Mining Massive Datasets: Review

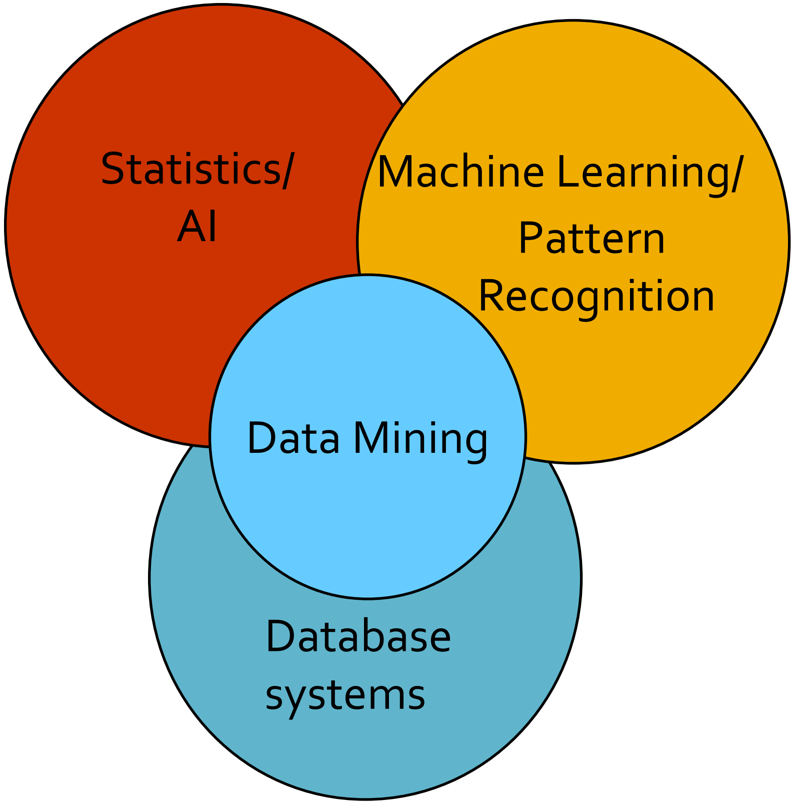
CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
http://cs246.stanford.edu

Data Mining

- **Models** **and** **tools for discovering patterns** **and** **answering queries** **that are:**
  - **Valid:** Hold on new data with some certainty
  - **Useful:** Should be possible to act on the item
  - **Unexpected:** Non-obvious to the system
  - **Understandable:** Humans should be able to interpret the pattern

Mining Massive Datasets

- **Overlaps with machine learning, statistics, artificial intelligence, databases, but more stress on**
  - **Scalability** of number of features and instances
  - **Algorithms** and **architectures**
  - Automation for handling **large data**



What We Have Covered

- Apriori
- MapReduce
- Association rules
- Frequent itemsets
- PCY
- Recommender systems
- PageRank
- TrustRank
- HITS
- SVM
- Decision Trees
- Perceptron
- Web Advertising
- DGIM
- Bandits
- BFR
- Regret
- LSH
- MinHash
- SVD
- Clustering
- Matrix factorization
- CUR
- Bloom filters
- Flajolet-Martin
- CURE
- Submodularity
- SGD
- Collaborative Filtering
- SimRank
- Random hyperplanes
- Trawling
- AND-OR constructions
- k-means

How It All Fits Together

- **Based on different types of data:**
  - Data is **high dimensional**
  - Data is a **graph**
  - Data is **never-ending**
  - Data is **labeled**
- **Based on different models of computation:**
  - **Single machine in-memory**
  - **MapReduce**
  - **Streams**
  - **Batch (offline) vs. Active (online) algorithms**

How It All Fits Together

- Based on different applications:
  - **Recommender systems**
  - **Market basket analysis**
  - **Link analysis, spam detection**
  - **Duplicate detection** and **similarity search**
  - **Web advertising**
- Based on different "tools":
  - **Linear algebra:** SVD, Matrix factorization
  - **Optimization:** Stochastic gradient descent
  - **Dynamic programming:** Frequent itemsets
  - **Hashing:** LSH, Bloom filters

How It All Fits Together

High dim. data	Graph data	Infinite data	Machine learning	Apps
Locality sensitive hashing	PageRank, SimRank	Filtering data streams	SVM	Recommen der systems
Clustering	Community Detection	Web advertising	Decision Trees	Association Rules
Dimensional ity reduction	Spam Detection	Queries on streams	Perceptron, kNN, Bandits	Duplicate document detection

How it all fits together?

Data is High-dimensional:

- Locality Sensitive Hashing
- Dimensionality reduction
- Clustering

Data is a graph:

- Link Analysis: PageRank, TrustRank, Hubs & Authorities

Data is Labeled (Machine Learning):

- kNN, Perceptron, SVM, Decision Trees

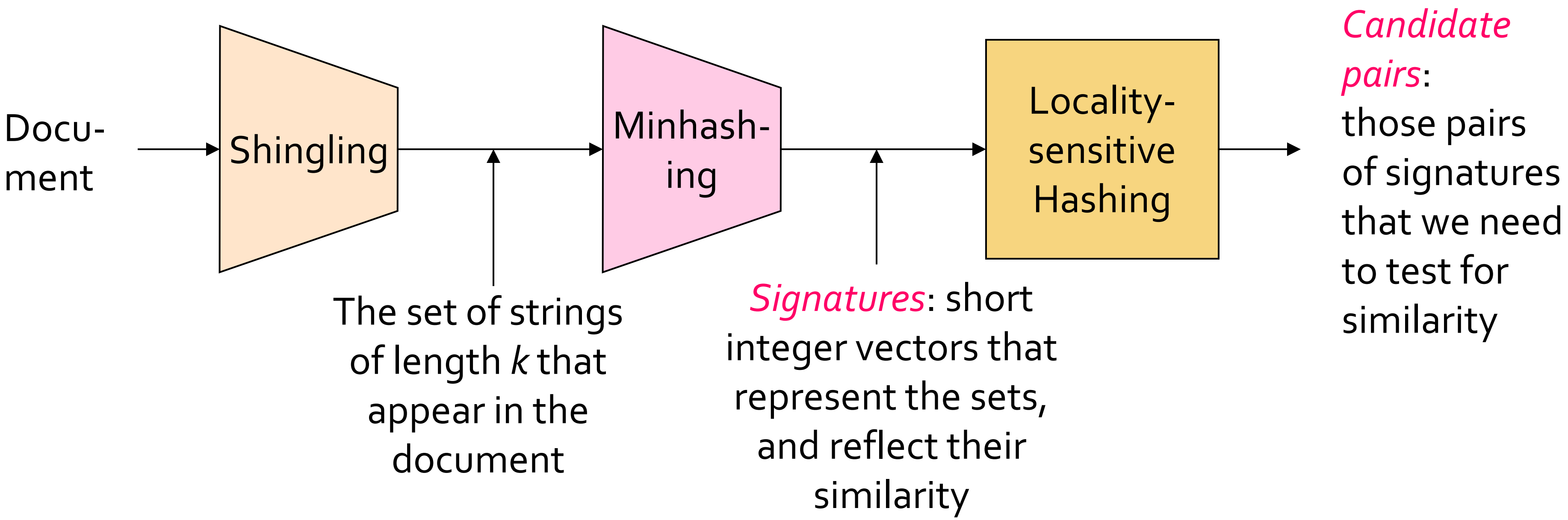
Data is infinite:

- Mining data streams
- Advertising on the Web

Applications:

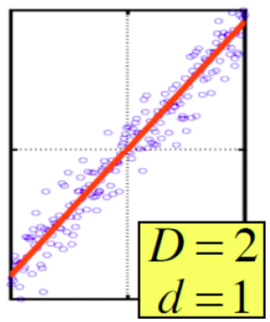
- Association Rules
- Recommender systems

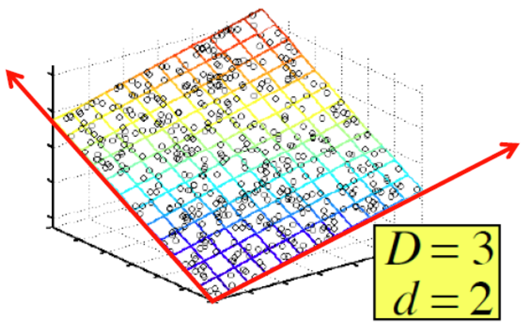
(1) Finding "similar" sets



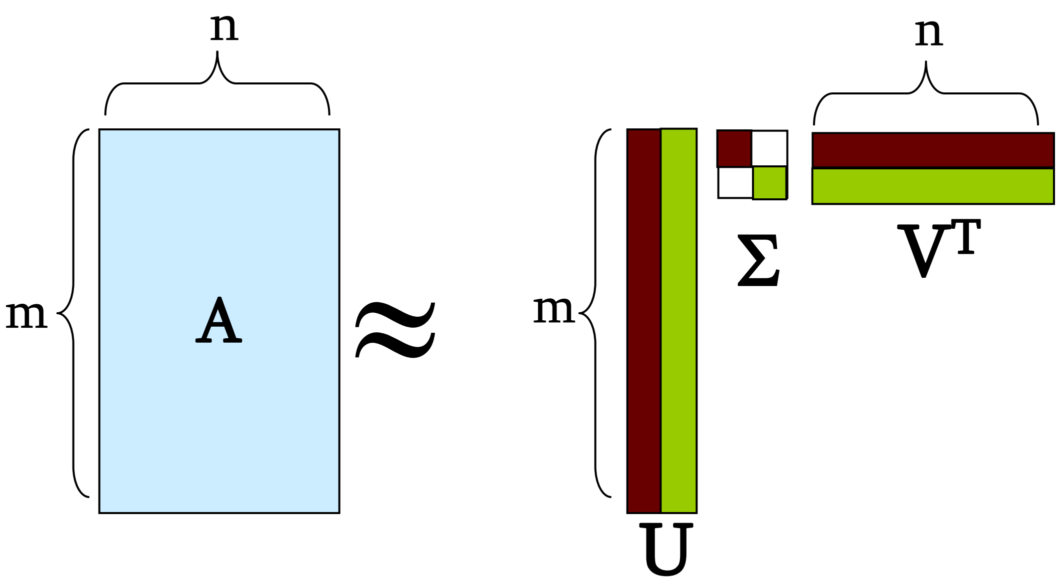
1. ***Shingling:*** Convert docs to sets
2. ***Minhashing:*** Convert large sets to short signatures, while preserving similarity
3. ***Locality-sensitive hashing:*** Focus on pairs of signatures likely to be of similar documents

(2) Dimensionality Reduction



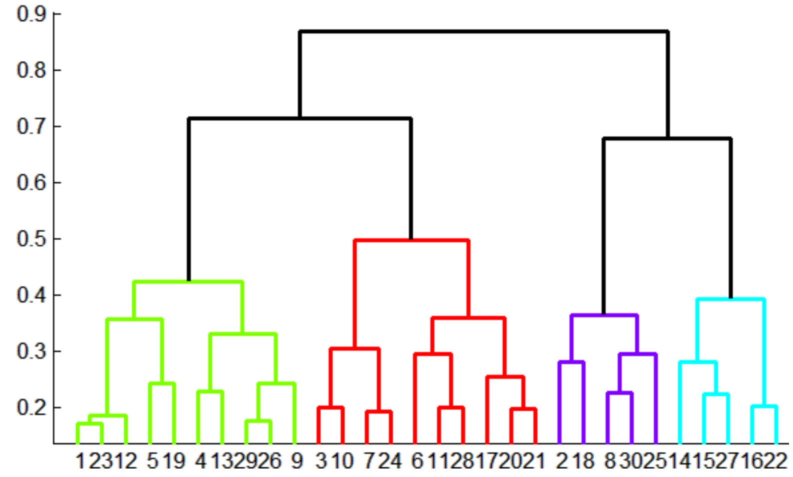


$$\mathbf{A} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i$$

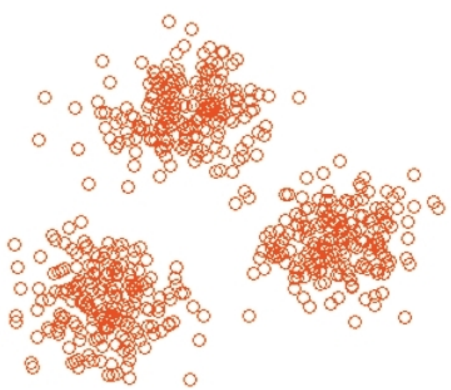


(3) Clustering

- **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: k-means, BFR**
  - Maintain a set of clusters
  - Points belong to “nearest” cluster



High-dim data methods: Comparison

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

When to use which method?

- **Find all similar pairs of items: LSH**
  - Have to know the threshold ahead of time
  - Allow for some error
- **Identify clusters (structure in data): k-means**
  - ***k*** is usually relatively small (10~1000)
  - Useful for identifying 'types' or 'classes' of datapoints
- **Build low-dimensional representation of data: SVD**
  - More robust (noise-fee) similarity computation
  - Data compression (memory saving, speed-up)

How it all fits together?

Data is high-dimensional:

- Locality Sensitive Hashing
- Dimensionality reduction
- Clustering

The data is a graph:

- Link Analysis: PageRank, TrustRank, Hubs & Authorities

Data is labeled (Machine Learning):

- kNN, Perceptron, SVM, Decision Trees

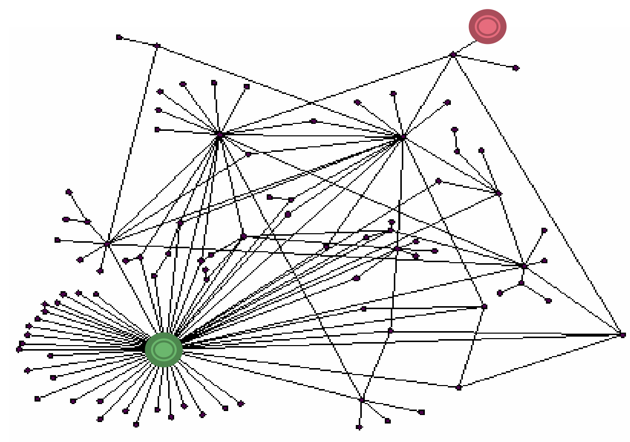
Data is infinite:

- Mining data streams
- Advertising on the Web

Applications:

- Association Rules
- Recommender systems

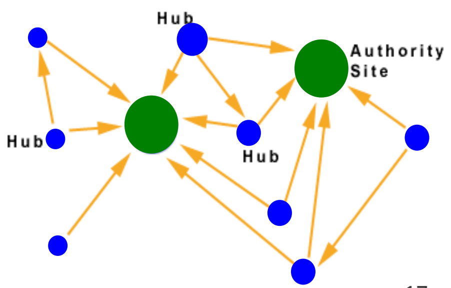
Link Analysis: PageRank



- **Rank nodes using the network link structure**
- **PageRank:**
  - **Link voting:**
    - **Page** of importance ***x*** has ***n*** out-links, each gets ***x/n*** votes
    - Page ***R***'s importance is the sum of the votes on its in-links
  - **Complications:** Spider traps, Dead-ends
  - **Solution:** At each step, random surfer has 2 options
    - With probability β, follow a link at random
    - With prob. **1-**β, jump to some page **uniformly** at random
  - **Power method to compute PageRank**

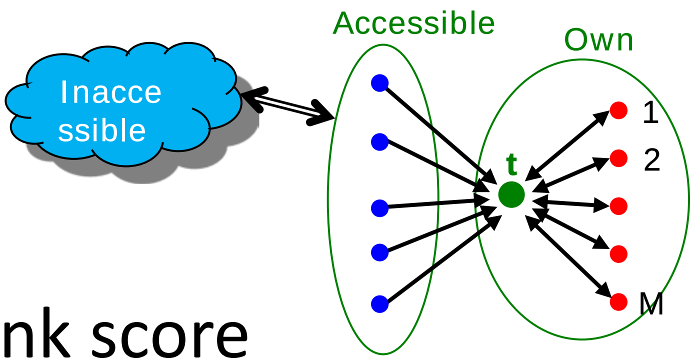
PPR, SimRank, HITS

- **Personalized (topic specific) PageRank**
  - Random walker teleports to a preselected set of nodes
- **Random Walk with Restarts**
  - Random walker always jumps back to the starting node
- **SimRank**
  - **Measure similarity between items**
  - ***k***-partite graph with ***k*** types of nodes
  - Perform a random-walk with restarts from node ***N***
  - Resulting prob. distrib. is similarity of other nodes to ***N***
- **Hubs & Authorities**
  - **Experts vs. Content provides**
  - Principle of repeated improvement

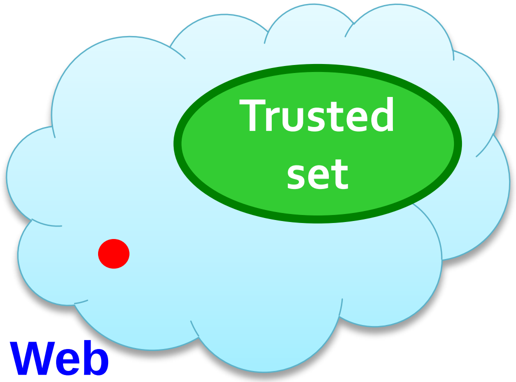


WebSpam and PageRank

- **Web spam farming**
  - Architecture of a spam farm
  - Effect of spam farms on PageRank score

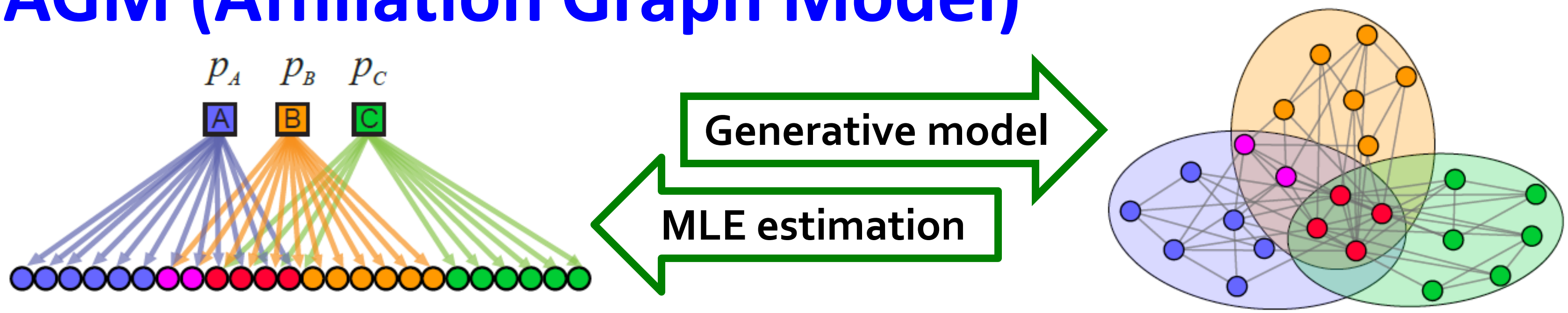


- **TrustRank**
  - Topic specific PageRank with a teleport set of "trusted" pages
  - **Spam Mass** of a page

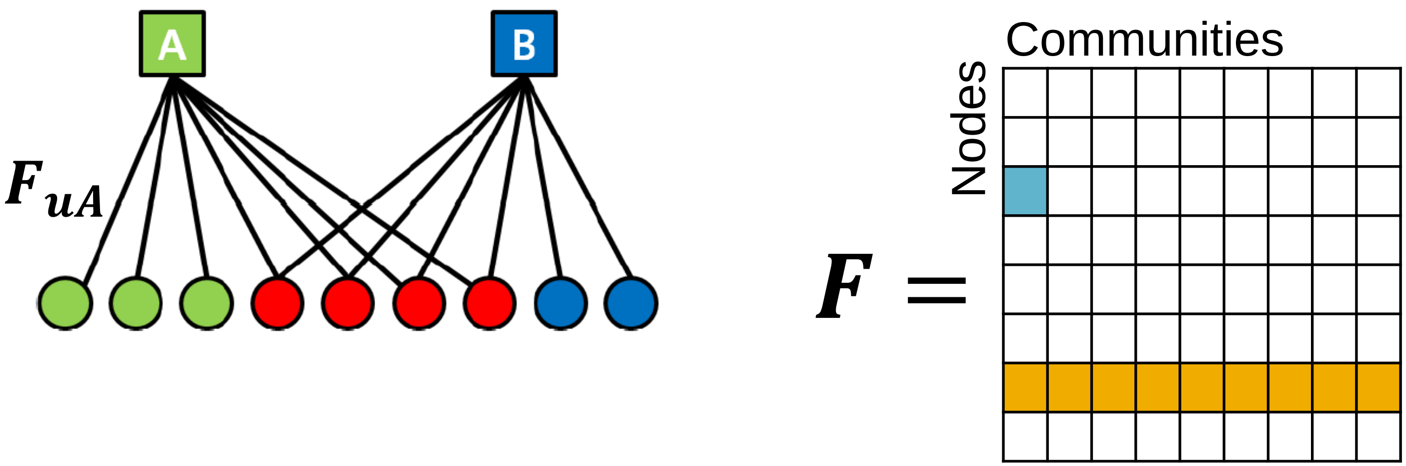


Analysis of Large Graphs

- **AGM (Affiliation Graph Model)**



- **BigCLAM (CLuster Affiliation Model)**



How it all fits together?

Data is high-dimensional:

Locality Sensitive Hashing
Dimensionality reduction
Clustering

The data is a graph:

Link Analysis: PageRank, TrustRank, Hubs & Authorities

Data is labeled (Machine Learning):

kNN, Perceptron, SVM, Decision Trees

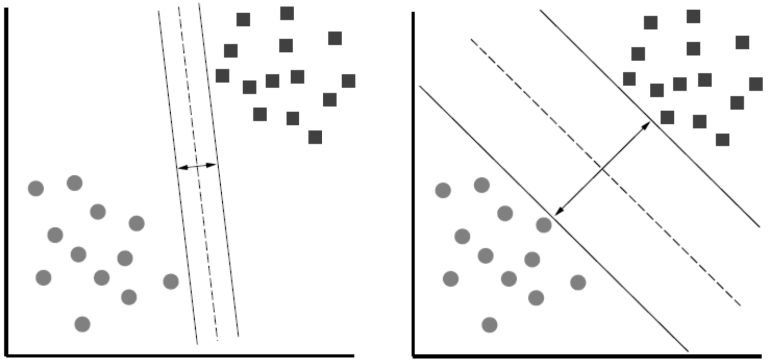
Data is infinite:

Mining data streams
Advertising on the Web

Applications:

Association Rules
Recommender systems

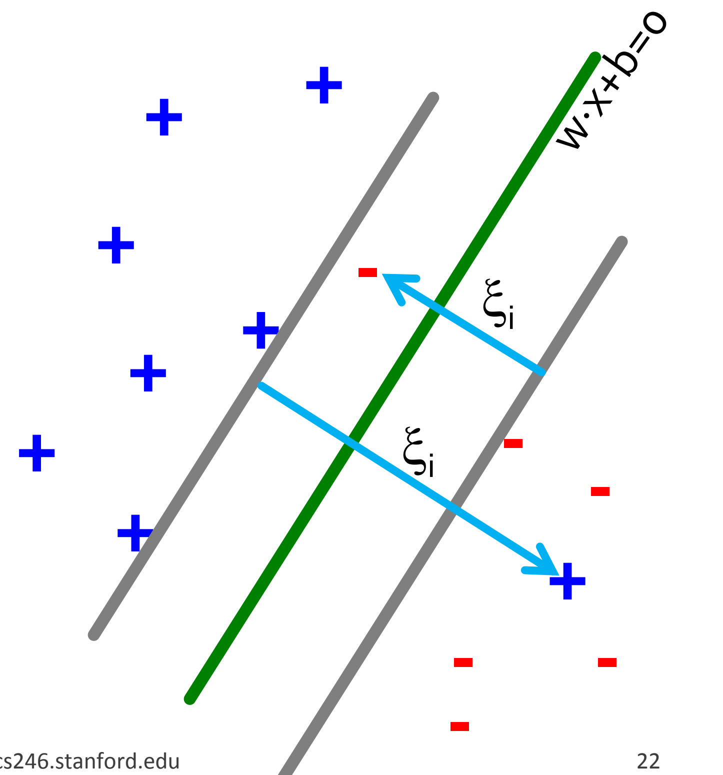
Support Vector Machines



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

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

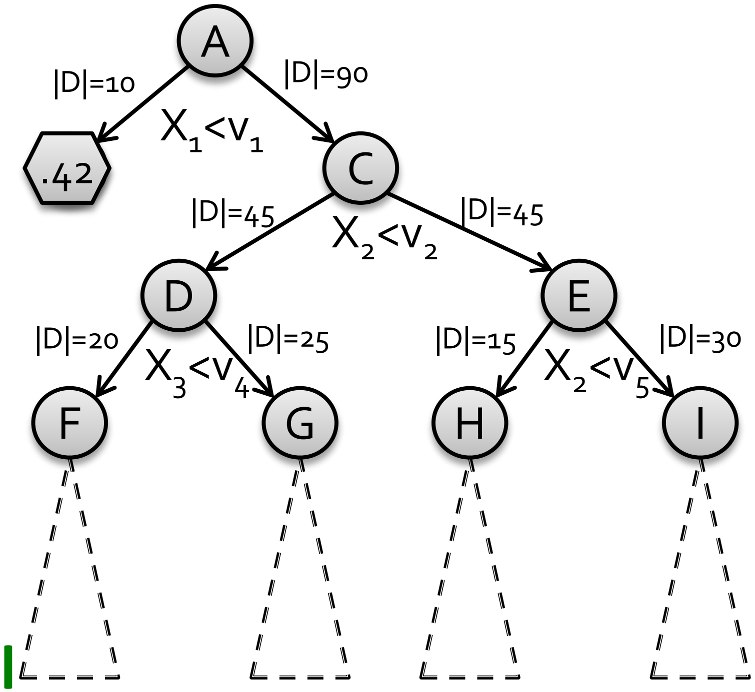
$$s.t. \forall i, y_i(w \cdot x_i + b) \geq 1 - \xi_i$$



- Find **w,b** using **Stochastic gradient descent**

Decision Trees

- **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**

$\quad n \rightarrow$left_prediction=FindPrediction(D_L)

else

$\quad$ **FindBestSplit**$(n \rightarrow \text{left}, D_L)$

if StoppingCriteria(D_R) **then**

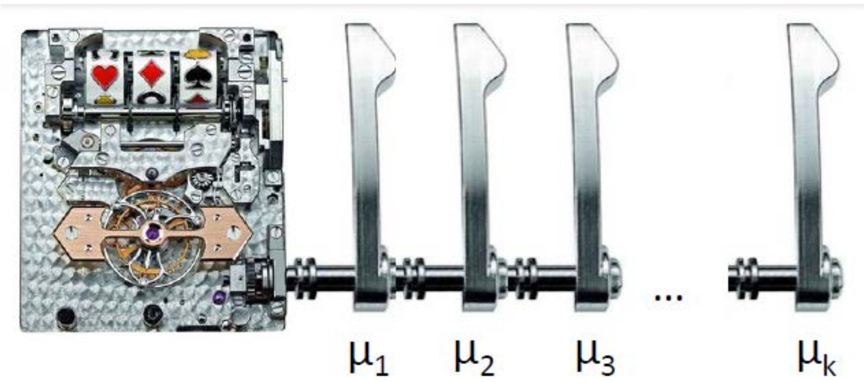
$\quad n \rightarrow$right_prediction=FindPrediction(D_R)

else

$\quad$ **FindBestSplit**$(n \rightarrow \text{right}, D_R)$

Learning Through Experimentation

- **Learning through experimentation**
  - **Exploration-Exploitation tradeoff**
  - **Regret**
- **Multiarmed Bandits**
  - **Epsilon-Greedy**
  - **UCB1 algorithm**
- **Submodular function optimization**
  - **Coverage**
  - Greedy and Lazy-Greedy algorithms
  - Multiplicative Weights algorithm

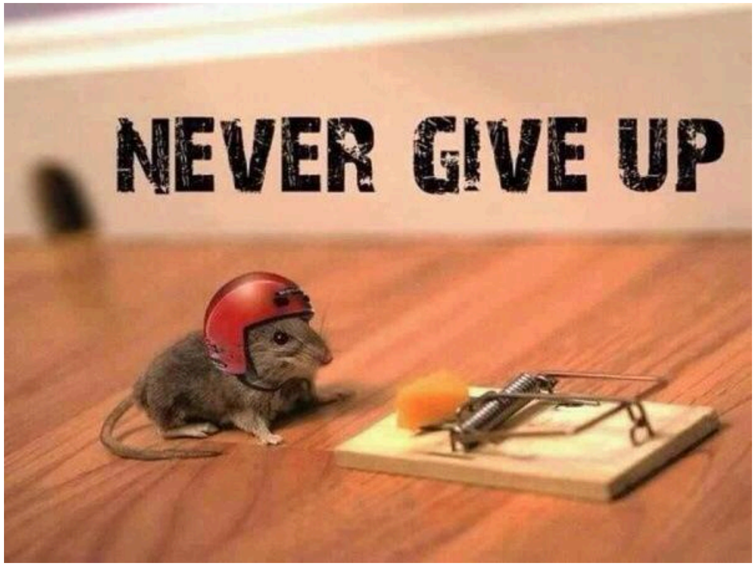


When to use which method?

- **SVM:** Classification
  - **Millions of sparse numerical features** (e.g., documents)
  - Simple (linear) decision boundary
  - Somewhat hard to interpret model
- **k-NN:** Classification or regression
  - (Many) numerical features
  - Many design decisions – distance metric, ***k***, weighting, … **there is no simple way to set them!**
- **Decision Trees:** Classification or Regression
  - Relatively few dense features (handles categorical features)
  - Complicated decision boundary: **Overfitting!**
  - **Easy to explain/interpret the classification**
  - **Bagged Decision Trees** – very, very hard to beat!
- **Bandits:** Learning through experimentation
  - Exploration-Exploitation tradeoff

What if "ML alg. doesn't work"?

- **Over- vs. under-fitting**
  - Compare error on the train/test set
  - Plot error vs. (regularization) parameter
- **Debugging:**
  - Compare performance to a simple baseline
  - Build synthetic datasets for which you know your method should work
- **Think about:**
  - The prediction problem
  - Error metrics
  - Model assumptions
  - Properties of the data



If the ML algorithm doesn't work

- **Get more training data**
  - Sometimes more data doesn't help but often it does
- **Try a smaller set a features**
  - Carefully select small subset
  - You can do this by hand, or use SVD
- **Try getting additional features**
  - **LOOK** at the data
  - Can be very time consuming
- **Adding polynomial features**
  - Include $\mathbf{x}$ and $\mathbf{x}^2$ as features
- **Building your own, new, better features**
  - Based on your knowledge of the problem
- **Try decreasing or increasing regularization parameter**
  - Change how important the regularization term is

How it all fits together?

Data is high-dimensional:

- Locality Sensitive Hashing
- Dimensionality reduction
- Clustering

The data is a graph:

- Link Analysis: PageRank, TrustRank, Hubs & Authorities

Data is labeled (Machine Learning):

- kNN, Perceptron, SVM, Decision Trees

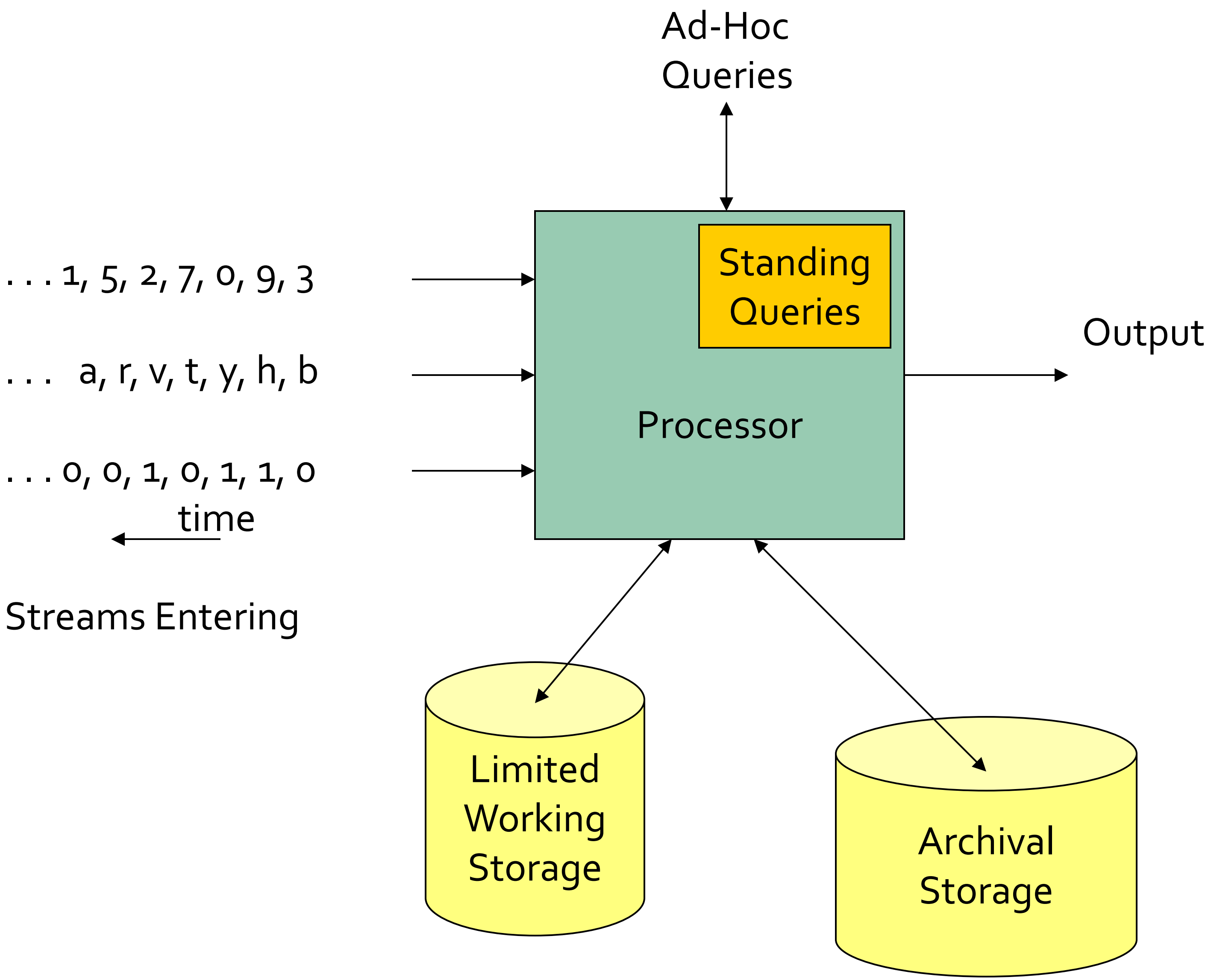
Data is infinite:

- Mining data streams
- Advertising on the Web

Applications:

- Association Rules
- Recommender systems

Mining Data Streams



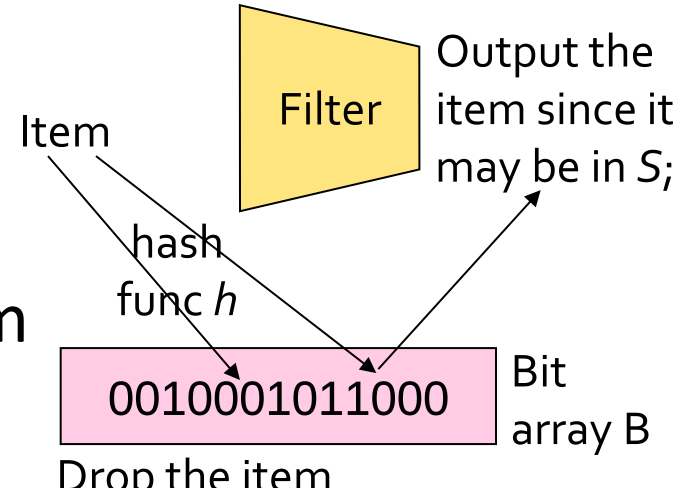
Problems on data streams

- **Sampling data from a stream:**
  - Each element is included with prob. **k/N**
- **Queries over sliding windows:**

How many **1**s are in last ***k*** bits?

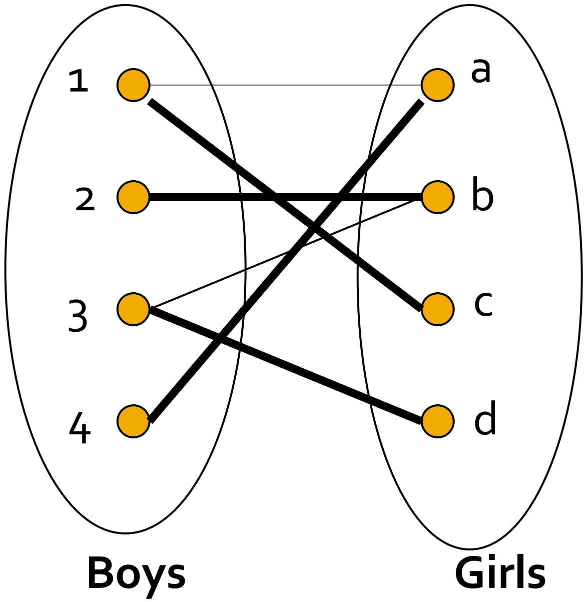
100101011000101101010101010101011010101010101111010101011101010001011001 0

- **Filtering a stream: Bloom filters**
  - Filter elements with property ***x***
- **Counting distinct elements:**
  - Number of distinct elements in the last ***k*** elements of the stream
- **Estimating moments**



Online algorithms & Advertising

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



How it all fits together?

Data is high-dimensional:

- Locality Sensitive Hashing
- Dimensionality reduction
- Clustering

The data is a graph:

- Link Analysis: PageRank, TrustRank, Hubs & Authorities

Data is labeled (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

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

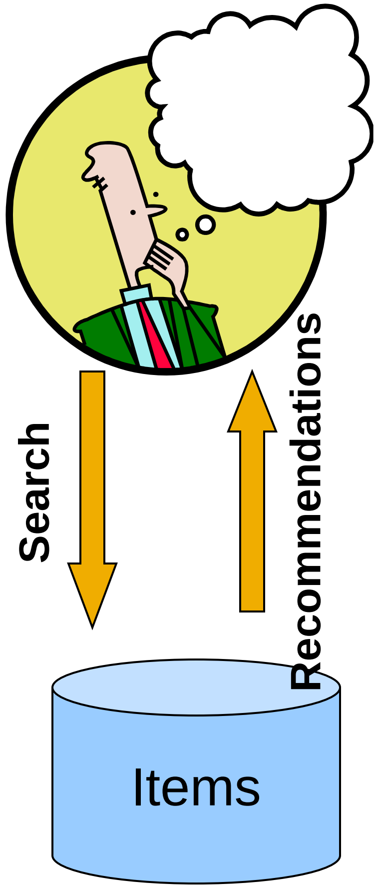
Discovering frequent items: A-priori, PCY

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}

Recommender Systems

- **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
- **Profile based**

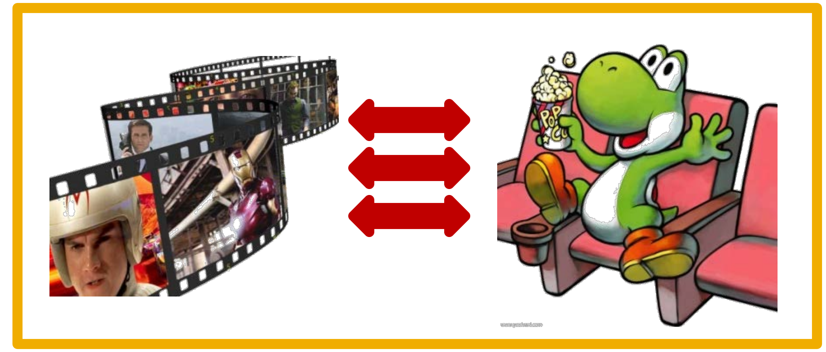


Latent Factor Models: Netflix

user bias

movie bias

user-movie interaction



Baseline predictor

- Separates users and movies
- Benefits from insights into user's behavior

User-Movie interaction

- Characterizes the matching between users and movies
- Attracts most research in the field

$$\min_{Q,P} \sum_{(u,i)\in R} \left(r_{ui} - (\mu + b_u + b_i + q_i\, p_u^T)\right)^2$$

$$+ \lambda \left(\sum_i \|q_i\|^2 + \sum_u \|p_u\|^2 + \sum_u \|b_u\|^2 + \sum_i \|b_i\|^2 \right)$$

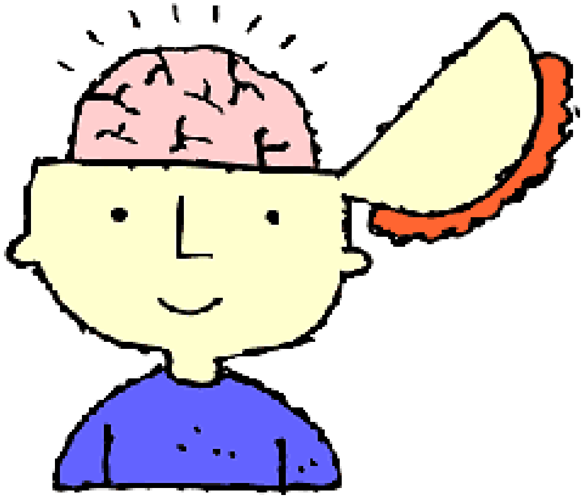
When to use which method?

- **Lots of rating data: CF**
  - Easy to tweak, easy to add lots of features/signals
  - Use optimization to learn weights on how to combine features
- **Lots^2 of rating data: CF + Latent factors**
  - Many ratings per user, many ratings per item
  - Depending on the amount of data make the model more/less complex (more/less parameters)
- **Cold start, little data: Profile based**
  - Need to have good user/item features and similarity metric

In closing…

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



Map of Superpowers

High dim. data	Graph data	Infinite data	Machine learning	Apps
Locality sensitive hashing	PageRank, SimRank	Filtering data streams	SVM	Recommen der systems
Clustering	Community Detection	Web advertising	Decision Trees	Association Rules
Dimensional ity reduction	Spam Detection	Queries on streams	Perceptron, kNN	Duplicate document detection

Applying Your Superpowers

THE BIG PICTURE

- **How to analyze large datasets to discover models and patterns 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

What next? Seminars

- **Seminars:**
  - InfoSeminar: http://i.stanford.edu/infoseminar
  - RAIN Seminar: http://rain.stanford.edu
- **Conferences:**
  - **KDD:** ACM Conf. on Knowledge Discovery & Data Mining
  - **WSDM:** ACM Conf. on Web Search and Data Mining
  - **ICDM:** IEEE International Conf. on Data Mining
  - **WWW:** World Wide Web Conference
  - **ICML:** International Conf. on Machine Learning
  - **VLDB:** Very Large Data Bases

CS341: Project in Data Mining

- **Data mining research project on real data**
  - Groups of 3 students
  - **We provide interesting data, computing resources (Amazon EC2) and mentoring**
  - **You provide project ideas**
  - There are (practically) no lectures, only individual group mentoring

Information session:
Today 6pm in Gates 415
(there will be pizza)

What Next? Courses

- **Other relevant courses**
  - **CS224W**: Social and Information Network Analysis
  - **CS276**: Information Retrieval and Web Search
  - **CS229**: Machine Learning
  - **CS245**: Database System Principles
  - **CS347**: Distributed Databases
  - **CS448g**: Interactive Data Analysis

What Next? Final Exam

In Closing

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

Thank You for the Hard Work!!!