LEMP: Fast Retrieval of Large Entries in a Matrix Product

Christina Teflioudi\textsuperscript{1}  Rainer Gemulla\textsuperscript{2}  Olga Mykytiuk

\textsuperscript{1}Max Planck Institute for Computer Science
Saarbrücken, Germany

\textsuperscript{2}Mannheim University
Mannheim, Germany

May 4, 2015
Recommender Systems

- **Problem**
  - Set of users
  - Set of items (movies, books, jokes, products, stories, ...)
  - Feedback (ratings, purchase, click-through, tags, ...)
  - Predict the preference of each user for each item

<table>
<thead>
<tr>
<th></th>
<th>Die Hard</th>
<th>Taken</th>
<th>Once</th>
<th>Amelie</th>
<th>Titanic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Adam</strong></td>
<td>5</td>
<td>?</td>
<td>1</td>
<td>2</td>
<td>?</td>
</tr>
<tr>
<td><strong>Bob</strong></td>
<td>5</td>
<td>4</td>
<td>?</td>
<td>?</td>
<td>1</td>
</tr>
<tr>
<td><strong>Charlie</strong></td>
<td>2</td>
<td>?</td>
<td>5</td>
<td>?</td>
<td>4</td>
</tr>
<tr>
<td><strong>Dennis</strong></td>
<td>?</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>?</td>
</tr>
</tbody>
</table>
Latent Factor Models

Die Hard
Taken
Once
Amelie
Titanic
Adam
Bob
Charlie
Dennis
Action
Romance
Recommender Systems

- Given
  - A query matrix $Q$

$\begin{bmatrix}
  Adam & 3.2 & -0.4 \\
  Bob & 3.1 & -0.2 \\
  Charlie & 0 & 1.8 \\
  Dennis & -0.4 & 1.9 \\
\end{bmatrix}
\quad Q^T$
Recommender Systems

- Given
  - A query matrix $Q$
  - A probe matrix $P$

\[
Q^T = \begin{pmatrix}
3.2 & -0.4 \\
3.1 & -0.2 \\
0 & 1.8 \\
-0.4 & 1.9
\end{pmatrix}
\]

\[
P = \begin{pmatrix}
1.6 & 1.3 & 0.7 & 1 & 0.4 \\
0.6 & 0.8 & 2.7 & 2.8 & 2.2
\end{pmatrix}
\]
Recommender Systems

- Given
  - A query matrix $Q$
  - A probe matrix $P$
  - A threshold $\theta > 0$

- Find good recommendations

Given:

$$Q^T P$$

$Q^T$ and $P$ are matrices such that:

$$Q = \begin{pmatrix} 3.2 & -0.4 \\ 3.1 & -0.2 \\ 0 & 1.8 \\ -0.4 & 1.9 \end{pmatrix}$$

$$P = \begin{pmatrix} 1.6 & 1.3 & 0.7 & 1 & 0.4 \\ 0.6 & 0.8 & 2.7 & 2.8 & 2.2 \end{pmatrix}$$

$$Q^T P = \begin{pmatrix} Q^T & P \end{pmatrix} = \begin{pmatrix} 3.2 & -0.4 \\ 3.1 & -0.2 \\ 0 & 1.8 \\ -0.4 & 1.9 \end{pmatrix} \begin{pmatrix} 1.6 & 1.3 & 0.7 & 1 & 0.4 \\ 0.6 & 0.8 & 2.7 & 2.8 & 2.2 \end{pmatrix}$$

Die Hard: $1.6 
Taken: $1.3 
Once: $0.7 
Amelie: $1 
Titanic: $0.4$
Recommender Systems

- Given
  - A query matrix $Q$
  - A probe matrix $P$
  - A threshold $\theta > 0$

- Find good recommendations
  - All entries in $Q^T P$ that are $\geq \theta$

<table>
<thead>
<tr>
<th></th>
<th>Die Hard</th>
<th>Taken</th>
<th>Once</th>
<th>Amelie</th>
<th>Titanic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adam</td>
<td>3.2</td>
<td>-0.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>3.1</td>
<td>-0.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Charlie</td>
<td>0</td>
<td>1.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dennis</td>
<td>-0.4</td>
<td>1.9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$$Q^T Q$$

$$Q^T P$$
Recommender Systems

- Given
  - A query matrix $Q$
  - A probe matrix $P$
  - A threshold $\theta > 0$

- Find good recommendations
  - All entries in $Q^T P$ that are $\geq \theta$
  - Each entry is an inner product $q^T p = \sum_{i=1}^{r} q_i p_i$

$$
\begin{pmatrix}
1.6 & 1.3 & 0.7 & 1 & 0.4 \\
0.6 & 0.8 & 2.7 & 2.8 & 2.2
\end{pmatrix}
\begin{pmatrix}
Adam \\
3.2 & -0.4
\end{pmatrix}
= 
\begin{pmatrix}
4.9 & 3.8 & 1.2 & 2.1 & 0.4 \\
4.8 & 3.9 & 1.6 & 2.5 & 0.8
\end{pmatrix}
$$
Problem Statement

Maximum Inner Product Search
Find pairs of vectors with large inner products

Given

- a query vector $q^T$ (from matrix $Q^T$)
- a set $P$ of probe vectors (the matrix $P$)
- a threshold $\theta > 0$

Find

- all vectors $p$ such that $q^T p \geq \theta$
Naive Solution

- Compute full matrix product $Q^T P$
- Determine which entries are $\geq \theta$
- Complexity $O(mnr)$
- Usually
  - $m, n$: order of millions
  - $10 < r < 500$
- Example
  - $m = 10$ millions
  - $n = 1$ million
  - #entries = 10 trillion
  - Avg. Inner Product Time = 100nsec
  - Runtime $> 11$ days
- Can we do better than that?
When is an inner product large?

\[ q^T p = \|q\| \|p\| \cos \angle(q, p) \geq \theta, \quad -1 \leq \cos \angle(q, p) \leq 1 \]
When is an inner product large?

\[ q^T p = \|q\| \|p\| \cos \angle(q, p) \geq \theta, \quad -1 \leq \cos \angle(q, p) \leq 1 \]

<table>
<thead>
<tr>
<th>Length ((|q| |p|))</th>
<th>short ((&lt; \theta))</th>
<th>medium ((\approx \theta))</th>
<th>long ((\gg \theta))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle ((\cos \angle(q, p)))</td>
<td>small positive</td>
<td>large positive</td>
<td></td>
</tr>
</tbody>
</table>

Suitable method

Prune

Cosine Similarity

Naive-like

Search

\(\theta' = \theta\|q\| \|p\|\)
When is an inner product large?

\[ q^T p = \|q\|\|p\| \cos \angle(q, p) \geq \theta, \quad -1 \leq \cos \angle(q, p) \leq 1 \]

<table>
<thead>
<tr>
<th>Length ((|q||p|))</th>
<th>short ((&lt; \theta))</th>
<th>medium ((\approx \theta))</th>
<th>long ((\gg \theta))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle ((\cos \angle(q, p)))</td>
<td>small positive</td>
<td>large positive</td>
<td></td>
</tr>
<tr>
<td>Suitable method</td>
<td>Prune</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
When is an inner product large?

\[ q^T p = \|q\| \|p\| \cos \angle (q, p) \geq \theta, \quad -1 \leq \cos \angle (q, p) \leq 1 \]

<table>
<thead>
<tr>
<th>Length ((|q| |p|))</th>
<th>short ((&lt; \theta))</th>
<th>medium ((\approx \theta))</th>
<th>long ((\gg \theta))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle ((\cos \angle (q, p)))</td>
<td>small positive large positive</td>
<td>small positive large positive</td>
<td></td>
</tr>
<tr>
<td>Suitable method</td>
<td>Prune</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Short** \((< \theta)\): Length is short, angle is small positive. Prune.
- **Medium** \((\approx \theta)\): Length is medium, angle is small positive. Likely large positive.
- **Long** \((\gg \theta)\): Length is long, angle is small positive. Likely large positive.
When is an inner product large?

\[ q^T p = \|q\| \|p\| \cos \angle(q, p) \geq \theta , \ -1 \leq \cos \angle(q, p) \leq 1 \]

<table>
<thead>
<tr>
<th>Length ((|q| |p|))</th>
<th>short ((&lt; \theta))</th>
<th>medium ((\approx \theta))</th>
<th>long ((\gg \theta))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle ((\cos \angle(q, p)))</td>
<td>small positive large positive</td>
<td>small positive large positive</td>
<td></td>
</tr>
<tr>
<td>Suitable method</td>
<td>Prune</td>
<td>Cosine Similarity Search (\theta' = \frac{\theta}{|q| |p|})</td>
<td></td>
</tr>
</tbody>
</table>
When is an inner product large?

\[ q^T p = \|q\| \|p\| \cos \angle(q, p) \geq \theta, -1 \leq \cos \angle(q, p) \leq 1 \]

<table>
<thead>
<tr>
<th>Length ((|q| |p|))</th>
<th>short ((&lt; \theta))</th>
<th>medium ((\approx \theta))</th>
<th>long ((\gg \theta))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle ((\cos \angle(q, p)))</td>
<td>small positive large positive</td>
<td>small positive large positive</td>
<td>small positive large positive</td>
</tr>
<tr>
<td>Suitable method</td>
<td>Prune</td>
<td>Cosine Similarity Search (\theta' = \frac{\theta}{|q| |p|})</td>
<td></td>
</tr>
</tbody>
</table>
When is an inner product large?

\[ q^T p = \|q\| \|p\| \cos \angle(q, p) \geq \theta, \quad -1 \leq \cos \angle(q, p) \leq 1 \]

<table>
<thead>
<tr>
<th>Length ((|q| |p|))</th>
<th>short ((&lt; \theta))</th>
<th>medium ((\approx \theta))</th>
<th>long ((\gg \theta))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angle ((\cos \angle(q, p)))</td>
<td>small positive large positive</td>
<td>small positive large positive</td>
<td>small positive large positive</td>
</tr>
<tr>
<td>Suitable method</td>
<td>Prune</td>
<td>Cosine Similarity Search (\theta' = \frac{\theta}{|q| |p|})</td>
<td>Naive-like retrieval</td>
</tr>
</tbody>
</table>

\(\theta\) = \text{threshold for similarity}
Main idea: bucketize by length

- Partition $\mathbf{P}$ in buckets with vectors of similar length
- Index buckets suitably

\[
\begin{array}{c}
\|q_1\| = 1 \\
q_1 \\
Q \\
\|q_2\| = 0.1 \\
q_2
\end{array}
\]

\[
\begin{array}{c}
\mathbf{Q}^T \\
\theta = 0.9
\end{array}
\]

\[
\begin{array}{c}
\mathbf{P}^T \\
\text{max length} \\
\theta_b(q)
\end{array}
\]

\[
\begin{array}{c}
\mathbf{P_1} \\
l_1 = 2 \\
\mathbf{P_2} \\
l_2 = 1 \\
\mathbf{P_3} \\
l_3 = 0.5
\end{array}
\]
Main idea: bucketize by length

- Partition $\mathbf{P}$ in buckets with vectors of similar length
- Index buckets suitably
- For each query vector and bucket
  - Determine local threshold
  - Prune bucket if possible

Local threshold on cosine similarity:

$$\theta_b(\mathbf{q}) = \frac{\theta}{\|\mathbf{q}\| l_b}$$
Main idea: bucketize by length

- Partition $\mathbf{P}$ in buckets with vectors of similar length
- Index buckets suitably
- For each query vector and bucket
  - Determine local threshold
  - Prune bucket if possible
  - Otherwise, select best retrieval method

Local threshold on cosine similarity:

$$\theta_b(\mathbf{q}) = \frac{\theta}{\|\mathbf{q}\| l_b}$$
Discussion

LEMP

- Prunes whole buckets
- Selects a suitable retrieval algorithm per query and bucket
- Can leverage existing methods
- Cache-friendly
Bucket-level retrieval

Choose among a variety of algorithms

- Fagin’s Threshold Algorithm \((r < 10)\)
- All pairs similarity search family \((1000 < r)\)
- Space-partitioning trees \((r < 10)\)

Our vectors are

- Not necessarily sparse
- Real values
- Medium dimensionality \((10 < r < 500)\)

Two new algorithms

- COORD
- INCR
INCR: Main Idea

- $\mathbf{q}, \mathbf{p}$ qualify if $||\mathbf{q}|| \cdot ||\mathbf{p}|| \left( \sum_{i=1}^{r} \bar{q}_i \bar{p}_i \right) \geq \theta$

Table:

| $||\mathbf{q}||$ | $\bar{q}_1$ | $\bar{q}_2$ | $\bar{q}_3$ | $\bar{q}_4$ | ... | $\bar{q}_r$ |
|------------------|-------------|-------------|-------------|-------------|-----|-------------|
| $||\mathbf{p}||$ | $\bar{p}_1$ | $\bar{p}_2$ | $\bar{p}_3$ | $\bar{p}_4$ | ... | $\bar{p}_r$ |
INCR: Main Idea

- $\mathbf{q}, \mathbf{p}$ qualify if $\|\mathbf{q}\| \|\mathbf{p}\| (\sum_{i=1}^{r} \bar{q}_i \bar{p}_i) \geq \theta$

- Assume you have a budget of $\phi = 3$ multiplications for the $\sum_{i=1}^{r} \bar{q}_i \bar{p}_i$

- In practice, $\phi$ is automatically tuned. The $\phi$ coordinates do not have to be consecutive.
INCR: Main Idea

- \( q, p \) qualify if \( \|q\|\|p\|(\sum_{i=1}^{r} \bar{q}_i\bar{p}_i) \geq \theta \)

- Assume you have a budget of \( \phi = 3 \) multiplications for the \( \sum_{i=1}^{r} \bar{q}_i\bar{p}_i \)

- Goal: decide after seeing \( \phi = 3 \) coordinates:

\[
\begin{array}{c|cccccc}
\|q\| & \bar{q}_1 & \bar{q}_2 & \bar{q}_3 & \bar{q}_4 & \ldots & \bar{q}_r \\
\|p\| & \bar{p}_1 & \bar{p}_2 & \bar{p}_3 & \bar{p}_4 & \ldots & \bar{p}_r \\
\end{array}
\]
INCR: Main Idea

- \( \mathbf{q}, \mathbf{p} \) qualify if \( \|\mathbf{q}\|\|\mathbf{p}\|(\sum_{i=1}^{r} \bar{q}_i \bar{p}_i) \geq \theta \)

- Assume you have a budget of \( \phi = 3 \) multiplications for the \( \sum_{i=1}^{r} \bar{q}_i \bar{p}_i \)

- Goal: decide after seeing \( \phi = 3 \) coordinates:
  \[ \|\mathbf{q}\|\|\mathbf{p}\|(\sum_{i=1}^{3} \bar{q}_i \bar{p}_i + \text{UpperBoundFor}(\sum_{i=4}^{r} \bar{q}_i \bar{p}_i)) \geq \theta \]

- In practice, \( \phi \) is automatically tuned.
- The \( \phi \) coordinates do not have to be consecutive.
INCR: Main Idea

- $\mathbf{q}, \mathbf{p}$ qualify if $\|\mathbf{q}\|\|\mathbf{p}\| (\sum_{i=1}^{r} \bar{q}_i \bar{p}_i) \geq \theta$

- Assume you have a budget of $\phi = 3$ multiplications for the $\sum_{i=1}^{r} \bar{q}_i \bar{p}_i$

- Goal: decide after seeing $\phi = 3$ coordinates:
  $\|\mathbf{q}\|\|\mathbf{p}\| (\sum_{i=1}^{3} \bar{q}_i \bar{p}_i + \text{UpperBoundFor}(\sum_{i=4}^{r} \bar{q}_i \bar{p}_i)) \geq \theta$

- In practice
  - $\phi$ is automatically tuned
  - the $\phi$ coordinates do not have to be consecutive

<table>
<thead>
<tr>
<th>$|\mathbf{q}|$</th>
<th>$\bar{q}_1$</th>
<th>$\bar{q}_2$</th>
<th>$\bar{q}_3$</th>
<th>$\bar{q}_4$</th>
<th>...</th>
<th>$\bar{q}_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$|\mathbf{p}|$</td>
<td>$\bar{p}_1$</td>
<td>$\bar{p}_2$</td>
<td>$\bar{p}_3$</td>
<td>$\bar{p}_4$</td>
<td>...</td>
<td>$\bar{p}_r$</td>
</tr>
</tbody>
</table>
INCR

\[ \bar{q} \]

Pick coordinates (largest \( |q_i| \) first)

Scan indexes and update "\( \sum \)" and "UB" quantities

Prune vectors for which \( \|q\| \|p\| (\sum + UB) < \theta \)

No need to scan the whole lists!

Bounds exist

\[ \bar{q} \]

Probe Bucket - Sorted Lists

\[ p_1 \quad p_2 \quad p_3 \cdots \]
INCR

Pick coordinates (largest | q_i | first) 
Scan indexes and update "\( \sum \) and "UB" quantities 
Prune vectors for which \(| q_i \| \| p_i \| (\sum + UB) < \theta \)
No need to scan the whole lists!

Bounds exist

Probe Bucket - Sorted Lists

\( \bar{q} \) 

\( \bar{p}_1 \)  \( \bar{p}_2 \)  \( \bar{p}_3 \)  \( \bar{p}_r \) 

\( \text{id} \) \( \sum \)  UB
- Pick coordinates (largest $|\bar{q}_i|$ first)
- Pick coordinates (largest $|\bar{q}_i|$ first)
- Scan indexes and update “$\sum$” and “UB” quantities
INCR

- Pick coordinates (largest $|\bar{q}_i|$ first)
- Scan indexes and update “$\sum$” and “UB” quantities
- Prune vectors for which $\|q\|\|p\|(\sum + UB) < \theta$

![Diagram of Probe Bucket - Sorted Lists](image-url)
INCR

- Pick coordinates (largest $|\tilde{q}_i|$ first)
- Scan indexes and update “$\sum$” and “UB” quantities
- Prune vectors for which $\|q\| \|p\| (\sum + UB) < \theta$
- No need to scan the whole lists! Bounds exist
Discussion

INCR
- Computes partial inner products
- Uses simple index - cheap to construct
- Scans part of the index - does not necessarily scan from top
- Sequential memory access pattern, fast
How fast is it? (Top-1 movie per user)

### Dimensions
- **IE-NMF\(^T\) (seconds)**: 22.3x
- **IE-SVD\(^T\) (seconds)**: 13.8x
- **Netflix (seconds)**: 2x
- **KDD (hours)**: 2x

### Data Characteristics
<table>
<thead>
<tr>
<th># rows</th>
<th># cols</th>
<th>Sparse?</th>
<th>Length skew</th>
<th>Dimensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>132K</td>
<td>771K</td>
<td>Yes</td>
<td>Strong</td>
<td>50</td>
</tr>
<tr>
<td>132K</td>
<td>771K</td>
<td>No</td>
<td>Strong</td>
<td>50</td>
</tr>
<tr>
<td>480K</td>
<td>17K</td>
<td>No</td>
<td>Mild</td>
<td>50</td>
</tr>
<tr>
<td>1000K</td>
<td>624K</td>
<td>No</td>
<td>Mild</td>
<td>50</td>
</tr>
</tbody>
</table>
Summary

Large inner-product search
- Matrix factorization common technique in data mining
- Large entries in matrix products are usually of particular interest

The LEMP algorithm
- Bucketizes vectors by length
- Prunes buckets whenever possible
- For the remaining buckets: selects efficient retrieval algorithms
- Consistently fastest bucket method: INCR
Summary

Large inner-product search

- Matrix factorization common technique in data mining
- Large entries in matrix products are usually of particular interest

The LEMP algorithm

- Bucketizes vectors by length
- Prunes buckets whenever possible
- For the remaining buckets: selects efficient retrieval algorithms
- Consistently fastest bucket method: INCR

Thank you!

Questions?
Performance of bucket algorithms

IE-NMF\(^T\)

IE-SVD\(^T\)
Do we need to scan the whole lists?
Do we need to scan the whole lists?
COORD: bounds

- Find for each coordinate $\bar{p}_f$ of $\bar{p}$ an upper and lower bound $[L_f, U_f]$ such that $\bar{p}_f \notin [L_f, U_f] \Rightarrow \bar{q}^T \bar{p} < \theta_b(q)$
Algorithm selection

- Take a sample of queries
- Run both naive and INCR/COORD
- Estimate the value of $t_b$
- Linear classifier with $\theta_b(q)$ as feature