Matrix Sketching in Data Streams

CS246: Mining Massive Datasets
Jure Leskovec, Stanford University
Mina Ghashami, Amazon
http://cs246.stanford.edu
Data as a Matrix

In many applications, we can represent data as a matrix: e.g. text analysis, recommendation

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D4</th>
<th>D5</th>
</tr>
</thead>
<tbody>
<tr>
<td>complexity</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>algorithm</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>entropy</td>
<td>1</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>traffic</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>network</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Term-document matrix
Data as a Matrix

- Think of data as $A \in \mathbb{R}^{n \times d}$ containing $n$ row vectors in $\mathbb{R}^d$, and typically $n \gg d$

- Some examples of typical web-scale data:

<table>
<thead>
<tr>
<th>Data</th>
<th>Rows</th>
<th>Columns</th>
<th>$n$</th>
<th>$d$</th>
<th>sparse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Textual</td>
<td>Documents</td>
<td>Words</td>
<td>$&gt; 10^{10}$</td>
<td>$10^5 - 10^7$</td>
<td>yes</td>
</tr>
<tr>
<td>Visual</td>
<td>Images</td>
<td>Pixels, SIFT</td>
<td>$&gt; 10^8$</td>
<td>$10^5 - 10^6$</td>
<td>no</td>
</tr>
<tr>
<td>Audio</td>
<td>Songs</td>
<td>Frequencies</td>
<td>$&gt; 10^8$</td>
<td>$10^5 - 10^6$</td>
<td>no</td>
</tr>
<tr>
<td>Machine Learning</td>
<td>Examples</td>
<td>Features</td>
<td>$&gt; 10^6$</td>
<td>$10^2 - 10^4$</td>
<td>yes/no</td>
</tr>
<tr>
<td>Financial</td>
<td>Prices</td>
<td>Items, Stocks</td>
<td>$&gt; 10^6$</td>
<td>$10^3 - 10^5$</td>
<td>no</td>
</tr>
</tbody>
</table>
Review: rank-k approximation

- Rank-k approximation to $A$ computes a smaller matrix $B$ of rank $k$ such that $B$ approximates $A$.

**Rank-k Approximation**

Given $A \in \mathbb{R}^{n \times d}$ with $\text{rank}(A) = r$, compute a concise matrix $B$ with rank $k \ll r$ such that it approximates $A$ "accurately".
Review: rank-k approximation

- Rank-k approximation to $A$ computes a smaller matrix $B$ of rank $k$ such that $B$ approximates $A$

<table>
<thead>
<tr>
<th>Rank-k Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given $A \in \mathbb{R}^{n \times d}$ with $\text{rank}(A) = r$, compute a concise matrix $B$ with rank $k \ll r$ such that it approximates $A$ “accurately”.</td>
</tr>
</tbody>
</table>

- B is much smaller than A that it fits in memory
- $\text{Rank}(B) \ll \text{rank}(A)$
  - If $A$ is a document-term matrix with 10 billion documents and 1 million words $A \in \mathbb{R}^{10^{10} \times 10^6}$ then $B$ would probably be $B \in \mathbb{R}^{1000 \times 106}$
Review: rank-k approximation

- Rank-k approximation to $A$ computes a smaller matrix $B$ of rank $k$ such that $B$ approximates $A$ accurately.

---

**Rank-k Approximation**

Given $A \in \mathbb{R}^{n \times d}$ with $\text{rank}(A) = r$, compute a concise matrix $B$ with rank $k \ll r$ such that it approximates $A$ "accurately".
Rank-k approximation to $A$ computes a smaller matrix $B$ of rank $k$ such that $B$ approximates $A$.

**Rank-k Approximation**

Given $A \in \mathbb{R}^{n \times d}$ with $\text{rank}(A) = r$, compute a concise matrix $B$ with rank $k \ll r$ such that it approximates $A$ "accurately".

Error difference between $A$ and $B$ is small:
Rank-k approximation to $A$ computes a smaller matrix $B$ of rank $k$ such that $B$ approximates $A$

Error difference between $A$ and $B$ is small:
- The covariance error $\|A^TA - B^TB\|_2$ is small
Review: rank-k approximation

- **Rank-k approximation** to \( A \) computes a smaller matrix \( B \) of rank \( k \) such that \( B \) approximates \( A \)

**Rank-k Approximation**

Given \( A \in \mathbb{R}^{n \times d} \) with \( \text{rank}(A) = r \), compute a concise matrix \( B \) with rank \( k \ll r \) such that it approximates \( A \) “accurately”.

- **Error difference between \( A \) and \( B \) is small:**
  - The **covariance error** \( \|A^T A - B T B\|_2 \) is small
  - The **projection error** \( \|A - \Pi_B A\|_{2,F} \) is small
    - \( \Pi_B A \) := projecting rows of \( A \) onto the subspace of \( B \)
    - If \( B = USV^T \) then, the subspace of \( B \) is \( VV^T \)
    - Therefore \( \Pi_B A = AVV^T \)
Best Rank-\(k\) Approximation

- We saw that SVD computes the **best** rank-\(k\) approximation to \(A\)

\[
A = U S V^T
\]

- \(A\) is an \(n\times d\) matrix
- \(U\) is an \(n\times n\) orthogonal matrix
- \(S\) is a \(n\times d\) diagonal matrix with singular values
- \(V^T\) is a \(d\times d\) orthogonal matrix

right singular vectors

left singular vectors

singular values
SVD computes the best rank-k approximation to $A$

$$A_k = \underset{\text{rank}(B) \leq k}{\text{arg min}} \| A - B \|_{F,2}$$

We compare error of other algorithms to $\|A - A_k\|$ as it is the smallest error.
Best Rank-k Approximation

- SVD computes the **best** rank-k approximation to $A$
- SVD requires $O(nd^2)$ time and $O(nd)$ space
- Not applicable in streaming, or distributed settings
- Not efficient for sparse matrices
Can we compute \textit{rank-k approximation} in streaming setting?
Streaming matrix sketching
Every element of the stream is a row vector of fixed $d$-dimension.

We’d like to process $A$ in one pass and using a small amount of memory (sublinear in $n$)
Streaming data matrix

- Streaming data such as any time series data:
  - ecommerce purchases
  - Traffic sensors
  - Activity logs

- We can not store the entire data

---

Application of rank-k approximations

- A large set of data analysis tasks rely on obtaining a **low rank approximation**:
  - Dimension reduction
  - Anomaly detection
  - Data denoising
  - Clustering
  - Recommendation systems
Sketch of a Streaming Matrix

- B is a **sketch** of a streaming matrix A iff
  - B is of a fixed **small size** that fits in memory
  - At any point in stream, B approximates A
Almost any matrix sketching methods in streaming setting falls into one of these categories:

1. Row sampling based
2. Random projection based and Hashing
3. Iterative sketching

They compute a significantly smaller sketch matrix B such that $A \approx B$ or $A^T A \approx B^T B$
Row Sampling Methods
Row Sampling Methods

- They select a subset of "important" rows of the original matrix $A$
  - Sampling is done w.r.t a well-defined probability distribution
  - Often sampling is done with replacement

- And show that sampled matrix $B$ is a good approximation to original one

- Methods differ in how they define notion of "importance"
They construct sketch B by:

- assign a probability $p_i$ to each row $a_i$
- sample $l$ rows from A to construct B
- rescale B appropriately to make it unbiased
An Intuitive way to define “importance” of an item:

- the weight associated to the item, e.g.
  - file records → weights as size of the file,
  - IP addresses → weights as number of times the IP address makes a request

why it is necessary to sample important items?

Consider a set of weighted items \( S = \{(a_1, w_1), (a_2, w_2), \ldots, (a_n, w_n)\} \) that we want to summarize with a small & representative sample.

We define a representative sample as the one estimates total weight of \( S \) (i.e. \( W_s = \sum_{i=1}^{n} w_i \)) in expectation.
This is achievable with a sample set of size one!
- we sample any item \((a_j, w_j)\) with an arbitrary fixed probability \(p\),
- and rescale it to have weight \(W_s/p\).

This sample set has total weight \(W_s\) in expectation
- but has a large variance too
- To lower down the variance, it is necessary to allow heavy items (i.e. important items) to get sampled with higher probability
Row Sampling algorithms

- Row sampling based on L2 norm:
  - Sample with probability \( p_i = \frac{\|a_i\|^2}{\|A\|_F^2} \)
  - Rescale rows of B by \( \frac{1}{\sqrt{l} \cdot p_i} \)
  - We can show that \( E[\|B\|_F] = \|A\|_F \)
  - And it is proved that if we sample \( \ell = O(k/\varepsilon^2) \) rows, then:

\[
\|A - \pi_B(A)\|_F^2 \leq \|A - A_k\|_F^2 + \varepsilon \|A\|_F^2
\]

CUR: Row/column sampling

- Row sampling based on L2 norm:
  - CUR method: samples rows/columns with probability = squared norm of rows/columns

\[
\begin{pmatrix}
A
\end{pmatrix}
\approx
\begin{pmatrix}
C
\end{pmatrix}
\cdot
\begin{pmatrix}
U
\end{pmatrix}
\cdot
\begin{pmatrix}
R
\end{pmatrix}
\]
CUR: Row/column sampling

- Row sampling based on L2 norm:
  - CUR method: samples rows/columns with probability = squared norm of rows/columns

\[
\begin{align*}
\begin{pmatrix}
A \\
\end{pmatrix}
\approx
\begin{pmatrix}
C \\
\end{pmatrix}
\cdot
\begin{pmatrix}
U \\
\end{pmatrix}
\cdot
\begin{pmatrix}
R \\
\end{pmatrix}
\end{align*}
\]

Pseudo-inverse of the intersection of \(C\) and \(R\)
CUR: Row/column sampling

- Row sampling based on L2 norm:
  - CUR method: samples rows/columns with probability = squared norm of rows/columns

- Error guarantee: If we sample \( c = O\left(\frac{k \log k}{\varepsilon^2}\right) \) columns and \( r = O\left(\frac{k \log k}{\varepsilon^2}\right) \) rows, then

\[
\begin{align*}
\text{CUR error} & \quad \left\| A - CUR \right\|_F \leq (2 + \varepsilon) \left\| A - A_K \right\|_F \\
\text{SVD error} & \quad \left\| A - CUR \right\|_F \leq (2 + \varepsilon) \left\| A - A_K \right\|_F
\end{align*}
\]

With probability >= 98%
Row Sampling Methods

+ Easy interpretation of basis
  • Since the basis vectors are actual rows/columns

+ Suitable for Sparse data
  • Since the basis vectors are actual rows/columns

- Duplicate columns and rows
  • Columns of large norms will be sampled multiple times
Random Projection Methods
Random Projection Methods

- **Key idea**: if points in a vector space are projected onto a randomly selected subspace of suitably high dimension, then the distances between points are approximately preserved.

- **Johnson-Lindenstrauss Transform (JLT)**: $d$ datapoints in any dimension (\( \mathbb{R}^n \) for \( n \gg d \)) can get embedded into roughly \( \log d \) dimensional space, such that their pair-wise distances are preserved to some extent.
We define JLT more precisely:

- A random matrix $S \in \mathbb{R}^{r \times n}$ has JLT property if for all vectors $v, v' \in \mathbb{R}^n$,
  $$
  \|Sv - Sv'\|^2 = (1 \pm \epsilon)\|v - v'\|^2
  $$
  with probability at least $1 - \delta$

- There are many ways to construct a matrix $S$ that preserve pair-wise distances.
  - All such matrices are called to have the Johnson-Lindenstrauss Transform (JLT) property
How to construct a JLT matrix

One simple construction of S:

- Pick matrix \( S \in \mathbb{R}^{r \times n} \) as an orthogonal projection on a random \( r \)-dimensional subspace of \( \mathbb{R}^n \) with \( r = O(\epsilon^{-2} \log d) \)
  - Rows of S are orthogonal vectors

- Then for any matrix \( A \in \mathbb{R}^{n \times d} \), \( SA \) preserves pair-wise distances between \( d \) datapoints in A
How to construct a JLT matrix

- A simpler construction for $S \in \mathbb{R}^{r \times n}$ is:
  - to have entries as independent random variables with the standard normal distribution

$$S = \sqrt{\frac{1}{r}} \mathbf{X}$$

[matrix with entries draw from $N(0,1)$]

[Diagram showing the construction of $S$]
Another construction for $S \in \mathbb{R}^{r \times n}$ is:

$$S = \sqrt{\frac{1}{r}} \text{[entries as independent +/- 1 random var]}$$

This is computationally simpler to construct.
They use a JLT matrix $S \in \mathbb{R}^{r \times n}$

Construct the sketch as $B = SA \in \mathbb{R}^{r \times d}$

- this projects datapoints from a high-dim space $\mathbb{R}^n$ onto a lower-dim subspace $\mathbb{R}^r$

They show $\mathbb{E}[B^T B] = A^T \mathbb{E}[S^T S] A = A^T A$
Random Projection Methods

- Depending on JLT construction, we achieve different error bounds:
  - If $S \in \mathbb{R}^{r \times n}$ has iid zero-mean $\pm 1$ entries and $r = O\left(\frac{k}{\varepsilon} + k \log k\right)$ and, then
    \[
    \|A - \pi_{SA}(A)\|_F \leq (1 + \varepsilon)\|A - A_k\|_F
    \]
Random Projection Methods

- Computationally efficient
- Sufficiently accurate in practice
- A great pre-processing step in applications

- **Data-oblivious** as their computation involves only a random matrix $S$
  - Compare to row sampling methods that need to access data to form a sketch
Matrix Hashing Techniques

- Use matrix S that contains one $\pm 1$ per column.

Only one non-zero entry in each column of S. The rest of entries are zero.

- To build S, use two hash functions:
  - $h: [n] \rightarrow [r]$, and $g: [n] \rightarrow \{-1, +1\}$
Matrix Hashing Techniques

- Very efficient for sparse matrices A
  - can be applied in $O(\text{nnz}(A))$ operations
  - $\text{nnz}(A) =$ number of non-zeros of A
Iterative Sketching
Iterative Sketching

- They work over a stream $A = \langle a_1, a_2, \ldots, a_n \rangle$
- each $a_i$ is read once, get processed quickly and not read again
- with only a small amount of memory available
Iterative Sketching

- State of the art method in this group is called “Frequent Directions”
- It is based on Misra-Gries algorithm for finding frequent items in a data stream
- We first see how Misra-Gries algorithm for finding frequent items work
  - Then we extend it to matrices
Suppose there is a stream of items, and we want to find frequency $f(i)$ of each item.
If I keep \( d \) counters, I can count frequency of every item...

- But it’s not good enough (IP addresses, queries,...)
Frequent Items: Misra-Gries

- Let’s keep $l$ counters where $l \ll d$
If a new item arrives in the stream that is already in the counters, we add 1 to its count.
If the new item is not in the counters and we have space, we create a counter for it and set it to 1.
Frequent Items: Misra-Gries

- But what if we don’t have space for it?
Let $\delta$ be the median counter at time $t$. 

\[ \delta = f_{e/2} = 2 \]
Decrease all counters by $\delta$ (or set it to zero if less than $\delta$)
Now we have space for new item, so we continue...
Frequent Items: Misra-Gries

- At any time in the stream, the approximated counts for items are what we have kept so far.
Frequent Items: Misra-Gries

- This method undercounts
  \[ 0 \leq f'(i) \leq f(i) \]
- We decrease each counter by at most \( \delta_t \)
  \[ f'(i) \geq f(i) - \sum \delta_t \]
- At any point that we have seen \( n \) elements in stream:
  \[ \frac{l}{2} \sum \delta_t \leq n \]
- The error guarantee:
  \[ 0 \leq f(i) - f'(i) \leq 2n/l \]
Frequent Items: Misra-Gries

- Misra-Gries produces a non-zero approximated frequency $f'(i)$ for all items that their true frequency $f(i)$ is higher than $2n/l$,
- $f(i) - 2n/l \leq f'(i)$

- To find items that appear more than 20% of the time i.e. $f(i) > n/5$, take $l = 10$ counters and run Misra-Gries algo.
Let’s extend it to vectors and matrices

Stream items are row vectors in \( d \) dimension

At any time \( n \) in the stream, they form a tall matrix \( A \in \mathbb{R}^{n \times d} \)

The goal is to find the most frequent directions of \( A \)
**Frequent Directions**

**Input:** \( A \in \mathbb{R}^{n \times d} \), and an integer \( \ell \)

\( B \leftarrow \text{empty matrix} \in \mathbb{R}^{\ell \times d} \)

**for** \((a_i \in A)\)

- Insert \( a_i \) into \( B \)
- If \((B \text{ is full})\)
  - \([U, S, V] \leftarrow \text{svd}(B)\)
  - \( \tilde{S} \leftarrow [\sqrt{S_1^2 - S_{l/2}^2}, \sqrt{S_2^2 - S_{l/2}^2}, \ldots, 0, \ldots, 0] \)
  - \( B \leftarrow \tilde{S} V^T \)

**return** \( B \)
Frequent Directions

**Input:** $A \in \mathbb{R}^{n \times d}$, and an integer $\ell$

- $B \leftarrow$ empty matrix $\in \mathbb{R}^{\ell \times d}$
- **for** ($a_i \in A$)
  - Insert $a_i$ into $B$
- **if** ($B$ is full)
  - $[U, S, V] \leftarrow \text{svd}(B)$
  - $\tilde{S} \leftarrow [\sqrt{S_1^2 - S_{i/2}^2}, \sqrt{S_2^2 - S_{i/2}^2} ... 0, ..., 0]$ 
  - $B \leftarrow \tilde{S} V^T$

**return** $B$
Frequent Directions

\textbf{Input:} \( A \in \mathbb{R}^{n \times d} \), and an integer \( \ell \)

\( B \leftarrow \) empty matrix \( \in \mathbb{R}^{\ell \times d} \)

\textbf{for} \((a_i \in A)\)

\quad \text{Insert} \ a_i \ \text{into} \ B

\quad \textbf{if} \ (B \ \text{is full})

\quad \quad \left[ U, S, V \right] \leftarrow \text{svd}(B)

\quad \quad \tilde{S} \leftarrow \left[ \sqrt{S_1^2 - S_{\ell/2}^2}, \ \sqrt{S_2^2 - S_{\ell/2}^2} \ \ldots \ 0, \ \ldots, \ 0 \right]

\quad \quad B \leftarrow \tilde{S} V^T

\textbf{return} \ B
**Frequent Directions** (Lib’13)

**Input:** $A \in \mathbb{R}^{n \times d}$, and an integer $\ell$

$B \leftarrow$ empty matrix $\in \mathbb{R}^{\ell \times d}$

for($a_i \in A$)

Insert $a_i$ into $B$

if ($B$ is full)

$[U, S, V] \leftarrow \text{svd}(B)$

$\tilde{S} \leftarrow [\sqrt{S_1^2 - S_{l/2}^2}, \sqrt{S_2^2 - S_{l/2}^2} ... 0, ..., 0]$

$B \leftarrow \tilde{S}V^T$

return $B$
Frequent Directions (Lib’13)

**Input:** $A \in \mathbb{R}^{n \times d}$, and an integer $\ell$

$B \leftarrow$ empty matrix $\in \mathbb{R}^{\ell \times d}$

**for** $(a_i \in A)$

- Insert $a_i$ into $B$

**if** ($B$ is full)

- $[U, S, V] \leftarrow \text{svd}(B)$

- $\tilde{S} \leftarrow [\sqrt{S_1^2 - S_{i/2}^2}, \sqrt{S_2^2 - S_{i/2}^2} \ldots 0, \ldots, 0]$

- $B \leftarrow \tilde{S}V^T$

**return** $B$
Frequent Directions

- Similar to the frequent items case, this method has the following error guarantee:

\[ \| A^T A - B TB \| \leq \frac{2}{l} \| A \|_F^2 \]

- More accurate error bounds:

\[ \| A - \pi_B (A) \|_F^2 \leq (1 + \varepsilon) \| A - Ak \|_F^2 \]
Sketching in Experiment

\[ \text{cov-err} := \frac{\|A^T A - B^T B\|_2^2}{\|A\|_F^2} \]

- Random Projections
- Hashing
- Sampling
- FrequentDirections
- Naive
- Brute Force

[Sarlos FOCS06]
[Clarkson+Woodruff STOC13]
[Drineas, Kannan, Mahoney SIAM JoC06]

Sketch Size

Covariance Error

[all 0s]
[SVD]
Sketching in Experiment

\[ \text{proj-err} := \frac{\|A - \pi_B(A)\|_2^2}{\|A - A_k\|_F^2}, \quad k = 10 \]

- Random Projections
- Hashing
- Sampling
- FrequentDirections
- Naive
- Brute Force

[Clarkson+Woodruff STOC13]
[Drineas, Kannan, Mahoney SIAM JoC06]
Matrix Sketching in Streams:
- Row sampling methods
  - CUR
  - L2 norm based sampling
- Random projection methods
  - Johnson Lindenstrauss Transform (JLT)
  - Different ways to construct a JLT matrix
- Iterative sketching methods
  - Misra-Gries algorithm for frequent items
  - Frequent Directions method (state of the art)