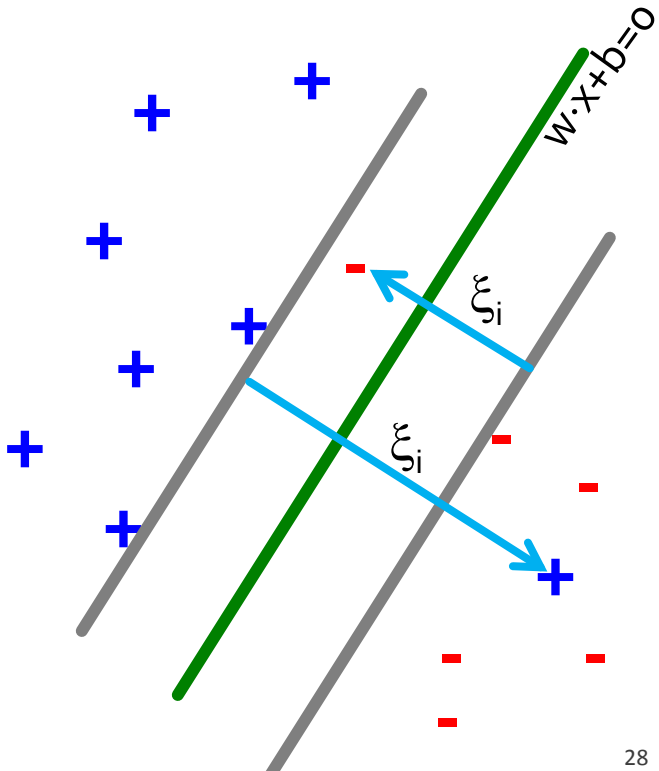


- Prediction = sign($w \cdot x + b$)
  - Model parameters w, b
- Margin: $\gamma = \frac{\|w\|}{w \cdot w} = \frac{1}{\|w\|}$
- SVM optimization problem:

$$\min_{w,b,\xi_i \geq 0} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$

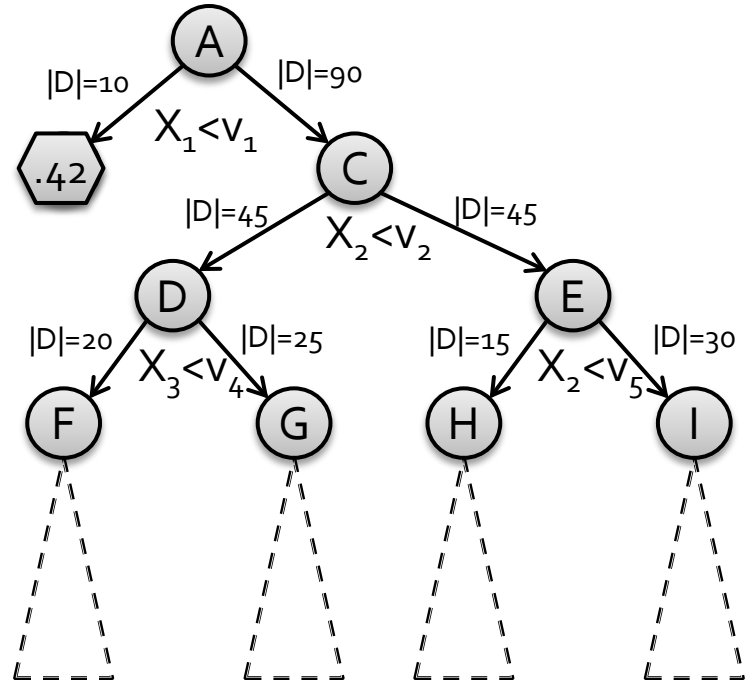
$$s.t. \forall i, y_i(w \cdot x_i + b) \geq 1 - \xi_i$$

- Find w,b using Stochastic gradient descent



Decision Trees: PLANET

- Building decision trees using MapReduce
  - How to predict?
    - Predictor: avg. y_i of the examples in the leaf
  - When to stop?
    - # of examples in the leaf is small
  - How to build?
    - One MapReduce job per level
      - Need to compute split quality for each attribute and each split value for each current leaf



Algorithm 1 **FindBestSplit**

Require: Node n, Data $D \subseteq D^*$

$(n \rightarrow \text{split}, D_L, D_R) =$ **FindBestSplit**(D)
if StoppingCriteria(D_L) **then**
  $n \rightarrow$left_prediction=FindPrediction(D_L)
else
  FindBestSplit$(n \rightarrow \text{left}, D_L)$
if StoppingCriteria(D_R) **then**
  $n \rightarrow$right_prediction=FindPrediction(D_R)
else
  FindBestSplit$(n \rightarrow \text{right}, D_R)$

When to use which method

- **SVM:** classification
  - Millions of numerical features (e.g., documents)
  - Simple (linear) decision boundary
  - Hard to interpret model
- kNN: classification or regression
  - (Many) numerical features
  - Many parameters to play with –distance metric, k, weighting, … there is no simple way to set them!
- **Decision Trees:** classification or regression
  - Relatively few features (handles categorical features)
  - Complicated decision boundary
    - Overfitting can be a problem
  - Easy to explain/interpret the classification

How it all fits together?

High-dimensional data:

- Locality Sensitive Hashing
- Dimensionality reduction
- Clustering

The data is a graph:

- Link Analysis: PageRank, TrustRank, Hubs & Authorities

Machine Learning:

- kNN, Perceptron, SVM, Decision Trees

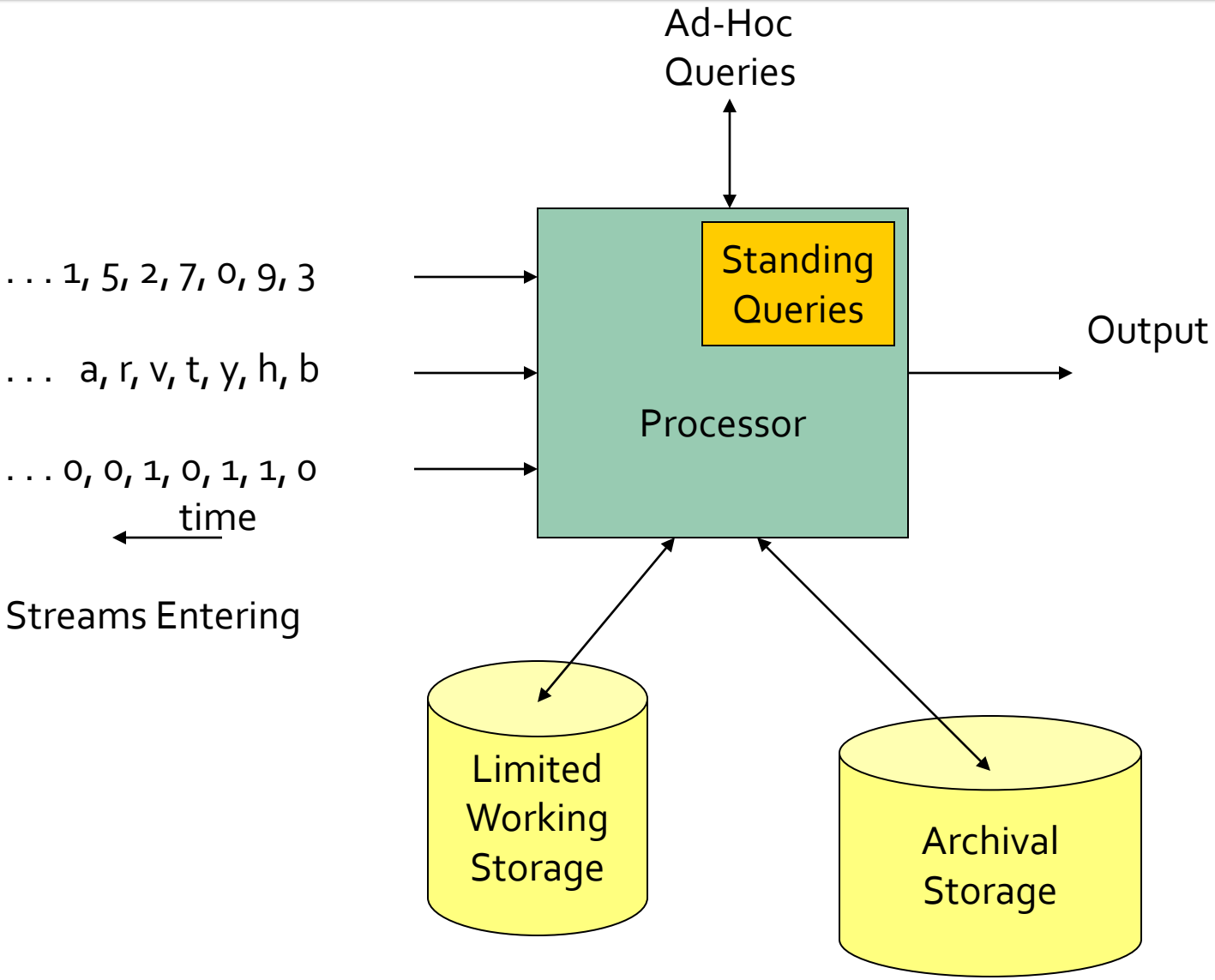
Data is infinite:

- Mining data streams
- Advertising on the Web

Applications:

- Association Rules
- Recommender systems

Mining Data Streams



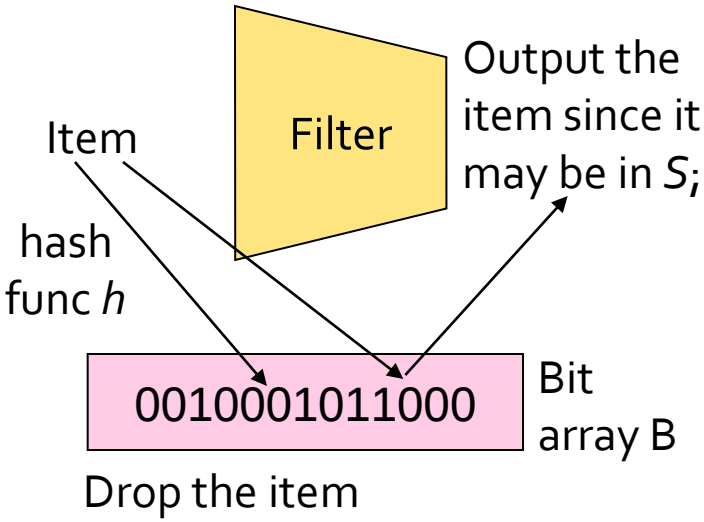
Problems on data streams

- Can't store the whole stream but we are happy with an approximate answer
  - Sampling data from a stream:
    - Sample of size k: each element is included with prob. k/N
  - Queries over sliding windows:
    How many 1s are in last k bits?
    - DGIM: summarize blocks with specific number of 1s
    - To estimate the number of 1s in the most recent *N* bits:
      - Sum the sizes of all buckets but the last
      - Add half the size of the last bucket

1001010110001011010101010101011010101010101011101010101110101000101100010

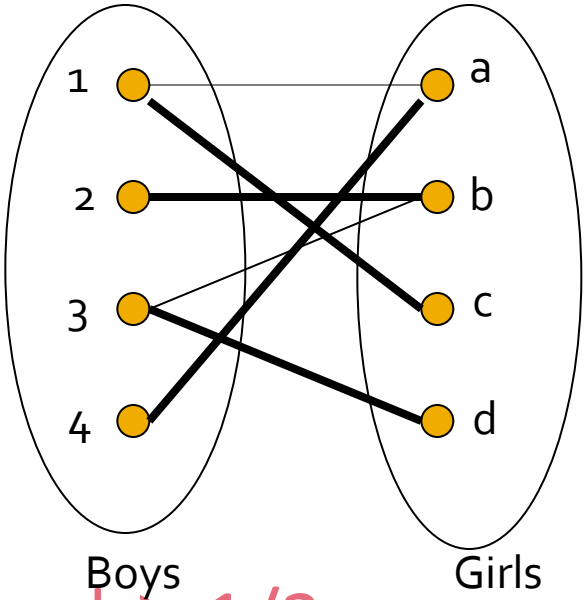
Problems on data streams

- Filtering a stream:
  - Select elements with property x from stream
  - **Bloom filters**
- Counting distinct elements:
  - Number of distinct elements in the last *k* elements of the stream
  - **Flajolet-Martin**:
    - For each item *a*, let *r*(*a*) be the # of trailing 0s in *h*(*a*)
    - Record *R* = the maximum *r*(*a*) seen
      - R = $\max_a$ r(a), over all the items *a* seen so far
    - Estimated number of distinct elements = 2^R



Online algorithms & Advertising

- You get to see one input piece at a time, and need to make irrevocable decisions
- Competitive ratio = $\min_{\text{all inputs I}} (|M_{my\_alg}|/|M_{opt}|)$
- Greedy online matching: competitive ratio= $|M_{greedy}|/|M_{opt}| \geq 1/2$
- Addwords problem:
  - Query arrives to a search engine
  - Several advertisers bid on the query query
  - Pick a subset of advertisers whose ads are shown



BALANCE algorithm

- BALANCE Algorithm:
  - For each query, pick the advertiser with the largest unspent budget
    - Break ties arbitrarily (in a deterministic way)
  - Two advertisers A and B
    - A bids on query x, B bids on x and y
    - Both have budgets of $4
    - Query stream: xxxxyyyy
    - BALANCE choice: ABABBB__
      - Optimal: AAAABBBB, Competitive ratio = ¾
  - Generally, competitive ratio = 1-1/e

How it all fits together?

High-dimensional data:

- Locality Sensitive Hashing
- Dimensionality reduction
- Clustering

The data is a graph:

- Link Analysis: PageRank, TrustRank, Hubs & Authorities

Machine Learning:

- kNN, Perceptron, SVM, Decision Trees

Data is infinite:

- Mining data streams
- Advertising on the Web

Applications:

- Association Rules
- Recommender systems

Association Rule Discovery

Supermarket shelf management – Market-basket model:

- Goal: To identify items that are bought together by sufficiently many customers
- Approach: Process the sales data collected with barcode scanners to find dependencies among items

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Rules Discovered:

{Milk} --> {Coke}

{Diaper, Milk} --> {Beer}

Apriori algorithm

- Observation: Subsets of a frequent itemset are freq
- Consequence: Build frequent itemsets bottom up
- Example: Items = {milk, coke, pepsi, beer, juice}
  - Min Support = 3 baskets

    B_1 = {m, c, b} B_2 = {m, c, j}

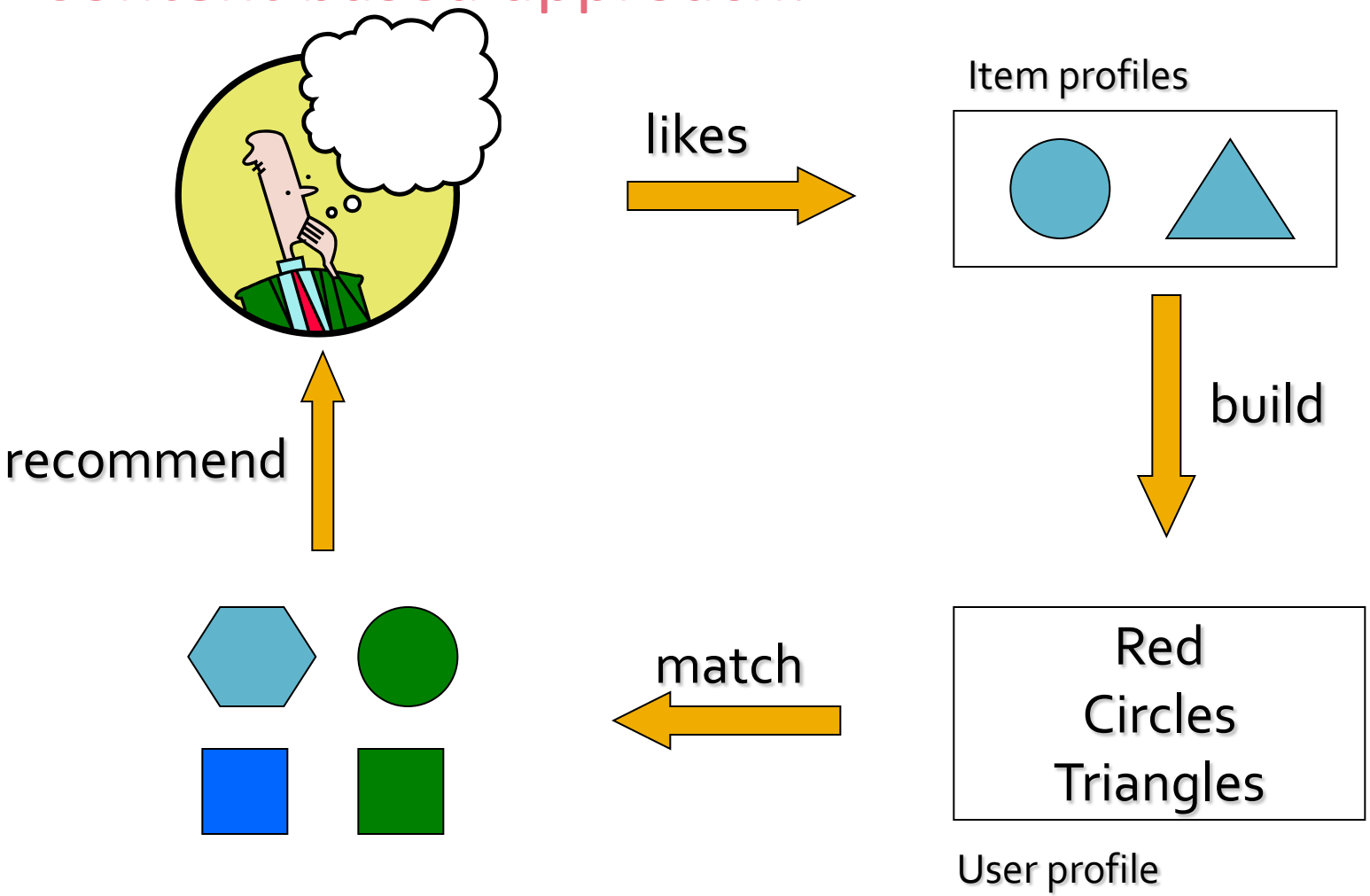
    B_3 = {m, b} B_4= {c, j}

    B_5 = {m, p, b} B_6 = {m, c, b, j}

    B_7 = {c, b, j} B_8 = {b, c}
- Frequent 1-sets: {m}, {c}, {b}, {j}
- Frequent 2-sets: {m,c}, {m,b}, ~~{m,j}~~, {c,b}, {c,j}, ~~{b,j}~~
  - Need not even consider sets {p, *} as {p} is not frequent
- Frequent 3-sets: only need to check ~~{m, c, b}~~

(5) Recommender Systems

- Content based approach:



Collaborative filtering

- User-user collaborative filtering
  - Consider user *c*
  - Find set *D* of other users whose ratings are "similar" to *c*'s ratings
  - Estimate user's ratings based on the ratings of users in *D*
- Item-item collaborative filtering
  - Estimate rating for item based on ratings for similar items

Closing…

CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
http://cs246.stanford.edu

What we've learned this quarter

- MapReduce
- Association Rules
- Apriori algorithm
- Finding Similar Items
- Locality Sensitive Hashing
- Random Hyperplanes
- Dimensionality Reduction
- Singular Value Decomposition
- CUR method
- Clustering
- Recommender systems
- Collaborative filtering
- PageRank and TrustRank
- Hubs & Authorities
- k-Nearest Neighbors
- Perceptron
- Support Vector Machines
- Stochastic Gradient Descent
- Decision Trees
- Mining data streams
- Bloom Filters
- Flajolet-Martin
- Advertising on the Web

BIG PICTURE

- How to analyze large datasets to discover patterns and models that are:
  - valid: hold on new data with some certainty
  - novel: non-obvious to the system
  - useful: should be possible to act on the item
  - understandable: humans should be able to interpret the pattern
- How to do this using massive data (that does not fit into main memory)

What next?

- Seminars:
  - InfoSeminar: http://i.stanford.edu/infoseminar
  - RAIN Seminar: http://rain.stanford.edu
- Conferences:
  - KDD: ACM Conference on Knowledge Discovery and Data Mining
  - ICDM: IEEE International Conference on Data Mining
  - WWW: World Wide Web Conference
  - ICML: International Conference on Machine Learning
  - NIPS: Neural Information Processing Systems
- Some courses:
  - CS341: Research Project in Data Mining
  - CS224W: Social and Information Network Analysis
  - CS276: Information Retrieval and Web Search
  - CS229: Machine Learning
  - CS448g: Interactive Data Analysis

You have done a lot!!!

- And (hopefully) learned a lot!!!
  - Implemented a number of methods
  - Answered questions and proved many interesting results
  - And did excellently on the final!

Thank You for the Hard Work!!!