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

Map Task 2

Map Task 3

Partitioning Function

Sort and Group

Reduce Task 1

Reduce Task 2
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 bot 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**

<table>
<thead>
<tr>
<th>(key, value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(the, 1)</td>
</tr>
<tr>
<td>(crew, 1)</td>
</tr>
<tr>
<td>(of, 1)</td>
</tr>
<tr>
<td>(the, 1)</td>
</tr>
<tr>
<td>(space, 1)</td>
</tr>
<tr>
<td>(shuttle, 1)</td>
</tr>
<tr>
<td>(Endeavor, 1)</td>
</tr>
<tr>
<td>(recently, 1)</td>
</tr>
</tbody>
</table>

**Provided by the programmer**

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

<table>
<thead>
<tr>
<th>(key, value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(crew, 1)</td>
</tr>
<tr>
<td>(crew, 1)</td>
</tr>
<tr>
<td>(space, 1)</td>
</tr>
<tr>
<td>(the, 1)</td>
</tr>
<tr>
<td>(the, 1)</td>
</tr>
<tr>
<td>(the, 1)</td>
</tr>
<tr>
<td>(shuttle, 1)</td>
</tr>
<tr>
<td>(recently, 1)</td>
</tr>
</tbody>
</table>

**Provided by the programmer**

**Group by key:**
Collect all pairs with same key

<table>
<thead>
<tr>
<th>(key, value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(crew, 2)</td>
</tr>
<tr>
<td>(space, 1)</td>
</tr>
<tr>
<td>(the, 3)</td>
</tr>
<tr>
<td>(shuttle, 1)</td>
</tr>
<tr>
<td>(recently, 1)</td>
</tr>
</tbody>
</table>

**Reduce:**
Collect all values belonging to the key and output

<table>
<thead>
<tr>
<th>(key, value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(crew, 1)</td>
</tr>
<tr>
<td>(crew, 1)</td>
</tr>
<tr>
<td>(space, 1)</td>
</tr>
<tr>
<td>(the, 1)</td>
</tr>
<tr>
<td>(the, 1)</td>
</tr>
<tr>
<td>(the, 1)</td>
</tr>
<tr>
<td>(shuttle, 1)</td>
</tr>
<tr>
<td>(recently, 1)</td>
</tr>
</tbody>
</table>
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
Many problems can be expressed as finding “similar” sets:

- Find near-neighbors in high-D space

Distance metrics:
- Points in $\mathbb{R}^n$: 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.

**Document**

- **Shingling**: The set of strings of length $k$ that appear in the document.

**Minhashing**

- **Signatures**: short integer vectors that represent the sets, and reflect their similarity.

**Locality-sensitive Hashing**

- **Candidate pairs**: those pairs of signatures that we need to test for similarity.
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.:
  \[
  \Pr[h_\pi(S(D_1)) = h_\pi(S(D_2))] = Sim(S(D_1), S(D_2))
  \]
  - For Jaccard use permutation of columns and index of first 1.
# Min Hashing – Example

## Input matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>4</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

## Signature matrix $M$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

## Similarities:

<table>
<thead>
<tr>
<th></th>
<th>1-3</th>
<th>2-4</th>
<th>1-2</th>
<th>3-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Col/Col</td>
<td>0.75</td>
<td>0.75</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sig/Sig</td>
<td>0.67</td>
<td>1.00</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
**LSH**

- **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

- 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

<table>
<thead>
<tr>
<th>$s$</th>
<th>$1-(1-s^r)^b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>.2</td>
<td>.006</td>
</tr>
<tr>
<td>.3</td>
<td>.047</td>
</tr>
<tr>
<td>.4</td>
<td>.186</td>
</tr>
<tr>
<td>.5</td>
<td>.470</td>
</tr>
<tr>
<td>.6</td>
<td>.802</td>
</tr>
<tr>
<td>.7</td>
<td>.975</td>
</tr>
<tr>
<td>.8</td>
<td>.9996</td>
</tr>
</tbody>
</table>
(2) Dimensionality Reduction

\[ A \approx U \Sigma V^T = \sum_i \sigma_i u_i \odot v_i \]
- **A = UΣVT - example:**

  
  ![User-to-concept similarity matrix](image)

  
  \[
  \begin{pmatrix}
  1 & 1 & 1 & 0 & 0 \\
  2 & 2 & 2 & 0 & 0 \\
  1 & 1 & 1 & 0 & 0 \\
  5 & 5 & 5 & 0 & 0 \\
  0 & 0 & 0 & 2 & 2 \\
  0 & 0 & 0 & 3 & 3 \\
  0 & 0 & 0 & 1 & 1 \\
  \end{pmatrix}
  \]

  \[
  \begin{pmatrix}
  0.18 & 0 \\
  0.36 & 0 \\
  0.18 & 0 \\
  0.90 & 0 \\
  0 & 0.53 \\
  0 & 0.80 \\
  0 & 0.27 \\
  \end{pmatrix}
  \]

  \[
  \begin{pmatrix}
  9.64 & 0 \\
  0 & 5.29 \\
  \end{pmatrix}
  \]

  \[
  \begin{pmatrix}
  0.58 & 0.58 & 0.58 & 0 & 0 \\
  0 & 0 & 0 & 0.71 & 0.71 \\
  \end{pmatrix}
  \]
\[ A = U \Sigma V^T \] - example:

\[
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27
\end{bmatrix}
\begin{bmatrix}
9.64 & 0 \\
0 & 5.29
\end{bmatrix}
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71
\end{bmatrix}
\]


- **A = UΣVT** - example:

  ![Movie-to-concept similarity matrix](image)

  - **SciFi**
  - **Romance**
  - **Matrix**
  - **Alien**
  - **Serenity**
  - **Casablanca**
  - **Amelie**

  The similarity matrix is calculated using the formula **A = UΣVT**.
How to do dimensionality reduction:
- Set small singular values to zero

How to query?
- Map query vector into “concept space” –
  - How? Compute $q \cdot V$

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

Clusters after first round

Reassigned points
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 surferer has two options:
    - With probability $\beta$, follow a link at random
    - With prob. $1-\beta$, 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: $\frac{|r(p) - r^+(p)|}{r(p)}$

- **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
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
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:**
  - \( w_i = \exp\left(-\frac{d(x_i, q)^2}{K_w}\right) \)
    - Nearby points to query \( q \) are weighted more strongly. \( K_w \)...kernel width.

- **How to fit with the local points?**
  - Predict weighted average: \( \frac{\sum w_i y_i}{\sum w_i} \)
Support Vector Machines

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

\[
\min_{w,b,\xi_i \geq 0} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \xi_i \\
\text{s.t. } \forall i, y_i (w \cdot x_i + b) \geq 1 - \xi_i
\]

- Find \( w, b \) using Stochastic gradient descent
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 $D \subseteq D^*$
1: ($n \rightarrow$split, $D_L, D_R$) = FindBestSplit($D$)
2: if StoppingCriteria($D_L$) then
3:   $n \rightarrow$left_prediction = FindPrediction($D_L$)
4: else
5:   FindBestSplit($n \rightarrow$left, $D_L$)
6: if StoppingCriteria($D_R$) then
7:   $n \rightarrow$right_prediction = FindPrediction($D_R$)
8: else
9:   FindBestSplit($n \rightarrow$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

Streams Entering

- 1, 5, 2, 7, 0, 9, 3
- a, r, v, t, y, h, b
- 0, 0, 1, 0, 1, 1, 0

Processor

Ad-Hoc Queries

Output

Standing Queries

Limited Working Storage

Archival Storage
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
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 \( R = \max_a r(a) \) 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 = \( \min_{\text{all inputs}} \left( \frac{|M_{\text{my_alg}}|}{|M_{\text{opt}}|} \right) \)
- Greedy online matching:
  competitive ratio = \( \frac{|M_{\text{greedy}}|}{|M_{\text{opt}}|} \geq 1/2 \)
- 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:

- 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: xxxxyyyyy
- BALANCE choice: ABABBB__
  - Optimal: AAAABBBBB, Competitive ratio = \(\frac{3}{4}\)
- Generally, competitive ratio = \(1-\frac{1}{e}\)
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
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

<table>
<thead>
<tr>
<th>TID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bread, Coke, Milk</td>
</tr>
<tr>
<td>2</td>
<td>Beer, Bread</td>
</tr>
<tr>
<td>3</td>
<td>Beer, Coke, Diaper, Milk</td>
</tr>
<tr>
<td>4</td>
<td>Beer, Bread, Diaper, Milk</td>
</tr>
<tr>
<td>5</td>
<td>Coke, Diaper, Milk</td>
</tr>
</tbody>
</table>

Rules Discovered:

- \{Milk\} --> \{Coke\}
- \{Diaper, Milk\} --> \{Beer\}
Apriori algorithm

- **Observation:** Subsets of a frequent itemset are frequent.
- **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:

  - Item profiles
    - Red
    - Circles
    - Triangles
  - User profile
  - recommend
  - likes
  - match
  - build
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...
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
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](http://i.stanford.edu/infoseminar)
  - RAIN Seminar: [http://rain.stanford.edu](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!!!