Mining at scale with latent factor models for matrix completion

Fabio Petroni
Marco wants to watch a movie.

DVD rental store

A (1998)
A-ge-man (1990)
A Nous Amours (1983)
... many pages later ...
Azumi (2003)
Recommender systems

Marco wants to watch a movie.

But there are so many movies! Which ones will he like?
Collaborative filtering

- problem
  - set of users
  - set of items (movies, books, songs, ...)
  - feedback
    - explicit (ratings, ...)
    - implicit (purchase, click-through, ...)
- predict the preference of each user for each item
  - assumption: similar feedback ↔ similar taste
- example (explicit feedback):

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>The Matrix</th>
<th>Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marco</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Luca</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Anna</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
Collaborative filtering

- **problem**
  - set of users
  - set of items (movies, books, songs, ...)
  - feedback
    - explicit (ratings, ...)
    - implicit (purchase, click-through, ...)

- predict the preference of each user for each item
  - assumption: similar feedback ↔ similar taste

- example (explicit feedback):

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>The Matrix</th>
<th>Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marco</td>
<td>?</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Luca</td>
<td>3</td>
<td>2</td>
<td>?</td>
</tr>
<tr>
<td>Anna</td>
<td>5</td>
<td>?</td>
<td>3</td>
</tr>
</tbody>
</table>
- matrix completion is currently considered the best approach
- advantages with respect to both scalability and accuracy
Matrix completion for collaborative filtering

- the completion is driven by a factorization

\[ R \approx P \times Q \]

- associate a latent factor vector with each user and each item
- missing entries are estimated through the dot product

\[ r_{ij} \approx p_i q_j \]
Latent factor models

(Koren et al., 2009)
Latent factor models - explicit feedback

- discover latent factors \( (d = 1) \)

<table>
<thead>
<tr>
<th></th>
<th>Avatar (2.24)</th>
<th>The Matrix (1.92)</th>
<th>Up (1.18)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marco (1.98)</td>
<td>?</td>
<td>4 (3.8)</td>
<td>2 (2.3)</td>
</tr>
<tr>
<td>Luca (1.21)</td>
<td>3 (2.7)</td>
<td>2 (2.3)</td>
<td>?</td>
</tr>
<tr>
<td>Anna (2.30)</td>
<td>5 (5.2)</td>
<td>?</td>
<td>3 (2.7)</td>
</tr>
</tbody>
</table>

- optimization criterion: \( \text{minimize squared loss} \)

\[
\minimize_{P, Q} \sum_{(i,j) \in O} (r_{ij} - p_i q_j)^2
\]
Latent factor models - explicit feedback

- discover latent factors \((d = 1)\)

<table>
<thead>
<tr>
<th></th>
<th>Avatar (2.24)</th>
<th>The Matrix (1.92)</th>
<th>Up (1.18)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marco (1.98)</td>
<td>4.4</td>
<td>4 (3.8)</td>
<td>2 (2.3)</td>
</tr>
<tr>
<td>Luca (1.21)</td>
<td>3 (2.7)</td>
<td>2 (2.3)</td>
<td>1.4</td>
</tr>
<tr>
<td>Anna (2.30)</td>
<td>5 (5.2)</td>
<td>4.4</td>
<td>3 (2.7)</td>
</tr>
</tbody>
</table>

- optimization criterion: minimize squared loss

\[
\text{minimize } \sum_{P,Q} \sum_{(i,j) \in O} (r_{ij} - p_iq_j)^2
\]
Latent factor models - explicit feedback

- discover latent factors \( (d = 1) \)

<table>
<thead>
<tr>
<th></th>
<th>Avatar (2.24)</th>
<th>The Matrix (1.92)</th>
<th>Up (1.18)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marco (1.98)</td>
<td>4.4</td>
<td>4 (3.8)</td>
<td>2 (2.3)</td>
</tr>
<tr>
<td>Luca (1.21)</td>
<td>3 (2.7)</td>
<td>2 (2.3)</td>
<td>1.4</td>
</tr>
<tr>
<td>Anna (2.30)</td>
<td>5 (5.2)</td>
<td>4.4</td>
<td>3 (2.7)</td>
</tr>
</tbody>
</table>

- optimization criterion: minimize squared loss

\[
\text{minimize} \sum_{P,Q} \sum_{(i,j) \in O} (r_{ij} - b_\mu - b_{ui} - b_{xj} - p_i q_j)^2 + \lambda (\|P\|_F + \|Q\|_F)
\]
Latent factor models - implicit feedback

- discover latent factors \((d = 1)\)

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>The Matrix</th>
<th>Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marco</td>
<td>?</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Luca</td>
<td>1</td>
<td>1</td>
<td>?</td>
</tr>
<tr>
<td>Anna</td>
<td>1</td>
<td>?</td>
<td>1</td>
</tr>
</tbody>
</table>

- the squared loss minimization criterion is not effective!
Latent factor models - implicit feedback

- discover latent factors \((d = 1)\)

<table>
<thead>
<tr>
<th></th>
<th>Avatar (1)</th>
<th>The Matrix (1)</th>
<th>Up (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marco (1)</td>
<td>1</td>
<td>1 (1)</td>
<td>1 (1)</td>
</tr>
<tr>
<td>Luca (1)</td>
<td>1 (1)</td>
<td>1 (1)</td>
<td>1</td>
</tr>
<tr>
<td>Anna (1)</td>
<td>1 (1)</td>
<td>1</td>
<td>1 (1)</td>
</tr>
</tbody>
</table>

- the squared loss minimization criterion is **not effective**!
- the system simply complete the matrix with all 1s
Bayesian personalized ranking

- problem related with ranking more than prediction
  - e.g., ranked list of items that the user might like the most

- the BPR criterion adopts a pairwise approach
  - predict whether item $x_j$ is more probable to be liked than $x_k$
  - assumption: the user prefers items for which a feedback exists

$$D_T := \{(u_i, x_j, x_k) | u_i \in U \land x_j, x_k \in I, j \neq k \land r_{ij} = 1 \land r_{ik} = ? \}$$

- user $u_i$ is assumed to prefer item $x_j$ over $x_k$

$$\text{maximize } \sum_{P, Q} \sum_{(u_i, x_j, x_k) \in D_T} \left[ \ln \sigma(p_i q_j - p_i q_k) \right]$$
Stochastic gradient descent

- parameters $\Theta = \{P, Q\}$
- find minimum $\Theta^*$ of loss function $L$, or maximum for BPR (ascent)
- pick a starting point $\Theta^0$
- iteratively update current estimations for $\Theta$

$$\Theta_{n+1} \leftarrow \Theta_n - \eta \frac{\partial L}{\partial \Theta}$$

- learning rate $\eta$
- an update for each given training point
Overview matrix completion

model

\[ P \]

user latent factor vectors

\[ Q \]

item latent factor vectors

explicit feedback

regularized squared loss

implicit feedback

Bayesian personalized ranking

train the model

solve optimization problem

optimization criteria

stochastic gradient descent (ascent)
Challenges of matrix completion

(1) **scalability**: handle large scale data

- 2B purchases on Amazon.com by 200M customers (2014)
- parallel and distributed approaches are essential!

(2) **quality**: improve prediction performance

- Netflix awarded a 10% improvement with $1M (2006)
- the performance of matrix completion can be boosted by:
  - feeding the system with more data
  - integrating contextual information in the model
Overview - state-of-the-art

<table>
<thead>
<tr>
<th>Scalability</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Centralized</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Context-agnostic</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Context-aware</strong></td>
</tr>
<tr>
<td><strong>Distributed</strong></td>
<td><strong>Context-agnostic</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Parallel</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Context-aware</strong></td>
</tr>
</tbody>
</table>

- **Regularized Squared Loss**
  - Positive and negative evidence
  - Multi-value revealed entries

- **Bayesian Personalized Ranking**
  - Only positive evidence
  - Single-value revealed entries

Authors and Year:
- Makari et al., 2014
- Ahmed et al., 2013
- Zhuang et al., 2013
- Recht et al., 2013
- Niu et al., 2011
- Ricci et al., 2011
- Shi et al., 2014
- Takács et al., 2009
- Rendle, 2012
- Koren et al., 2009
- Rendle et al., 2011
- Koren, 2008
- Karatzoglou et al., 2010
- Riedel et al., 2013
- Chen et al., 2012
- Menon et al., 2011
- Rendle et al., 2009

13 of 58
Overview - contributions

Regularized Squared Loss
- positive and negative evidence
- multi-value revealed entries

Bayesian Personalized Ranking
- only positive evidence
- single-value revealed entries

- Petroni et al., 2015a
- Petroni et al., 2014
- Makari et al., 2014
- Ahmed et al., 2013
- Makari et al., 2014
- Zhuang et al., 2013
- Recht et al., 2013
- Niu et al., 2011
- Ricci et al., 2011
- Shi et al., 2014
- Takács et al., 2009
- Rendle, 2012
- Koren et al., 2009
- Rendle et al., 2011
- Koren, 2008
- Karatzoglou et al., 2010
- Riedel et al., 2013
- Chen et al., 2012
- Menon et al., 2011
- Rendle et al., 2009
- Petroni et al., 2015b
- Riedel et al., 2013
- Recht et al., 2013
- Karatzoglou et al., 2010
- Ricci et al., 2011
- Koren et al., 2009
- Koren, 2008
- Takács et al., 2009
- Rendle, 2012
- Petroni et al., 2014
- Makari et al., 2014
- Ahmed et al., 2013
- Makari et al., 2014
- Zhuang et al., 2013
- Recht et al., 2013
- Niu et al., 2011
- Ricci et al., 2011
- Shi et al., 2014
- Takács et al., 2009
- Rendle, 2012
- Koren et al., 2009
- Rendle et al., 2011
- Koren, 2008
- Karatzoglou et al., 2010
- Riedel et al., 2013
- Chen et al., 2012
- Menon et al., 2011
- Rendle et al., 2009
- Petroni et al., 2015b
Contents

1. Distributed Matrix Completion
   1.1 Distributed Stochastic Gradient Descend
   1.2 Input Partitioner
   1.3 Evaluation

2. Context-Aware Matrix Completion
   2.1 Open Relation Extraction
   2.2 Context-Aware Open Relation Extraction
   2.3 Evaluation

3. Conclusion and Outlook
Contents

1. Distributed Matrix Completion
   1.1 Distributed Stochastic Gradient Descend
   1.2 Input Partitioner
   1.3 Evaluation

2. Context-Aware Matrix Completion
   2.1 Open Relation Extraction
   2.2 Context-Aware Open Relation Extraction
   2.3 Evaluation

3. Conclusion and Outlook
Problems of parallel and distributed SGD

- divide the training points (SGD updates) among threads
- SGD updates might depend on each other!

Both threads concurrently update the same latent vector

Lock-based approaches adversely affect concurrency
SGD Taxonomy

- parallel SGD is hardly applicable to very large datasets
  - the time-to-convergence may be too slow
  - the input data may not fit into the main memory
- ASGD has advantages in scalability and efficiency
Asynchronous stochastic gradient descent

- distributed shared-nothing environment (cluster of machines)

- R is split
- vectors are replicated
- replicas concurrently updated
- replicas deviate inconsistently
- synchronization
Bulk Synchronous Processing Model

1. local computation

2. communication

3. barrier synchronization

- currently used by most of the ASGD implementations
Challenges

1. **workload balance**
   - ensure that computing nodes are fed with the same load
   - improve efficiency

2. **minimize communication**
   - minimize vector replicas
   - improve scalability
Graph representation

- The rating matrix describes a graph.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

- Vertices represent users and items.

- Edges represent training points (e.g., ratings).
Graph representation

- the rating matrix describes a graph

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

- find a partitioning of the graph
- assign each part to a different machine
Balanced graph partitioning

- partition $G$ into smaller parts of (ideally) equal size

- a vertex can be cut in multiple ways and span several parts while a cut edge connects only two parts

- computation steps are associated with edges

- $v$-cut better on **power-law graphs** (Gonzalez et al, 2012)
Power-law graphs

- characteristic of real graphs: power-law degree distribution
  - most vertices have few connections while a few have many

\[ P(d) \propto d^{-\alpha} \]

- the probability that a vertex has degree \( d \) is \( P(d) \propto d^{-\alpha} \)
- \( \alpha \) controls the “skewness” of the degree distribution
Balanced Vertex-Cut Graph Partitioning

- \( v \in V \) vertex; \( e \in E \) edge; \( p \in P \) part
- \( A(v) \) set of parts where vertex \( v \) is replicated
- \( \sigma \geq 1 \) tolerance to load imbalance
- the size \( |p| \) of part \( p \) is its edge cardinality

minimize replicas
reduce (1) bandwidth, (2) memory usage and (3) synchronization

balance the load
efficient usage of available computing resources

\[
\min \frac{1}{|V|} \sum_{v \in V} |A(v)| \\
\text{s.t. } \max_{p \in P} |p| < \sigma \frac{|E|}{|P|}
\]

The objective function is the replication factor (RF)
- average number of replicas per vertex
Streaming Setting

- input data is a list of edges, consumed in streaming fashion, requiring only a **single pass**

- handle graphs that don’t fit in the main memory
- impose minimum overhead in time
- scalable, easy parallel implementations
- assignment decision taken cannot be later changed
Streaming algorithms

- History-agnostic
- History-aware

- Power-law-agnostic
- Power-law-aware

- Greedy (Gonzalez et al., 2012)
- PDS (Jain et al., 2013)
- Grid (Jain et al., 2013)
- Hashing (Gonzalez et al., 2012)
- DBH (Xie et al., 2014)

Less replicas → Better balance
Streaming algorithms - contributions

- **History-agnostic**
  - Greedy (Gonzalez et al., 2012)

- **History-aware**
  - Grid (Jain et al., 2013)
  - Hashing (Gonzalez et al., 2012)
  - PDS (Jain et al., 2013)
  - HDRF (Petroni et al., 2015)

- **Power-law-agnostic**
  - DBH (Xie et al., 2014)

- **Power-law-aware**

Graphical representation:

- **Less replicas**
- **Better balance**
favors the replication of high-degree vertices
the number of high-degree vertices in power-law graphs is very small
overall reduction of the replication factor
HDRF: High Degree are Replicated First

- in the context of robustness to network failure
- if few high-degree vertices are removed from a power-law graph then it is turned into a set of isolated clusters
- focus on the locality of low-degree vertices
The HDRF Algorithm

incoming edge

vertex without replicas

vertex with replicas

case 1

vertices not assigned to any part
The HDRF Algorithm

*incoming edge*

vertex without replicas

vertex with replicas

**case 1**

place e in the least loaded part
The HDRF Algorithm

**incoming edge**

- **case 1**
  - place $e$ in the least loaded part

- **case 2**
  - only one vertex has been assigned

- vertex without replicas
- vertex with replicas
The HDRF Algorithm

**case 1**
place $e$ in the least loaded part

**case 2**
place $e$ in the part
The HDRF Algorithm

*incoming edge*

**case 1**
place e in the least loaded part

**case 2**
place e in the part

**case 3**
vertices assigned, common part
The HDRF Algorithm

**Case 1**
place $e$ in the least loaded part

**Case 2**
place $e$ in the part

**Case 3**
place $e$ in the intersection
Create Replicas

\[ e \]

**case 4**

*empty intersection*
Create Replicas

**Case 4**

- *Empty Intersection*

**Standard Greedy Solution**

- *Least Loaded Part in the Union*
Create Replicas

**Case 4**

- **Empty intersection**

**Standard Greedy solution**

- **Case 4**
  - **Least loaded part in the union**

**HDRF**

- **Case 4**
  - **Replicate vertex with highest degree**

\[
\delta(v_1) > \delta(v_2)
\]
Experiments - Settings

- standalone partitioner
  - VGP, a software package for one-pass vertex-cut balanced graph partitioning
  - measure the performance: replication and balancing

- GraphLab
  - HDRF has been integrated in GraphLab PowerGraph 2.2
  - measure the impact on the execution time of graph computation in a distributed graph computing frameworks

- stream of edges in random order
Experiments - Datasets

- real-word graphs

| Dataset          | $|V|$  | $|E|$ |
|------------------|-------|------|
| MovieLens 10M    | 80.6K | 10M  |
| Netflix          | 497.9K| 100.4M|
| Tencent Weibo    | 1.4M  | 140M |
| twitter-2010     | 41.7M | 1.47B|

- synthetic graphs

  1M vertices

  60M to 3M edges
Results - Synthetic Graphs Replication Factor

- 128 parts

![Graph showing replication factor vs. alpha for different graph types, including HDRF, PDS, grid, greedy, and DBH. The y-axis represents replication factor, and the x-axis represents alpha values ranging from 1.8 to 4.0. The graph compares skewed and homogeneous distributions.]
Results - Synthetic Graphs Replication Factor

- 128 parts

![Graph showing replication factor vs. replication scheme and alpha value.](image)

- HDRF
- PDS $|P|=133$
- Grid $|P|=121$
- Greedy
- DBH

- Skewed graphs have less edges and are less dense, making them easier to partition.

- The x-axis represents the replication factor, and the y-axis represents the alpha value.
Results - Synthetic Graphs Replication Factor

- 128 parts

![Graph](image)

- More edges
- More dense
- Difficult to partition

- HDRF
- PDS $|P|=133$
- Grid $|P|=121$
- Greedy
- DBH

- Power-law-agnostic

- Skewed
- Homogeneous

- Replication factor

- Alpha

- More dense
Results - Synthetic Graphs Replication Factor

- 128 parts

![Graph showing replication factor vs. alpha for different graph types and network topologies.](image-url)
Results - Real-Word Graphs Replication Factor

- 133 parts

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PDS</th>
<th>DBH</th>
<th>Greedy</th>
<th>HDRF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tencent Weibo</td>
<td>7.9</td>
<td>2.8</td>
<td>1.3</td>
<td>1.5</td>
</tr>
<tr>
<td>Netflix</td>
<td>4.9</td>
<td>7.1</td>
<td>4.9</td>
<td>6.8</td>
</tr>
<tr>
<td>MovieLens 10M</td>
<td>8.1</td>
<td>8.1</td>
<td>9.1</td>
<td>5.9</td>
</tr>
<tr>
<td>twitter-2010</td>
<td>7.9</td>
<td>7.1</td>
<td>12.9</td>
<td>11.5</td>
</tr>
</tbody>
</table>
Results - Load Relative Standard Deviation

- MovieLens 10M

![Graph showing load relative standard deviation for different methods.]

- PDS
- grid
- greedy
- DBH
- HDRF
- hashing
Results - Load Relative Standard Deviation

MovieLens 10M

load relative standard deviation (%)

- PDS
- grid
- greedy
- DBH
- HDRF
- history-agnostic
- hashing

parts

load relative standard deviation (%)
Results - Load Relative Standard Deviation

MovieLens 10M

![Graph showing load relative standard deviation](image)
Results - Graph Algorithm Runtime Speedup

- **ASGD** algorithm for collaborative filtering on *Tencent Weibo*

![Graph showing speedup with different number of parts](image)

- The speedup is proportional to both:
  - the advantage in replication factor
  - the actual network usage of the algorithm
Summary

- HDRF is a simple and remarkably effective one-pass vertex-cut graph partitioning algorithm
  - achieves on average a replication factor
    - about 40% smaller than DBH
    - more than 50% smaller than greedy
    - almost 3× smaller than PDS
    - more than 4× smaller than grid
    - almost 14× smaller than hashing
  - close to optimal load balance
  - ASGD execution time is up to 2× faster when using HDRF
  - HDRF has been included in GraphLab!
1. Distributed Matrix Completion
   1.1 Distributed Stochastic Gradient Descend
   1.2 Input Partitioner
   1.3 Evaluation

2. Context-Aware Matrix Completion
   2.1 Open Relation Extraction
   2.2 Context-Aware Open Relation Extraction
   2.3 Evaluation

3. Conclusion and Outlook
Knowledge bases

- New generation algorithms for web information retrieval make use of a **knowledge base (KB)** to increase their accuracy.

- match the words in a query to real world entities (e.g., a person, a location, etc)

- use real world connections among these entities

- improve the task of providing the user with the proper content he was looking for

- KB represented as a graph
Knowledge bases challenges

- large-scale example: LOD contains over 60B facts

- despite the size, KBs are far from being complete
  - 75% people have unknown nationality in Freebase
  - 71% people with place of birth attribute missing in Freebase

- data not only incomplete but also uncertain, noisy or false

- challenges:
  - fill missing information
  - remove incorrect facts

- idea: scan the web extracting new information
Open relation extraction

- open relation extraction is the task of extracting new facts for a potentially unbounded set of relations from various sources

The New York Times

Enrico Fermi - The last breath of Caesar

natural language text

knowledge bases
Input data: facts from natural language text

Enrico Fermi was a professor in theoretical physics at Sapienza University of Rome.
Input data: facts from knowledge bases

KB fact

employee(Fermi,Sapienza)

KB relation

"professor at"(Fermi,Sapienza)
Matrix completion for open relation extraction

| (Caesar, Rome) | 1  |
| (Fermi, Rome) | 1  |
| (Fermi, Sapienza) | 1 | 1 |
| (de Blasio, NY) | 1  |

- **tuples x relations**
  - born in
  - professor at
  - mayor of
  - employee

- **surface relation**
- **KB relation**
Matrix completion for open relation extraction

<table>
<thead>
<tr>
<th>(Caesar, Rome)</th>
<th>1</th>
<th>?</th>
<th>?</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Fermi, Rome)</td>
<td>1</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>(Fermi, Sapienza)</td>
<td>?</td>
<td>1</td>
<td>?</td>
<td>1</td>
</tr>
<tr>
<td>(de Blasio, NY)</td>
<td>?</td>
<td>?</td>
<td>1</td>
<td>?</td>
</tr>
</tbody>
</table>

- **born in**
- **professor at**
- **mayor of**
- **employee**

*Surface relation*

*KB relation*

**tuples x relations**
we propose CORE (context-aware open relation extraction) that integrates contextual information into such models to improve prediction performance
Tom Peloso joined Modest Mouse to record their fifth studio album.
CORE - latent representation of variables

- associates latent representations $\mathbf{f}_v$ with each variable $v \in V$

<table>
<thead>
<tr>
<th>tuple</th>
<th>(Peloso, Modest Mouse)</th>
</tr>
</thead>
<tbody>
<tr>
<td>relation</td>
<td>join</td>
</tr>
<tr>
<td>entities</td>
<td>Peloso</td>
</tr>
<tr>
<td>context</td>
<td>Modest Mouse</td>
</tr>
<tr>
<td></td>
<td>person</td>
</tr>
<tr>
<td></td>
<td>organization</td>
</tr>
<tr>
<td></td>
<td>Music</td>
</tr>
<tr>
<td></td>
<td>record</td>
</tr>
<tr>
<td></td>
<td>album</td>
</tr>
</tbody>
</table>

latent factor vectors
CORE - modeling facts

- models the input data in terms of a matrix in which each row corresponds to a fact $x$ and each column to a variable $v$
- groups columns according to the type of the variables
- in each row the values of each column group sum up to unity

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>relations</th>
<th>tuples</th>
<th>entities</th>
<th>tuple types</th>
<th>tuple topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caesar,Rome</td>
<td>Fermi,Rome</td>
<td>Fermi,Sapienza</td>
<td>person, organization</td>
<td>person, location</td>
</tr>
<tr>
<td>physics</td>
<td>history</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Surface KB Context
CORE - modeling context

- aggregates and normalizes contextual information by tuple
  - a fact can be observed multiple times with different context
  - there is no context for new facts (never observed in input)
- this approach allows us to provide comprehensive contextual information for both observed and unobserved facts
CORE - factorization model

- uses factorization machines as underlying framework
- associates a score $s(x)$ with each fact $x$

$$s(x) = \sum_{v_1 \in V} \sum_{v_2 \in V \setminus \{v_1\}} x_{v_1} x_{v_2} f_{v_1}^T f_{v_2}$$

- weighted pairwise interactions of latent factor vectors
CORE - parameter estimation

- **parameters:** $\Theta = \{ f_v \mid v \in V \}$
- Bayesian personalized ranking, all observations are positive
- goal: produce a ranked list of tuples for each relation

- pairwise approach, $x$ is more likely to be true than $x-$

$$\maximize \sum_{x} f(s(x) - s(x-))$$

- stochastic gradient ascent

$$\Theta \leftarrow \Theta + \eta \nabla_{\Theta} \left( \right)$$
Experiments - dataset

440k facts extracted from The New York Times corpus

15k facts from Freebase

Contextual information

- article metadata
  - news desk (e.g., foreign desk)
  - descriptors (e.g., finances)
  - online section (e.g., sports)
  - section (e.g., a, d)
  - publication year

- entity type
  - person
  - organization
  - location
  - miscellaneous

- bag-of-word sentences where the fact has been extracted

letters to indicate contextual information considered
Experiments - methodology

- we consider (to keep experiments feasible):
  - 10k tuples
  - 19 Freebase relations
  - 10 surface relations

- for each relation and method:
  - we rank the tuples subsample
  - we consider the top-100 predictions and label them manually

- evaluation metrics:
  - number of true facts
  - MAP (quality of the ranking)

- methods:
  - PITF, tensor factorization method (designed to work in-KB)
  - NFE, matrix completion method (best context-agnostic)
  - CORE, uses relations, tuples and entities as variables
  - CORE+m, +t, +w, +mt, +mtw
# Results - Freebase relations

<table>
<thead>
<tr>
<th>Relation</th>
<th>#</th>
<th>PITF</th>
<th>NFE</th>
<th>CORE</th>
<th>CORE+m</th>
<th>CORE+t</th>
<th>CORE+w</th>
<th>CORE+mt</th>
<th>CORE+mtw</th>
</tr>
</thead>
<tbody>
<tr>
<td>person/company</td>
<td>208</td>
<td>70 (0.47)</td>
<td>92 (0.81)</td>
<td>91 (0.83)</td>
<td>90 (0.84)</td>
<td>91 (0.87)</td>
<td>92 (0.87)</td>
<td>95 (0.93)</td>
<td>96 (0.94)</td>
</tr>
<tr>
<td>person/place_of_birth</td>
<td>117</td>
<td>1 (0.0)</td>
<td>92 (0.9)</td>
<td>90 (0.88)</td>
<td>92 (0.9)</td>
<td>92 (0.9)</td>
<td>89 (0.87)</td>
<td>93 (0.9)</td>
<td>92 (0.9)</td>
</tr>
<tr>
<td>location/containedby</td>
<td>102</td>
<td>7 (0.0)</td>
<td>63 (0.47)</td>
<td>62 (0.47)</td>
<td>63 (0.46)</td>
<td>61 (0.47)</td>
<td>61 (0.44)</td>
<td>62 (0.49)</td>
<td>68 (0.55)</td>
</tr>
<tr>
<td>parent/child</td>
<td>88</td>
<td>9 (0.01)</td>
<td>64 (0.6)</td>
<td>64 (0.56)</td>
<td>64 (0.59)</td>
<td>64 (0.62)</td>
<td>64 (0.57)</td>
<td>67 (0.67)</td>
<td>68 (0.63)</td>
</tr>
<tr>
<td>person/place_of_death</td>
<td>71</td>
<td>1 (0.0)</td>
<td>67 (0.93)</td>
<td>67 (0.92)</td>
<td>69 (0.94)</td>
<td>67 (0.93)</td>
<td>67 (0.92)</td>
<td>69 (0.94)</td>
<td>67 (0.92)</td>
</tr>
<tr>
<td>person/persparents</td>
<td>67</td>
<td>20 (0.1)</td>
<td>51 (0.64)</td>
<td>52 (0.62)</td>
<td>51 (0.61)</td>
<td>49 (0.64)</td>
<td>47 (0.6)</td>
<td>53 (0.67)</td>
<td>53 (0.65)</td>
</tr>
<tr>
<td>author/works_written</td>
<td>65</td>
<td>24 (0.08)</td>
<td>45 (0.59)</td>
<td>49 (0.62)</td>
<td>51 (0.69)</td>
<td>50 (0.68)</td>
<td>50 (0.68)</td>
<td>51 (0.7)</td>
<td>52 (0.67)</td>
</tr>
<tr>
<td>person/nationality</td>
<td>61</td>
<td>21 (0.08)</td>
<td>25 (0.19)</td>
<td>27 (0.17)</td>
<td>28 (0.2)</td>
<td>26 (0.2)</td>
<td>29 (0.19)</td>
<td>27 (0.18)</td>
<td>27 (0.21)</td>
</tr>
<tr>
<td>neighbor./neighborhood_of</td>
<td>39</td>
<td>3 (0.0)</td>
<td>24 (0.44)</td>
<td>23 (0.45)</td>
<td>26 (0.5)</td>
<td>27 (0.47)</td>
<td>27 (0.49)</td>
<td>30 (0.51)</td>
<td>30 (0.52)</td>
</tr>
</tbody>
</table>

Average MAP$^{100}$ #

|          | 0.09 | 0.46 | 0.47 | 0.49 | 0.47 | 0.49 | 0.49 | 0.51  |

Weighted Average MAP$^{100}$ #

|          | 0.14 | 0.64 | 0.64 | 0.66 | 0.67 | 0.66 | 0.70 | 0.70  |

![Weighted Average MAP Graph]

52 of 58
## Results - surface relations

<table>
<thead>
<tr>
<th>Relation</th>
<th>#</th>
<th>PITF</th>
<th>NFE</th>
<th>CORE</th>
<th>CORE+m</th>
<th>CORE+t</th>
<th>CORE+w</th>
<th>CORE+mt</th>
<th>CORE+mtw</th>
<th>Average MAP¹⁰⁰</th>
<th>Weighted Average MAP¹⁰⁰</th>
</tr>
</thead>
<tbody>
<tr>
<td>head</td>
<td>162</td>
<td>34 (0.18)</td>
<td>80 (0.66)</td>
<td>83 (0.66)</td>
<td>82 (0.63)</td>
<td>76 (0.57)</td>
<td>77 (0.57)</td>
<td>83 (0.69)</td>
<td>88 (0.73)</td>
<td>0.04</td>
<td>0.08</td>
</tr>
<tr>
<td>scientist</td>
<td>144</td>
<td>44 (0.17)</td>
<td>76 (0.6)</td>
<td>74 (0.55)</td>
<td>73 (0.56)</td>
<td>74 (0.6)</td>
<td>73 (0.59)</td>
<td>78 (0.66)</td>
<td>78 (0.69)</td>
<td>0.53</td>
<td>0.62</td>
</tr>
<tr>
<td>base</td>
<td>133</td>
<td>10 (0.01)</td>
<td>85 (0.71)</td>
<td>86 (0.71)</td>
<td>86 (0.78)</td>
<td>88 (0.79)</td>
<td>85 (0.75)</td>
<td>83 (0.76)</td>
<td>89 (0.8)</td>
<td>0.53</td>
<td>0.63</td>
</tr>
<tr>
<td>visit</td>
<td>118</td>
<td>4 (0.0)</td>
<td>73 (0.6)</td>
<td>75 (0.61)</td>
<td>76 (0.64)</td>
<td>80 (0.68)</td>
<td>74 (0.64)</td>
<td>75 (0.66)</td>
<td>82 (0.74)</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>attend</td>
<td>92</td>
<td>11 (0.02)</td>
<td>65 (0.58)</td>
<td>64 (0.59)</td>
<td>65 (0.63)</td>
<td>62 (0.6)</td>
<td>66 (0.63)</td>
<td>62 (0.58)</td>
<td>69 (0.64)</td>
<td>0.53</td>
<td>0.62</td>
</tr>
<tr>
<td>adviser</td>
<td>56</td>
<td>2 (0.0)</td>
<td>42 (0.56)</td>
<td>47 (0.58)</td>
<td>44 (0.58)</td>
<td>43 (0.59)</td>
<td>45 (0.63)</td>
<td>43 (0.53)</td>
<td>44 (0.63)</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>criticize</td>
<td>40</td>
<td>5 (0.0)</td>
<td>31 (0.66)</td>
<td>33 (0.62)</td>
<td>33 (0.7)</td>
<td>33 (0.67)</td>
<td>33 (0.61)</td>
<td>35 (0.69)</td>
<td>37 (0.69)</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>support</td>
<td>33</td>
<td>3 (0.0)</td>
<td>19 (0.27)</td>
<td>22 (0.28)</td>
<td>18 (0.21)</td>
<td>19 (0.28)</td>
<td>22 (0.27)</td>
<td>23 (0.27)</td>
<td>21 (0.27)</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>praise</td>
<td>5</td>
<td>0 (0.0)</td>
<td>2 (0.0)</td>
<td>2 (0.01)</td>
<td>4 (0.03)</td>
<td>3 (0.01)</td>
<td>3 (0.02)</td>
<td>5 (0.03)</td>
<td>2 (0.01)</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>vote</td>
<td>3</td>
<td>2 (0.01)</td>
<td>3 (0.63)</td>
<td>3 (0.63)</td>
<td>3 (0.32)</td>
<td>3 (0.49)</td>
<td>3 (0.51)</td>
<td>3 (0.59)</td>
<td>3 (0.64)</td>
<td>0.08</td>
<td>0.08</td>
</tr>
<tr>
<td>Average MAP¹⁰⁰</td>
<td></td>
<td>0.04</td>
<td>0.53</td>
<td>0.53</td>
<td>0.51</td>
<td>0.53</td>
<td>0.53</td>
<td>0.55</td>
<td>0.59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Weighted Average MAP¹⁰⁰</td>
<td></td>
<td>0.08</td>
<td>0.62</td>
<td>0.61</td>
<td>0.63</td>
<td>0.63</td>
<td>0.61</td>
<td>0.65</td>
<td>0.70</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Graph showing weighted average MAP]
Anecdotal results

author(x,y)

<table>
<thead>
<tr>
<th>ranked list of tuples</th>
<th>similar relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Winston Groom, Forrest Gump)</td>
<td>0.98 “reviews x by y”(x,y)</td>
</tr>
<tr>
<td>2 (D. M. Thomas, White Hotel)</td>
<td>0.97 “book by”(x,y)</td>
</tr>
<tr>
<td>3 (Roger Rosenblatt, Life Itself)</td>
<td>0.95 “author of”(x,y)</td>
</tr>
<tr>
<td>4 (Edmund White, Skinned Alive)</td>
<td>0.95 “‘s novel”(x,y)</td>
</tr>
<tr>
<td>5 (Peter Manso, Brando: The Biography)</td>
<td>0.95 “‘s book”(x,y)</td>
</tr>
</tbody>
</table>

“scientist at”(x,y)

<table>
<thead>
<tr>
<th>ranked list of tuples</th>
<th>similar relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (Riordan Roett, Johns Hopkins University)</td>
<td>0.87 “scientist”(x,y)</td>
</tr>
<tr>
<td>2 (Dr. R. M. Roberts, University of Missouri)</td>
<td>0.84 “scientist with”(x,y)</td>
</tr>
<tr>
<td>3 (Linda Mayes, Yale University)</td>
<td>0.80 “professor at”(x,y)</td>
</tr>
<tr>
<td>4 (Daniel T. Jones, Cardiff Business School)</td>
<td>0.79 “scientist for”(x,y)</td>
</tr>
<tr>
<td>5 (Russell Ross, University of Iowa)</td>
<td>0.78 “neuroscientist at”(x,y)</td>
</tr>
</tbody>
</table>

- semantic similarity of relations is one aspect of our model
- similar relations treated differently in different contexts
Contents

1. Distributed Matrix Completion
   1.1 Distributed Stochastic Gradient Descend
   1.2 Input Partitioner
   1.3 Evaluation

2. Context-Aware Matrix Completion
   2.1 Open Relation Extraction
   2.2 Context-Aware Open Relation Extraction
   2.3 Evaluation

3. Conclusion and Outlook
Conclusion

- we tackle the **scalability** and the **quality** of MC
- we investigate graph partitioning techniques for ASGD
- we propose HDRF, one-pass $v$-cut graph partitioning alg
  - exploit power-law nature of real-word graphs
  - provides minimum replicas with close to optimal load balance
  - significantly reduces the time needed to perform computation
- we propose CORE, a matrix completion model for open relation extraction that incorporates contextual information
  - based on factorization machines and BPR
  - extensible model, additional information can be integrated
  - exploiting context significantly improve prediction quality
- all code released [https://github.com/fabiopetroni](https://github.com/fabiopetroni)
Overview - Future work

Regularized Squared Loss
- Positive and negative evidence
- Multi-value revealed entries

Bayesian Personalized Ranking
- Only positive evidence
- Single-value revealed entries

- Petroni et al., 2015a
- Petroni et al., 2014
- Makari et al., 2014
- Ahmed et al., 2013
- Makari et al., 2014
- Zhuang et al., 2013
- Recht et al., 2013
- Niu et al., 2011
- Ricci et al., 2011
- Shi et al., 2014
- Karatzoglou et al., 2010
- Rendle, 2012
- Koren et al., 2009
- Rendle et al., 2011
- Koren, 2008
- Riedel et al., 2013
- Chen et al., 2012
- Menon et al., 2011
- Rendle et al., 2009
- Petroni et al., 2015b
Future Directions

- distribute training for context-aware matrix completion
  - asynchronous approach: local vector copies, synchronization
  - challenging input placement, the input describes an hypergraph

- adaptation of the HDRF algorithm to hypergraphs
  - do high degree vertices play a crucial role?

- distributed Bayesian personalized ranking
  - sampling of a negative counterpart for each training point
  - sampling from the local portion of the dataset in current node?
Thank you!

Questions?

Fabio Petroni
Sapienza University of Rome, Italy