A latent factor model for highly multi-relational data

Rodolphe Jenatton† Nicolas Le Roux* Antoine Bordes○ Guillaume Obozinski* (RJ and NLR contributed equally)

CMAP, UMR CNRS 7641, Ecole Polytechnique, Palaiseau, France
INRIA-Sierra Project (INRIA/ENS/CNRS UMR 8548), 23, avenue d’Italie, 75214 Paris, France
Heudiasyc, UMR CNRS 7253, Université de Technologie de Compiegne, France

One minute overview
- Method to model data of the form \{subject, relation, object\}
  - e.g., recommender system, social networks, NLP...
- Interested in a setting with many relation types (≥ 10^4)
- Main idea: Learn latent representations of subjects, relations, and objects
  combined in a trilinear model
- Scalability: Sharing sparse latent factors among relations
- Good empirical performance:
  - Standard tensor-factorization benchmarks
  - Large-scale NLP application

Relational data modeling
Setting:
- n_s subjects \( \{S_i\}_{i \in \mathbb{N}} \)
- n_r relations \( \{R_j\}_{j \in \mathbb{N}} \)
- n_o objects \( \{O_k\}_{k \in \mathbb{N}} \)
- A relationship exists for the triplet \((S_i, R_j, O_k)\) if \(R_{j}(S_i, O_k) = 1\)
  - e.g., a subject and a direct object linked through a transitive verb in NLP
Goal: We want to model
\[ P[R_j(S_i, O_k) = 1] \]
(equivalently, approximate a binary tensor \( X \in \{0, 1\}^{n_s \times n_r \times n_o} \))
Our approach:
- Cast the problem as matrix factorizations
- Represent the subjects and objects as vectors in \( \mathbb{R}^{p} \)
  - \( \{S_i\}_{i \in \mathbb{N}} \rightarrow \mathbb{S} \cong \{s_1, \ldots, s_n\} \subset \mathbb{R}^{p} \)
  - \( \{O_k\}_{k \in \mathbb{N}} \rightarrow \mathbb{O} \cong \{o_1, \ldots, o_n\} \subset \mathbb{R}^{p} \)
- Relations are matrices on which subjects and objects operate
  - \( \{R_j\}_{j \in \mathbb{N}} \in \mathbb{R}^{p \times p} \)
A logistic model:
\[ P[R_j(S_i, O_k) = 1] \cong 1 - e^{-\lambda} = \alpha(S_i, R_j, O_k) \]

A multiple order log-odds ratio model
- \( \alpha(s', r, o') \) accounts for 1-, 2- and 3-way interactions
  - For instance: unigrams, bigrams and trigrams in NLP
- For some parameters \( y, y', z, z' \in \mathbb{R}^{p} \), we define
\[ \alpha(s', r, o') = \sum_{j} \left( \theta_{j} y_{j} + \theta_{j} y'_{j} + \theta_{j} z_{j} + \theta_{j} z'_{j} \right) \]

Sharing parameters across relations
Motivation:
- With many relation types (n_r ≫ 1), we might have few data per relation
- Relations can have similarities (e.g., synonyms in NLP)
- Maybe memory expensive to store \( n_r \times p^2 \) elements
- \( \{\Theta_j\}_{j \in [1, d]} \) represent some ‘canonical’ relations
- \( R_i = \sum_k \alpha' \Theta_{k} \)
  - for some sparse \( \alpha' \in \mathbb{R}^{d} \)
  - with \( \Theta_{k} = u' v_{k}^T \) for some \( u, v_{k} \in \mathbb{R}^{p} \)

Optimization
- \( P/N \) is the set of indices of positively/negatively labeled relations
We maximize the following likelihood:
\[ \mathcal{L} \triangleq \prod_{(i,j,k) \in P/N} P[R_j(S_i, O_k) = 1] \cdot \prod_{(i,j,k) \in N} P[R_j(S_i, O_k) = 0] \]
After proper normalization, it leads to the minimization problem:
\[ \min_{\Theta, \alpha'} \log(\mathcal{L}), \text{ with } \alpha' \leq \lambda, \Theta = u' v_{k}^T \]
\[ z = z' \triangleq \mathbf{O} = \mathbf{S} \]
\[ S, o', y, y', z, u, v_{k} \in \{w, w | w | \leq 1\} \]

Related work
- Tensor factorization methods (Tucker, 1966; Harshman et al., 1994)
- With latent and shared/clustered attributes:
  - Collective matrix factorization (Paccararo et al., 2001; Nickel et al., 2011)
  - Non-parametric Bayesian (Kemp et al., 2006; Sutskever et al., 2009; Miller et al., 2009; Zhu, 2012)
  - Markov-Logic networks (Kock et al., 2007)
  - Neural networks (Bordes et al., 2012)

Experiments
(1) Multi-relational benchmarks
- Setting:
  - Datasets: Kinships, UMLS and Nations
  - Dimensions: \( n_s = 100 \) and \( n_o = 50 \)
  - 10 cross-validation for choices of \( (p, d, \lambda) \)
- Goal: Predict relationships

<table>
<thead>
<tr>
<th>Method</th>
<th>AUC (PR)</th>
<th>p@5</th>
<th>p@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collobert et al. (2011)</td>
<td>0.96</td>
<td>0.98</td>
<td>N/A</td>
</tr>
<tr>
<td>SME</td>
<td>0.84</td>
<td>N/A</td>
<td>0.98</td>
</tr>
<tr>
<td>Best WordNet</td>
<td>0.95</td>
<td>0.96</td>
<td>N/A</td>
</tr>
</tbody>
</table>

(2) NLP application
- Setting:
  - Triplets (subject, verb, direct object) extracted from Wikipedia
  - 1,000,000/50,000/250,000 triplets for training/validation/test
  - 30,605 subjects and direct objects/4,547 verbs
- Goal: Predict a verb given the subject and object

<table>
<thead>
<tr>
<th>Method</th>
<th>AUC (PR)</th>
<th>p@5</th>
<th>p@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collobert et al. (2011)</td>
<td>0.96</td>
<td>0.98</td>
<td>N/A</td>
</tr>
<tr>
<td>SME</td>
<td>0.84</td>
<td>N/A</td>
<td>0.98</td>
</tr>
<tr>
<td>Best WordNet</td>
<td>0.95</td>
<td>0.96</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Evaluate latent representations: Lexical similarity classification
- Human annotated dataset from [Yang et al., 2006]
- 130 pairs of verbs labeled with a score in \{0, 1, 2, 3, 4\}
e.g., (divide, split) is labeled 4, while (postpone, show) has a score of 0
- Idea: if \( R_i = R_j \), then the verbs \( j \) and \( f \) should be similar

- The problem is non convex
- Apply stochastic projected gradient descent
- Use mini-batches
- In some applications, the set \( N \) is not given \rightarrow Sampling schemes
- Can be useful to down-weight negative triplets