Andrew McGregor
Lecture 9
Balancing Hit Rate and Query Time

Given two sets $x, y$, $MH$ is a random hash function such that

$$\Pr[MH(x) = MH(y)] = J(x, y) = \frac{|x \cap y|}{|x \cup y|}.$$ 

Use multiple such hash functions to reduce false negative probability (a high hit rate) and false positive probability (a small query time.)
Given two sets $x, y$, $MH$ is a random hash function such that

$$\Pr[\text{MH}(x) = \text{MH}(y)] = J(x, y) = \frac{|x \cap y|}{|x \cup y|}.$$ 

Use multiple such hash functions to reduce false negative probability (a high hit rate) and false positive probability (a small query time.)

Create $t$ hash tables. Each is indexed into not with a single MinHash value, but with $r$ values, appended together. A length $r$ signature.
Consider searching for matches in $t$ hash tables, using MinHash signatures of length $r$. For $x$ and $y$ with Jaccard similarity $J(x, y) = s$: 

- Probability that a single hash matches. 
- Probability that $x$ and $y$ having matching signatures in repetition $i$. 
- Probability that $x$ and $y$ don't match in repetition $i$: $1 - s^r$. 
- Probability that $x$ and $y$ don't match in all repetitions: $(1 - s^r)^t$. 
- Probability that $x$ and $y$ match in at least one repetition: Hit Probability: $1 - (1 - s^r)^t$. 


Consider searching for matches in $t$ hash tables, using MinHash signatures of length $r$. For $x$ and $y$ with Jaccard similarity $J(x, y) = s$:

- Probability that a single hash matches.
  \[ \Pr [MH_{i,j}(x) = MH_{i,j}(y)] = J(x, y) = s. \]
Consider searching for matches in $t$ hash tables, using MinHash signatures of length $r$. For $x$ and $y$ with Jaccard similarity $J(x, y) = s$:

- **Probability that a single hash matches.**
  $$\Pr [MH_{i,j}(x) = MH_{i,j}(y)] = J(x, y) = s.$$

- **Probability that $x$ and $y$ having matching signatures in repetition $i$.**
  $$\Pr [MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)].$$
Consider searching for matches in $t$ hash tables, using MinHash signatures of length $r$. For $x$ and $y$ with Jaccard similarity $J(x, y) = s$:

- Probability that a single hash matches.
  $\Pr[MH_{i,j}(x) = MH_{i,j}(y)] = J(x, y) = s$.

- Probability that $x$ and $y$ having matching signatures in repetition $i$.
  $\Pr[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)] = s^r$.
Consider searching for matches in $t$ hash tables, using MinHash signatures of length $r$. For $x$ and $y$ with Jaccard similarity $J(x, y) = s$:

- Probability that a single hash matches.
  $$\Pr [MH_{i,j}(x) = MH_{i,j}(y)] = J(x, y) = s.$$ 

- Probability that $x$ and $y$ having matching signatures in repetition $i$.
  $$\Pr [MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)] = s^r.$$ 

- Probability that $x$ and $y$ don’t match in repetition $i$:
  $$\Pr [MH_{i,1}(x) \neq MH_{i,1}(y), \ldots, MH_{i,r}(x) \neq MH_{i,r}(y)] = 1 - s^r.$$
Consider searching for matches in $t$ hash tables, using MinHash signatures of length $r$. For $x$ and $y$ with Jaccard similarity $J(x, y) = s$: 

- Probability that a single hash matches. 
  $$\Pr [MH_{i,j}(x) = MH_{i,j}(y)] = J(x, y) = s.$$ 

- Probability that $x$ and $y$ having matching signatures in repetition $i$. 
  $$\Pr [MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)] = s^r.$$ 

- Probability that $x$ and $y$ don’t match in repetition $i$: $1 - s^r$. 

BALANCING HIT RATE AND QUERY TIME
Consider searching for matches in $t$ hash tables, using MinHash signatures of length $r$. For $x$ and $y$ with Jaccard similarity $J(x, y) = s$:

- Probability that a single hash matches.
  $$\Pr [MH_{i,j}(x) = MH_{i,j}(y)] = J(x, y) = s.$$  

- Probability that $x$ and $y$ having matching signatures in repetition $i$.
  $$\Pr [MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)] = s^r.$$  

- Probability that $x$ and $y$ don’t match in repetition $i$: $1 - s^r$.  

- Probability that $x$ and $y$ don’t match in all repetitions:
Consider searching for matches in $t$ hash tables, using MinHash signatures of length $r$. For $x$ and $y$ with Jaccard similarity $J(x, y) = s$:

- Probability that a single hash matches.
  $\Pr[\text{MH}_{i,j}(x) = \text{MH}_{i,j}(y)] = J(x, y) = s$.

- Probability that $x$ and $y$ having matching signatures in repetition $i$.
  $\Pr[\text{MH}_{i,1}(x), \ldots, \text{MH}_{i,r}(x) = \text{MH}_{i,1}(y), \ldots, \text{MH}_{i,r}(y)] = s^r$.

- Probability that $x$ and $y$ don’t match in repetition $i$: $1 - s^r$.

- Probability that $x$ and $y$ don’t match in all repetitions: $(1 - s^r)^t$. 

BALANCING HIT RATE AND QUERY TIME
Consider searching for matches in $t$ hash tables, using MinHash signatures of length $r$. For $x$ and $y$ with Jaccard similarity $J(x, y) = s$:

- Probability that a single hash matches.
  \[
  \Pr[MH_{i,j}(x) = MH_{i,j}(y)] = J(x, y) = s.
  \]

- Probability that $x$ and $y$ having matching signatures in repetition $i$.
  \[
  \Pr[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)] = s^r.
  \]

- Probability that $x$ and $y$ don’t match in repetition $i$: $1 - s^r$.

- Probability that $x$ and $y$ don’t match in all repetitions: $(1 - s^r)^t$.

- Probability that $x$ and $y$ match in at least one repetition:
Consider searching for matches in $t$ hash tables, using MinHash signatures of length $r$. For $x$ and $y$ with Jaccard similarity $J(x, y) = s$:

- Probability that a single hash matches.
  \[ \text{Pr} [MH_{i,j}(x) = MH_{i,j}(y)] = J(x, y) = s. \]

- Probability that $x$ and $y$ having matching signatures in repetition $i$.
  \[ \text{Pr} [MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)] = s^r. \]

- Probability that $x$ and $y$ don’t match in repetition $i$: $1 - s^r$.

- Probability that $x$ and $y$ don’t match in all repetitions: $(1 - s^r)^t$.

- Probability that $x$ and $y$ match in at least one repetition:
  \[ \text{Hit Probability: } 1 - (1 - s^r)^t. \]
Using $t$ repetitions each with a signature of $r$ MinHash values, the probability that $x$ and $y$ with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$. 
Using $t$ repetitions each with a signature of $r$ MinHash values, the probability that $x$ and $y$ with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$.
Using $t$ repetitions each with a signature of $r$ MinHash values, the probability that $x$ and $y$ with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$. 

![Graph showing the relationship between Jaccard Similarity and Hit Probability with $r = 10, t = 10$.]
Using $t$ repetitions each with a signature of $r$ MinHash values, the probability that $x$ and $y$ with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$. 

![Graph showing the hit probability for different Jaccard similarities with $r = 5$ and $t = 30$.]
Using $t$ repetitions each with a signature of $r$ MinHash values, the probability that $x$ and $y$ with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$.

$r$ and $t$ are tuned depending on application.
**S-CURVE EXAMPLE**

**For example:** Consider a database with 10,000,000 audio clips. You are given a clip $x$ and want to find any $y$ in the database with $J(x, y) \geq .9$. 

- There are 10 true matches in the database with $J(x, y) \geq .9$.
- There are 10,000 near matches with $J(x, y) \in [.7, .9]$.
- With signature length $r = 25$ and repetitions $t = 50$, hit probability for $J(x, y) = s$ is $1 - (1 - s)^{25/50}$.
- Hit probability for $J(x, y) \geq .9$ is $\geq 1 - (1 - .9)^{20/40} \approx .98$.
- Hit probability for $J(x, y) \in [.7, .9]$ is $\leq 1 - (1 - .7)^{20/40} \approx .98$.
- Hit probability for $J(x, y) \leq .7$ is $\leq 1 - (1 - .7)^{20/40} \approx .007$.

Expected Number of Items Scanned: (proportional to query time) $\leq 10 + .98 \times 10,000 + .007 \times 9,989,990\approx 80,000 \ll 10,000,000$. 


**For example:** Consider a database with 10,000,000 audio clips. You are given a clip $x$ and want to find any $y$ in the database with $J(x, y) \geq .9$.

- There are 10 **true matches** in the database with $J(x, y) \geq .9$.
- There are 10,000 **near matches** with $J(x, y) \in [.7, .9]$.
For example: Consider a database with 10,000,000 audio clips. You are given a clip $x$ and want to find any $y$ in the database with $J(x, y) \geq .9$.

• There are 10 true matches in the database with $J(x, y) \geq .9$.
• There are 10,000 near matches with $J(x, y) \in [.7, .9]$.

With signature length $r = 25$ and repetitions $t = 50$, hit probability for $J(x, y) = s$ is $1 - (1 - s^{25})^{50}$. 
For example: Consider a database with 10,000,000 audio clips. You are given a clip $x$ and want to find any $y$ in the database with $J(x, y) \geq .9$.

- There are 10 true matches in the database with $J(x, y) \geq .9$.
- There are 10,000 near matches with $J(x, y) \in [.7, .9]$.

With signature length $r = 25$ and repetitions $t = 50$, hit probability for $J(x, y) = s$ is $1 - (1 - s^{25})^{50}$.

- Hit probability for $J(x, y) \geq .9$ is $\geq 1 - (1 - .9^{20})^{40} \approx .98$
- Hit probability for $J(x, y) \in [.7, .9]$ is $\leq 1 - (1 - .9^{20})^{40} \approx .98$
- Hit probability for $J(x, y) \leq .7$ is $\leq 1 - (1 - .7^{20})^{40} \approx .007$
For example: Consider a database with 10,000,000 audio clips. You are given a clip \( x \) and want to find any \( y \) in the database with \( J(x, y) \geq .9 \).

- There are 10 true matches in the database with \( J(x, y) \geq .9 \).
- There are 10,000 near matches with \( J(x, y) \in [.7, .9] \).

With signature length \( r = 25 \) and repetitions \( t = 50 \), hit probability for \( J(x, y) = s \) is \( 1 - (1 - s^{25})^{50} \).

- Hit probability for \( J(x, y) \geq .9 \) is \( \geq 1 - (1 - .9^{20})^{40} \approx .98 \)
- Hit probability for \( J(x, y) \in [.7, .9] \) is \( \leq 1 - (1 - .9^{20})^{40} \approx .98 \)
- Hit probability for \( J(x, y) \leq .7 \) is \( \leq 1 - (1 - .7^{20})^{40} \approx .007 \)

**Expected Number of Items Scanned:** (proportional to query time)

\[ \leq 10 + .98 \times 10,000 + .007 \times 9,989,990 \approx 80,000 \]
**For example:** Consider a database with 10,000,000 audio clips. You are given a clip $x$ and want to find any $y$ in the database with $J(x, y) \geq .9$.

- There are 10 **true matches** in the database with $J(x, y) \geq .9$.
- There are 10,000 **near matches** with $J(x, y) \in [.7, .9]$.

With signature length $r = 25$ and repetitions $t = 50$, hit probability for $J(x, y) = s$ is $1 - (1 - s^{25})^{50}$.

- Hit probability for $J(x, y) \geq .9$ is $\geq 1 - (1 - .9^{20})^{40} \approx .98$
- Hit probability for $J(x, y) \in [.7, .9]$ is $\leq 1 - (1 - .9^{20})^{40} \approx .98$
- Hit probability for $J(x, y) \leq .7$ is $\leq 1 - (1 - .7^{20})^{40} \approx .007$

**Expected Number of Items Scanned:** (proportional to query time)

$$\leq 10 + .98 \times 10,000 + .007 \times 9,989,990 \approx 80,000 \ll 10,000,000.$$
Repetition and s-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

Cosine Similarity:

\[
\cos(\theta(x, y)) = \frac{\langle x, y \rangle}{\|x\|_2 \cdot \|y\|_2}.
\]

- \(\cos(\theta(x, y)) = 1\) when \(\theta(x, y) = 0\)
- \(\cos(\theta(x, y)) = 0\) when \(\theta(x, y) = \pi/2\)
- \(\cos(\theta(x, y)) = -1\) when \(\theta(x, y) = \pi\)
Repetition and s-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.
GENERALIZING LOCALITY SENSITIVE HASHING

Repetition and s-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.
Repetition and s-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.
Repetition and s-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.

Cosine Similarity: \( \cos(\theta(x, y)) \)
Repetition and s-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.

**Cosine Similarity:** $\cos(\theta(x, y))$

- $\cos(\theta(x, y)) = 1$ when $\theta(x, y) = 0$ and $\cos(\theta(x, y)) = 0$ when $\theta(x, y) = \pi/2$, and $\cos(\theta(x, y)) = -1$ when $\theta(x, y) = \pi$
Repetition and s-curve tuning can be used for fast similarity search with any similarity metric, given a locality sensitive hash function for that metric.

- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.

**Cosine Similarity:** $\cos(\theta(x, y)) = \frac{\langle x, y \rangle}{\|x\|_2 \cdot \|y\|_2}$.

- $\cos(\theta(x, y)) = 1$ when $\theta(x, y) = 0$ and $\cos(\theta(x, y)) = 0$ when $\theta(x, y) = \pi/2$, and $\cos(\theta(x, y)) = -1$ when $\theta(x, y) = \pi$
SimHash Algorithm: LSH for cosine similarity.
SimHash Algorithm: LSH for cosine similarity.

\[ \text{SimHash}(x) = \text{sign}(\langle x, t \rangle) \] for a random vector \( t \).
SimHash Algorithm: LSH for cosine similarity.

SimHash(x) = \text{sign}(\langle x, t \rangle) \text{ for a random vector } t.
SimHash Algorithm: LSH for cosine similarity.
SimHash Algorithm: LSH for cosine similarity.

$SimHash(x) = \text{sign}(\langle x, t \rangle)$ for a random vector $t$. 
SimHash Algorithm: LSH for cosine similarity.

SimHash \( (x) = \text{sign}(\langle x, t \rangle) \) for a random vector \( t \).

What is \( \Pr [\text{SimHash}(x) = \text{SimHash}(y)] \)?
What is \( \Pr[SimHash(x) = SimHash(y)] \)?

\[
\Pr[SimHash(x) \neq SimHash(y)] = \theta(x, y) \pi
\]

\[
\Pr[SimHash(x) = SimHash(y)] = 1 - \theta(x, y) \pi \approx \cos(\theta(x, y)) + \frac{1}{2}
\]

if \( \theta(x, y) \approx \frac{\pi}{2} \).
What is \( \Pr [\text{SimHash}(x) = \text{SimHash}(y)] \)?

\( \text{SimHash}(x) \neq \text{SimHash}(y) \) when the plane separates \( x \) from \( y \).
What is $\Pr [\text{SimHash}(x) = \text{SimHash}(y)]$?

$\text{SimHash}(x) \neq \text{SimHash}(y)$ when the plane separates $x$ from $y$. 

\[
\Pr [\text{SimHash}(x) = \text{SimHash}(y)] = \frac{1}{\pi} \approx \cos(\frac{\theta(x, y)}{2}) + 1 \text{ if } \theta(x, y) \approx \frac{\pi}{2}.
\]
What is $\Pr [\text{SimHash}(x) = \text{SimHash}(y)]$?

$\text{SimHash}(x) \neq \text{SimHash}(y)$ when the plane separates $x$ from $y$.

- $\Pr [\text{SimHash}(x) \neq \text{SimHash}(y)] = \frac{\theta(x,y)}{\pi}$
What is $\Pr [\text{SimHash}(x) = \text{SimHash}(y)]$?

$\text{SimHash}(x) \neq \text{SimHash}(y)$ when the plane separates $x$ from $y$.

- $\Pr [\text{SimHash}(x) \neq \text{SimHash}(y)] = \frac{\theta(x,y)}{\pi}$
- $\Pr [\text{SimHash}(x) = \text{SimHash}(y)] = 1 - \frac{\theta(x,y)}{\pi} \approx \frac{\cos(\theta(x,y)) + 1}{2}$ if $\theta \approx \frac{\pi}{2}$.
Questions on MinHash and Locality Sensitive Hashing?
**k-Frequent Items (Heavy-Hitters) Problem**: Consider a stream of $n$ items $x_1, \ldots, x_n$ (with possible duplicates). Return any item that appears at least $\frac{n}{k}$ times.
**k-Frequent Items (Heavy-Hitters) Problem**: Consider a stream of $n$ items $x_1, \ldots, x_n$ (with possible duplicates). Return any item that appears at least $\frac{n}{k}$ times.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$x_8$</th>
<th>$x_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>
**THE FREQUENT ITEMS PROBLEMS**

**k-Frequent Items (Heavy-Hitters) Problem**: Consider a stream of \( n \) items \( x_1, \ldots, x_n \) (with possible duplicates). Return any item that appears at least \( \frac{n}{k} \) times.

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( x_5 )</th>
<th>( x_6 )</th>
<th>( x_7 )</th>
<th>( x_8 )</th>
<th>( x_9 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>
\textbf{k-Frequent Items (Heavy-Hitters) Problem}: Consider a stream of \( n \) items \( x_1, \ldots, x_n \) (with possible duplicates). Return any item that appears at least \( \frac{n}{k} \) times.

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( x_5 )</th>
<th>( x_6 )</th>
<th>( x_7 )</th>
<th>( x_8 )</th>
<th>( x_9 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

- What is the maximum number of items that must be returned?
  - a) \( n \)
  - b) \( k \)
  - c) \( n/k \)
  - d) \( \log n \)
**k-Frequent Items (Heavy-Hitters) Problem:** Consider a stream of $n$ items $x_1, \ldots, x_n$ (with possible duplicates). Return any item at appears at least $\frac{n}{k}$ times.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$x_8$</th>
<th>$x_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

- What is the maximum number of items that must be returned?
  a) $n$   b) $k$   c) $n/k$   d) $\log n$
**k-Frequent Items (Heavy-Hitters) Problem:** Consider a stream of $n$ items $x_1, \ldots, x_n$ (with possible duplicates). Return any item that appears at least $\frac{n}{k}$ times.

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$x_8$</th>
<th>$x_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>12</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

- What is the maximum number of items that must be returned?
  a) $n$  
  b) $k$  
  c) $n/k$  
  d) $\log n$

- Trivial with $O(n)$ space: Store the count for each item and return the one that appears $\geq n/k$ times.

- Can we do it with less space? I.e., without storing all $n$ items?
THE FREQUENT ITEMS PROBLEM

Applications of Frequent Items:

• Finding top/viral items (i.e., products on Amazon, videos watched on Youtube, Google searches, etc.)
• Finding very frequent IP addresses sending requests (to detect DoS attacks/network anomalies).
• ‘Iceberg queries’ for all items in a database with frequency above some threshold.
Generally want very fast detection, without having to scan through database/logs. That is we want to maintain a running list of frequent items that appear in a stream.
Applications of Frequent Items:

- Finding top/viral items (i.e., products on Amazon, videos watched on Youtube, Google searches, etc.)
Applications of Frequent Items:

- Finding top/viral items (i.e., products on Amazon, videos watched on Youtube, Google searches, etc.)
- Finding very frequent IP addresses sending requests (to detect DoS attacks/network anomalies).
Applications of Frequent Items:

- Finding top/viral items (i.e., products on Amazon, videos watched on Youtube, Google searches, etc.)
- Finding very frequent IP addresses sending requests (to detect DoS attacks/network anomalies).
- ‘Iceberg queries’ for all items in a database with frequency above some threshold.
Applications of Frequent Items:

- Finding top/viral items (i.e., products on Amazon, videos watched on Youtube, Google searches, etc.)
- Finding very frequent IP addresses sending requests (to detect DoS attacks/network anomalies).
- ‘Iceberg queries’ for all items in a database with frequency above some threshold.

Generally want very fast detection, without having to scan through database/logs. That is we want to maintain a running list of frequent items that appear in a stream.
FREQUENT ITEMSET MINING

**Association rule learning**: A very common task in data mining is to identify common associations between different events.
**Association rule learning:** A very common task in data mining is to identify common associations between different events.

**Frequent itemset mining:**

- Identified via frequent itemset counting. Find all sets of \( k \) items that appear many times in the same basket.
- Frequency of an itemset is known as its support.
- A single basket includes many different itemsets, and with many different baskets an efficient approach is critical. E.g., baskets are Twitter users and itemsets are subsets of who they follow.
**Association rule learning:** A very common task in data mining is to identify common associations between different events.
**Association rule learning:** A very common task in data mining is to identify common associations between different events.

- Identified via frequent itemset counting. Find all sets of $k$ items that appear many times in the same basket.
**Association rule learning:** A very common task in data mining is to identify common associations between different events.

- Identified via *frequent itemset* counting. Find all sets of $k$ items that appear many times in the same basket.
- Frequency of an itemset is known as its support.
Association rule learning: A very common task in data mining is to identify common associations between different events.

- Identified via frequent itemset counting. Find all sets of $k$ items that appear many times in the same basket.
- Frequency of an itemset is known as its support.
- A single basket includes many different itemsets, and with many different baskets an efficient approach is critical. E.g., baskets are Twitter users and itemsets are subsets of who they follow.
**Issue:** No algorithm using $o(n)$ space can output just the items with frequency $\geq n/k$. Hard to tell between an item with frequency $n/k$ (should be output) and $n/k - 1$ (should not be output).
**Issue:** No algorithm using $o(n)$ space can output just the items with frequency $\geq n/k$. Hard to tell between an item with frequency $n/k$ (should be output) and $n/k - 1$ (should not be output).

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_{n-k+1}$</th>
<th>$x_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>12</td>
<td>9</td>
<td>27</td>
<td>4</td>
<td>101</td>
<td>3</td>
<td>...</td>
</tr>
</tbody>
</table>

$n/k - 1$ occurrences
**Issue:** No algorithm using $o(n)$ space can output just the items with frequency $\geq n/k$. Hard to tell between an item with frequency $n/k$ (should be output) and $n/k - 1$ (should not be output).

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$\ldots$</th>
<th>$x_{n-n/k+1}$</th>
<th>$\ldots$</th>
<th>$x_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>12</td>
<td>9</td>
<td>27</td>
<td>4</td>
<td>101</td>
<td>$\ldots$</td>
<td>3</td>
<td>$\ldots$</td>
<td>3</td>
</tr>
</tbody>
</table>

$(\epsilon, k)$-Frequent Items Problem: Consider a stream of $n$ items $x_1, \ldots, x_n$. Return a set $F$ of items, including all items that appear at least $\frac{n}{k}$ times and only items that appear at least $(1 - \epsilon) \cdot \frac{n}{k}$ times.
**Issue:** No algorithm using $o(n)$ space can output just the items with frequency $\geq n/k$. Hard to tell between an item with frequency $n/k$ (should be output) and $n/k - 1$ (should not be output).

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>...</th>
<th>$x_{n-k+1}$</th>
<th>...</th>
<th>$x_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>12</td>
<td>9</td>
<td>27</td>
<td>4</td>
<td>101</td>
<td>...</td>
<td>3</td>
<td>...</td>
<td>3</td>
</tr>
</tbody>
</table>

$n/k - 1$ occurrences

**$(\epsilon, k)$-Frequent Items Problem:** Consider a stream of $n$ items $x_1, \ldots, x_n$. Return a set $F$ of items, including all items that appear at least $\frac{n}{k}$ times and only items that appear at least $(1 - \epsilon) \cdot \frac{n}{k}$ times.

- An example of relaxing to a ‘promise problem’: for items with frequencies in $[(1 - \epsilon) \cdot \frac{n}{k}, \frac{n}{k}]$ no output guarantee.
Today: Count-min sketch – a random hashing based method closely related to bloom filters.
**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.

We use $A[h(x)]$ to estimate $f(x)$, the frequency of $x$ in the stream. I.e., $|\{x_i : x_i = x\}|$.

Random hash function $h$

$m$ length array $A$:

$$
\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
$$
Today: Count-min sketch – a random hashing based method closely related to bloom filters.

Let $A[h(x)]$ estimate $f(x)$, the frequency of $x$ in the stream. I.e., $|\{x_i : x_i = x\}|$.
**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.

Consider a stream of elements \( X_1, X_2, X_3, \ldots, X_n \). We use a random hash function \( h(x) \) to estimate the frequency of each element \( x \) in the stream. The frequency of \( x \), denoted as \( f(x) \), is estimated as follows:

\[
|\{x_i : x_i = x\}|
\]

where \( A \) is an \( m \)-length array with hash values.

![Diagram](image-url)
**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.

- **Random hash function** \( h \)
- **m length array** \( A \)

\[
\begin{align*}
A &\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\
X_1 & \rightarrow & h(X_1) & \rightarrow & \text{array index} & \rightarrow & \text{value} & \rightarrow & \text{sum} & \rightarrow & \text{estimated frequency}
\end{align*}
\]
**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.

Random hash function $h$ maps elements $x_1, x_2, x_3, x_4, \ldots, x_n$ to indices in an $m$-length array $A$.
Today: Count-min sketch – a random hashing based method closely related to bloom filters.

![Diagram](image-url)

random hash function $h$

m length array $A$
**Today:** Count-min sketch – a random hashing based method closely related to bloom filters.

Will use $A[h(x)]$ to estimate $f(x)$, the frequency of $x$ in the stream. I.e., $|\{x_i : x_i = x\}|$. 
Use $A[h(x)]$ to estimate $f(x)$.

**Claim 1:** We always have $A[h(x)] \geq f(x)$. Why?

$f(x)$: frequency of $x$ in the stream (i.e., number of items equal to $x$). $h$: random hash function. $m$: size of Count-min sketch array.
Use $A[h(x)]$ to estimate $f(x)$.

**Claim 1:** We always have $A[h(x)] \geq f(x)$. Why?

- $A[h(x)]$ counts the number of occurrences of any $y$ with $h(y) = h(x)$, including $x$ itself.

---

$f(x)$: frequency of $x$ in the stream (i.e., number of items equal to $x$). $h$: random hash function. $m$: size of Count-min sketch array.
Use $A[h(x)]$ to estimate $f(x)$.

**Claim 1:** We always have $A[h(x)] \geq f(x)$. Why?

- $A[h(x)]$ counts the number of occurrences of any $y$ with $h(y) = h(x)$, including $x$ itself.
- $A[h(x)] = f(x) + \sum_{y \neq x: h(y) = h(x)} f(y)$.

---

$f(x)$: frequency of $x$ in the stream (i.e., number of items equal to $x$). $h$: random hash function. $m$: size of Count-min sketch array.
COUNT-MIN SKETCH ACCURACY

\[ A[h(x)] = f(x) + \sum_{y \neq x : h(y) = h(x)} f(y) \]

error in frequency estimate

**f(x):** frequency of \( x \) in the stream (i.e., number of items equal to \( x \)). **h:** random hash function. **m:** size of Count-min sketch array.
\[ A[h(x)] = f(x) + \sum_{y \neq x: h(y) = h(x)} f(y) . \]

**Expected Error:**
\[
\mathbb{E} \left[ \sum_{y \neq x: h(y) = h(x)} f(y) \right] = \text{error in frequency estimate}
\]

\[ f(x): \text{ frequency of } x \text{ in the stream (i.e., number of items equal to } x). \]
\[ h: \text{ random hash function. } m: \text{ size of Count-min sketch array.} \]
**Count-Min Sketch Accuracy**

\[ A[h(x)] = f(x) + \sum_{y \neq x: h(y) = h(x)} f(y) \]

**Expected Error:**

\[
\mathbb{E} \left[ \sum_{y \neq x: h(y) = h(x)} f(y) \right] = \sum_{y \neq x} \Pr(h(y) = h(x)) \cdot f(y)
\]

- **f(x):** frequency of \( x \) in the stream (i.e., number of items equal to \( x \)).
- **h:** random hash function.
- **m:** size of Count-min sketch array.
COUNT-MIN SKETCH ACCURACY

\[ A[h(x)] = f(x) + \sum_{y \neq x: h(y) = h(x)} f(y) \]

**Expected Error:**

\[
\mathbb{E} \left[ \sum_{y \neq x: h(y) = h(x)} f(y) \right] = \sum_{y \neq x} \Pr(h(y) = h(x)) \cdot f(y)
\]

\[
= \sum_{y \neq x} \frac{1}{m} \cdot f(y)
\]

\( f(x) \): frequency of \( x \) in the stream (i.e., number of items equal to \( x \)). \( h \): random hash function. \( m \): size of Count-min sketch array.
\[ A[h(x)] = f(x) + \sum_{y \neq x: h(y) = h(x)} f(y) \]

Expected Error:

\[
\mathbb{E} \left[ \sum_{y \neq x: h(y) = h(x)} f(y) \right] = \sum_{y \neq x} \Pr(h(y) = h(x)) \cdot f(y)
\]

\[
= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m}
\]

\[ f(x): \text{ frequency of } x \text{ in the stream (i.e., number of items equal to } x). \]
\[ h: \text{ random hash function. } m: \text{ size of Count-min sketch array.} \]
A[h(x)] = f(x) + \sum_{y \neq x: h(y) = h(x)} f(y).

**Expected Error:**

\[
\mathbb{E} \left[ \sum_{y \neq x: h(y) = h(x)} f(y) \right] = \sum_{y \neq x} \Pr(h(y) = h(x)) \cdot f(y)
\]

\[
= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m}
\]

What is a bound on probability that the error is $\geq \frac{2n}{m}$?

---

$f(x)$: frequency of $x$ in the stream (i.e., number of items equal to $x$). $h$: random hash function. $m$: size of Count-min sketch array.
$A[h(x)] = f(x) + \sum_{y \neq x: h(y) = h(x)} f(y)$. 

Expected Error:

$$E \left[ \sum_{y \neq x: h(y) = h(x)} f(y) \right] = \sum_{y \neq x} \Pr(h(y) = h(x)) \cdot f(y)$$

$$= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m}$$

What is a bound on probability that the error is $\geq \frac{2n}{m}$?

Markov’s inequality: $\Pr \left[ \sum_{y \neq x: h(y) = h(x)} f(y) \geq \frac{2n}{m} \right] \leq \frac{1}{2}$.
COUNT-MIN SKETCH ACCURACY

\[ A[h(x)] = f(x) + \sum_{y \neq x: h(y) = h(x)} f(y) \]

**Expected Error:**

\[
\mathbb{E} \left[ \sum_{y \neq x: h(y) = h(x)} f(y) \right] = \sum_{y \neq x} \Pr(h(y) = h(x)) \cdot f(y)
\]

\[
= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m}
\]

What is a bound on probability that the error is \( \geq \frac{2n}{m} \)?

**Markov's inequality:**

\[
\Pr \left[ \sum_{y \neq x: h(y) = h(x)} f(y) \geq \frac{2n}{m} \right] \leq \frac{1}{2}.
\]

What property of \( h \) is required to show this bound?  

a) fully random  \quad b) pairwise independent  \quad c) 2-universal  \quad d) locality sensitive

\( f(x) \): frequency of \( x \) in the stream (i.e., number of items equal to \( x \)).  \( h \): random hash function.  \( m \): size of Count-min sketch array.
COUNT-MIN SKETCH ACCURACY

\[ A[h(x)] = f(x) + \sum_{y \neq x: h(y) = h(x)} f(y) \quad . \]

**Expected Error:**

\[
\mathbb{E}\left[\sum_{y \neq x: h(y) = h(x)} f(y)\right] = \sum_{y \neq x} \Pr(h(y) = h(x)) \cdot f(y)
\]

\[= \sum_{y \neq x} \frac{1}{m} \cdot f(y) = \frac{1}{m} \cdot (n - f(x)) \leq \frac{n}{m} \]

What is a bound on probability that the error is \( \geq \frac{2n}{m} \)?

**Markov’s inequality:** \( \Pr\left[\sum_{y \neq x: h(y) = h(x)} f(y) \geq \frac{2n}{m}\right] \leq \frac{1}{2} \).

What property of \( h \) is required to show this bound? a) fully random  
  b) pairwise independent  
  c) 2-universal  
  d) locality sensitive

\( f(x) \): frequency of \( x \) in the stream (i.e., number of items equal to \( x \)). \( h \): random hash function. \( m \): size of Count-min sketch array.
Claim: For any $x$, with probability at least $1/2$,

$$f(x) \leq A[h(x)] \leq f(x) + \frac{2n}{m}.$$
Claim: For any $x$, with probability at least $1/2$,

$$f(x) \leq A[h(x)] \leq f(x) + \frac{2n}{m}.$$ 

To solve the $(\epsilon, k)$-Frequent elements problem, set $m = \frac{2k}{\epsilon}$.

**f(x):** frequency of $x$ in the stream (i.e., number of items equal to $x$). **h:** random hash function. **m:** size of Count-min sketch array.
Claim: For any $x$, with probability at least $1/2$,

$$f(x) \leq A[h(x)] \leq f(x) + \frac{2n}{m}.$$ 

To solve the $(\epsilon, k)$-Frequent elements problem, set $m = \frac{2k}{\epsilon}$.

How can we improve the success probability?

$f(x)$: frequency of $x$ in the stream (i.e., number of items equal to $x$). $h$: random hash function. $m$: size of Count-min sketch array.
Claim: For any $x$, with probability at least $1/2$, 

$$f(x) \leq A[h(x)] \leq f(x) + \frac{2n}{m}.$$ 

To solve the $(\epsilon, k)$-Frequent elements problem, set $m = \frac{2k}{\epsilon}$.

How can we improve the success probability? **Repetition.**

- $f(x)$: frequency of $x$ in the stream (i.e., number of items equal to $x$).
- $h$: random hash function.
- $m$: size of Count-min sketch array.
Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$. (count-min sketch)

Why min instead of mean or median? The minimum estimate is always the most accurate since they are all overestimates of the true frequency!
Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$. (count-min sketch)

Why min instead of mean or median? The minimum estimate is always the most accurate since they are all overestimates of the true frequency!
Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$. (count-min sketch)

Why min instead of mean or median?
The minimum estimate is always the most accurate since they are all overestimates of the true frequency!
Count-Min Sketch Accuracy

Estimate \( f(x) \) with \( \tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)] \). (count-min sketch)

Why min instead of mean or median?

The minimum estimate is always the most accurate since they are all overestimates of the true frequency!
Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$. (count-min sketch)
Estimate $f(x)$ with $	ilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$. (count-min sketch)
Estimate $f(x)$ with $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$. (count-min sketch)

Why min instead of mean or median?
Estimate \( f(x) \) with \( \tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)] \). (count-min sketch)

Why min instead of mean or median? The minimum estimate is always the most accurate since they are all overestimates of the true frequency!
Estimate $f(x)$ by $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$
Estimate $f(x)$ by $	ilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$

- For every $x$ and $i \in [t]$, we know that for $m = \frac{2k}{\epsilon}$, with probability $\geq 1/2$:
  \[ f(x) \leq A_i[h_i(x)] \leq f(x) + \frac{\epsilon n}{k}. \]
Estimate $f(x)$ by $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$

- For every $x$ and $i \in [t]$, we know that for $m = \frac{2k}{\epsilon}$, with probability $\geq 1/2$:
  $$f(x) \leq A_i[h_i(x)] \leq f(x) + \frac{\epsilon n}{k}.$$ 

- What is $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}]$?
Estimate $f(x)$ by $	ilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$

- For every $x$ and $i \in [t]$, we know that for $m = \frac{2k}{\epsilon}$, with probability $\geq 1/2$:
  $$f(x) \leq A_i[h_i(x)] \leq f(x) + \frac{\epsilon n}{k}.$$ 

- What is $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}] = 1 - 1/2^t$. 

**Count-Min Sketch Analysis**

$$\begin{array}{c|c|c|c|c|c|c} 
\text{Layer} & x_1 & x_2 & x_3 & x_4 & \ldots & x_n \\
\hline 
A_1 & 2 & 5 & 1 & 0 & 6 & 12 & 104 & 1 & 3 & 4 \\
A_2 & 1 & 6 & 1 & 10 & 78 & 80 & 4 & 11 & 3 & 5 \\
\vdots & & & & & & & & & & \\
A_t & 90 & 1 & 52 & 6 & 3 & 12 & 33 & 9 & 3 & 2 \\
\end{array}$$

t random hash functions $h_1, h_2, \ldots, h_t$ 

t length m arrays
Estimate $f(x)$ by $\tilde{f}(x) = \min_{i \in [t]} A_i[h_i(x)]$

- For every $x$ and $i \in [t]$, we know that for $m = \frac{2k}{\epsilon}$, with probability $\geq 1/2$:
  $$f(x) \leq A_i[h_i(x)] \leq f(x) + \frac{\epsilon n}{k}.$$

- What is $\Pr[f(x) \leq \tilde{f}(x) \leq f(x) + \frac{\epsilon n}{k}] = 1 - 1/2^t$.

- To get a good estimate with probability $\geq 1 - \delta$, set $t = \log(1/\delta)$. 

```
<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Upshot: Count-min sketch lets us estimate the frequency of every item in a stream up to error $\frac{\epsilon n}{k}$ with probability $\geq 1 - \delta$ in $O(\log(1/\delta) \cdot k/\epsilon)$ space.
**Upshot:** Count-min sketch lets us estimate the frequency of every item in a stream up to error $\frac{\varepsilon n}{k}$ with probability $\geq 1 - \delta$ in $O(\log(1/\delta) \cdot k/\varepsilon)$ space.

- Accurate enough to solve the $(\varepsilon, k)$-Frequent elements problem – distinguish between items with frequency $\frac{n}{k}$ and those with frequency $(1 - \varepsilon)\frac{n}{k}$.
**Upshot:** Count-min sketch lets us estimate the frequency of every item in a stream up to error $\frac{\epsilon n}{k}$ with probability $\geq 1 - \delta$ in $O(\log(1/\delta) \cdot k/\epsilon)$ space.

- Accurate enough to solve the $(\epsilon, k)$-Frequent elements problem – distinguish between items with frequency $\frac{n}{k}$ and those with frequency $(1 - \epsilon)\frac{n}{k}$.
- How should we set $\delta$ if we want a good estimate for all items at once, with 99% probability?
Count-min sketch gives an accurate frequency estimate for every item in the stream. But how do we identify the frequent items without having to store/look up the estimated frequency for all elements in the stream?
Identifying frequent elements

Count-min sketch gives an accurate frequency estimate for every item in the stream. But how do we identify the frequent items without having to store/look up the estimated frequency for all elements in the stream?

**One approach:**

- When a new item comes in at step $i$, check if its estimated frequency is $\geq i/k$ and store it if so.
- At step $i$ remove any stored items whose estimated frequency drops below $i/k$.
- Store at most $O(k)$ items at once and have all items with frequency $\geq n/k$ stored at the end of the stream.
Questions on Frequent Elements?