Scalable Network Analysis

Inderjit S. Dhillon
University of Texas at Austin

COMAD, Ahmedabad, India
Dec 20, 2013
Outline

- Unstructured Data - Scale & Diversity
- Evolving Networks
- Machine Learning Problems arising in Networks
  - Recommender Systems
  - Link Prediction
  - Sign Prediction
- Formulation as Missing Value Estimation
- Scalable Algorithms
  - NOMAD: Distributed matrix completion algorithm
- Results on Applications
- Conclusions
Structured Data

- Data organized into fields: Relational databases, spreadsheets, XML
- Highly optimized for storage & retrieval (e.g. using SQL)
Structured Data

- Focus is on data format, efficient storage & search
- Less or no uncertainty in semantics: e.g. businesses know the fields of the data
Unstructured Data

Modern data is **unstructured** and **diverse**

Networks, Graphs

Text

Images, Videos
Unstructured Data

Much greater growth rate
Unstructured Data

• Dynamic aspects of unstructured data:
  • Constantly evolving
  • Uncertainties abound: *What* should I ask of the data?
  • Seek *insights*
• Heterogeneity renders traditional database models inadequate
Unstructured Data

• Buzzwords - “Big Data” & “Data Science”
  • *Machine Learning*: Predictive models for data
  • *Engineering perspective*: Scale matters
Network Graphs

Social networks (Friendship)

Bipartite networks (Membership, Ratings, etc.)

Gene networks (Functional interaction)
Graph Evolution

- Social networks are highly dynamic
- Constantly grow, change quickly over time
  - Users arrive/leave, relationships form/dissolve
- Understanding graph evolution is important
Graphs meet Machine Learning

- **Network analysis**: Understanding structure & evolution of networks
- Formulate predictive problems on the adjacency matrix of the graph
- Confluence of graph theory & machine learning
Scalable Network Analysis

Recommender Systems

Rating Matrix

Facebook growth

Link Prediction

Netflix problem:
100M ratings,
0.5M users,
20K movies

A Toy Problem In Comparison!
# Recommender Systems

## Rating Matrix

<table>
<thead>
<tr>
<th>Users</th>
<th>Movies</th>
<th>Movies</th>
<th>Movies</th>
<th>Movies</th>
<th>Movies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Movie 1</td>
<td>Movie 2</td>
<td>Movie 3</td>
<td>Movie 4</td>
<td>Movie 5</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Users</th>
<th>Movies</th>
<th>Movies</th>
<th>Movies</th>
<th>Movies</th>
<th>Movies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Movie 6</td>
<td>Movie 7</td>
<td>Movie 8</td>
<td>Movie 9</td>
<td>Movie 10</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

## Diagram

The diagram illustrates the connections between users and movies, with ratings shown in the matrix.
Link Prediction in Social Networks

- **Problem**: Infer *missing* relationships from a given snapshot of the network
Predicting gene-disease links

Candidates
1. AQP1
2. AQP6
3. AQP5
4. MIP
...
40. MYBL2

Dotted lines represent gene-phenotype relationships, solid lines represent gene-gene relationships, and dashed lines represent candidate links.
Signed Social Networks

- **Sign Prediction Problem**: Given a snapshot of the signed social network, predict the signs of missing edges.
Formulation as Missing Value Estimation
Low-rank Matrix Completion
Low-rank Matrix Completion

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

Users

Movies
Low-rank Matrix Completion
Low-rank Matrix Completion

\[
\min_{W \in \mathbb{R}^{m \times k}, H \in \mathbb{R}^{n \times k}} \sum_{(i,j) \in \Omega} \left( A_{ij} - W_i H_j^T \right)^2 + \lambda \left( \| W \|_F^2 + \| H \|_F^2 \right)
\]
Low-rank Matrix Completion

\[
\min_{W \in \mathbb{R}^{m \times k}, H \in \mathbb{R}^{n \times k}} \sum_{(i,j) \in \Omega} (A_{ij} - W_i H_j^T)^2 + \lambda (\|W\|_F^2 + \|H\|_F^2)
\]
Low-rank Matrix Completion

\[
\min_{W \in \mathbb{R}^{m \times k}, H \in \mathbb{R}^{n \times k}} \sum_{(i,j) \in \Omega} (A_{ij} - W_i H_j^T)^2 + \lambda(\|W\|_F^2 + \|H\|_F^2)
\]
Link Prediction

- Can be posed as matrix completion problem

\[
A^{(t)} = \begin{bmatrix}
. & 1 & 1 & . & . \\
1 & . & 1 & 1 & 1 \\
1 & 1 & . & . & 1 \\
. & 1 & . & . & . \\
. & 1 & 1 & . & . \\
\end{bmatrix} \approx \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}
\]

- **Issue**: Only positive relationships are observed

<table>
<thead>
<tr>
<th>Test Link</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4,5)</td>
<td>1</td>
</tr>
<tr>
<td>(1,4)</td>
<td>1</td>
</tr>
<tr>
<td>(3,4)</td>
<td>1</td>
</tr>
<tr>
<td>(1,5)</td>
<td>1</td>
</tr>
</tbody>
</table>
Link Prediction

• Formulate Biased Matrix Completion Problem:

\[
\min_{W \in \mathbb{R}^{n \times k}, H \in \mathbb{R}^{n \times k}} \sum_{(i,j) \in \Omega} (A_{ij} - W_i H_j^T)^2 + \alpha \sum_{(i,j) \notin \Omega} (A_{ij} - W_i H_j^T)^2 + \lambda (\|W\|_F^2 + \|H\|_F^2)
\]

\[
A^{(t)} = \begin{bmatrix}
. & 1 & 1 & . & . \\
1 & . & 1 & 1 & 1 \\
1 & 1 & . & . & 1 \\
. & 1 & . & . & . \\
. & 1 & 1 & . & . \\
\end{bmatrix} \approx \begin{bmatrix}
0.85 \\
1.09 \\
0.97 \\
0.69 \\
0.85 \\
\end{bmatrix}
\]

<table>
<thead>
<tr>
<th>Test Link</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>(4,5)</td>
<td>0.59</td>
</tr>
<tr>
<td>(1,4)</td>
<td>0.59</td>
</tr>
<tr>
<td>(3,4)</td>
<td>0.67</td>
</tr>
<tr>
<td>(1,5)</td>
<td>0.72</td>
</tr>
</tbody>
</table>
Signed Social Networks

<table>
<thead>
<tr>
<th>Balanced</th>
<th>Not balanced</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="%E5%B9%B3%E8%A1%A1%E7%9A%84%E7%BD%91%E7%BB%9C%E7%A4%BA%E6%84%8F%E5%9B%BE" alt="Diagram" /></td>
<td><img src="%E9%9D%9E%E5%B9%B3%E8%A1%A1%E7%9A%84%E7%BD%91%E7%BB%9C%E7%A4%BA%E6%84%8F%E5%9B%BE" alt="Diagram" /></td>
</tr>
</tbody>
</table>

- **Friend of a friend is a friend**
- **Enemy of an enemy is a friend**

- **Social Balance [Harary, 1953]:**
  - In real-world signed networks, triangles tend to be *balanced*
Theorem: All triangles in a network are balanced if and only if there exist two antagonistic groups.
## Signed Social Networks

<table>
<thead>
<tr>
<th>Weakly Balanced</th>
<th>Not balanced</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Diagram" /></td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
<tr>
<td><img src="image3" alt="Diagram" /></td>
<td><img src="image4" alt="Diagram" /></td>
</tr>
<tr>
<td><img src="image5" alt="Diagram" /></td>
<td><img src="image6" alt="Diagram" /></td>
</tr>
<tr>
<td><img src="image7" alt="Diagram" /></td>
<td><img src="image8" alt="Diagram" /></td>
</tr>
</tbody>
</table>

- Relaxation: **Weak balance**
- Allow triangles with all negative edges
Signed Social Networks

**Theorem**: All triangles in a network are weakly balanced if and only if there exist \( k \) antagonistic groups.
Sign Prediction

**Theorem:** A $k$-weakly balanced signed network has rank at most $k$.

- Sign inference can be posed as low-rank matrix completion

**Theorem:** If there are no “small” groups, the underlying network can be exactly recovered, under certain conditions.

K. Chiang *et al.* *Prediction and Clustering in Signed Networks: A Local to Global Perspective*. To appear in JMLR.
Scalable Algorithms
Stochastic Gradient

- Sample random index \((i,j)\) and update corresponding factors:

\[
\begin{align*}
    w_i &\leftarrow w_i - \eta ((A_{ij} - w_i^T h_j) h_j + \lambda w_i) \\
    h_j &\leftarrow h_j - \eta ((A_{ij} - w_i^T h_j) w_i + \lambda h_j)
\end{align*}
\]

- Time per update \(O(k)\)

- Effective for very large-scale problems
Distributed Stochastic Gradient Descent (DSGD) [Gemulla et al. KDD 2011]

- Decoupled updates
- Easy to parallelize
- But communication & computation are interleaved
DSGD
DSGD

Curse of the last reducer
DSGD
NOMAD

- **Goal**: Keep CPU & network *simultaneously* busy.

- Asynchronous distributed solution.

- Non-locking
  - Stochastic
  - Multi-machine algorithm for
  - Asynchronous &
  - Decentralized matrix factorization

- Asynchronous distributed solution.
NOMAD

Nomadic variables queue

Native variables

Workers
Nomadic variables queue

Native variables

Push $h_4$

Workers

Inderjit S. Dhillon University of Texas at Austin
Scalable Network Analysis
NOMAD
NOMAD
NOMAD
NOMAD
NOMAD
NOMAD
NOMAD
NOMAD
NOMAD Algorithm

1. **Initialize**: Randomly assign \( h_j \) columns to worker queues
2. **Parallel Foreach** \( q \) in \( \{1, 2, \ldots, p\} \)
   3. **If** queue[\( q \)] not empty **then**
   4. \( (j, h_j) \leftarrow \text{queue}[q].\text{pop}() \)
   5. **for** (\( i, j \)) in \( \Omega_{\Omega_j}^{(q)} \) **do**
   6. **Do** SGD updates
   7. **end for**
3. **Sample** \( q' \) uniformly from \( \{1, 2, \ldots, p\} \)
4. \( \text{queue}[q'].\text{push}((j, h_j)) \)
5. **end if**
6. **Parallel End**
NOMAD Algorithm

1. **Initialize**: Randomly assign $h_j$ columns to worker queues
2. **Parallel Foreach** $q$ in \{1,2,...,$p$\}
   3. If queue[$q$] not empty then
      4. $(j, h_j) \leftarrow$ queue[$q$].pop()
      5. for $(i, j)$ in $\Omega_{q_j}$ do
         6. Do SGD update
      7. end for
   8. Sample $q'$ uniformly from \{1,2,...,$p$\}
   9. queue[$q'$].push($(j, h_j)$)
10. end if
11. Parallel End

**Distributed setting**: Write over the network

**Concurrent object**
Algorithm Complexity

• Average space required per worker:
  \[ O\left(\frac{mk}{p} + \frac{nk}{p} + \frac{|\Omega|}{p}\right) \]

• Average time for one sweep:
  \[ O\left(\frac{|\Omega|k}{p}\right) \]
Results on Applications
Recommender Systems

Netflix dataset: 2,649,429 users, 17,770 movies, ~100M ratings

Multicore

Distributed
Recommender Systems

Synthetic dataset: ~85M users, 17,770 items, ~8.5B observations
Link Prediction

- Flickr dataset: 1.9M users & 42M links.
- Test set: sampled 5K users.

<table>
<thead>
<tr>
<th>Method</th>
<th>Prec</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preferential Attachment</td>
<td>1.02</td>
<td>0.6981</td>
</tr>
<tr>
<td>Common Neighbor</td>
<td>7.08</td>
<td>0.8649</td>
</tr>
<tr>
<td>Adamic-Adar</td>
<td>7.29</td>
<td>0.8758</td>
</tr>
<tr>
<td>Random Walk w/ Restarts</td>
<td>5.49</td>
<td>0.7872</td>
</tr>
<tr>
<td>Logistic Regression*</td>
<td>2.54</td>
<td>0.7115</td>
</tr>
<tr>
<td>Katz</td>
<td>7.17</td>
<td>0.8429</td>
</tr>
<tr>
<td>MSLP-Katz</td>
<td>13.34</td>
<td>0.8924</td>
</tr>
<tr>
<td>MF</td>
<td>12.05</td>
<td>0.9078</td>
</tr>
<tr>
<td>MSLP-MF</td>
<td>13.07</td>
<td>0.9145</td>
</tr>
</tbody>
</table>

*using network-based features.
Predicting Gene-Disease Links

![Graphs showing comparison of different methods for predicting gene-disease links.](image-url)
Sign Prediction

- Epinions dataset (+ve & -ve reviews):
  - 131K nodes, 840K edges, 15% edges negative.
  - MF-ALS is faster and achieves higher accuracy.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOC-3</td>
<td>560.64 s.</td>
</tr>
<tr>
<td>HOC-4</td>
<td>6156.8 s.</td>
</tr>
<tr>
<td>HOC-5</td>
<td>&gt; 10,000 s.</td>
</tr>
<tr>
<td>MF-ALS</td>
<td>28.67 s.</td>
</tr>
</tbody>
</table>

MF-ALS takes 455 secs on network with 1.1M nodes & 120M edges
Conclusions

- Rapid growth of unstructured data demands scalable machine learning solutions for analysis
- Machine learning problems arising in network analysis can be cast in the matrix completion framework
- Our proposed asynchronous distributed algorithm NOMAD outperforms state-of-the-art matrix completion solvers
- Beyond Matrix Completion: Asynchronous distributed framework for solving machine learning problems