Fast DTT – A Near Linear Algorithm for Decomposing a Tensor into Factor Tensors

Xiaomin Fang and Rong Pan
Department of Computer Science
Sun Yat-sen University
Introduction -

• Matrix: model second-order relations
  • Item recommendation (user-item)
  • Friend recommendation (friend-friend)
Introduction - Matrix and Tensor

• Tensor: model higher-order relations
  • Personalized tag recommendation (user-item-tag)
  • Personalized search (user-query-URL)
Motivation - Matrix Decomposition

Take item recommendation for example.

Decompose matrix into factor matrices

The relation between user i and item group 1.
Motivation - Example

Take personalized tag recommendation as an example.

Build user-item-tag tensor by making use of history annotation data.

Given user i and item j, recommend tags to user i when annotating item j.
Motivation - Traditional Tensor Decomposition

**CANDECOMP/PARAFAC (CP):**

Decompose tensor into factor matrices

**Tucker Decomposition (TD):**

Decompose tensor into factor matrices
Motivation - Traditional Tensor Decomposition

Traditional tensor decomposition:
Decompose a tensor into factor matrices

What is the meaning???

The relation between that user and item group 1? No!!!
The relation between that user and tag group 1? No!!!
Motivation - DTT: Decompose Tensor into Factor Tensors
Motivation - DTT: Decompose Tensor into Factor Tensors

DTT: Decompose a tensor into factor tensors

The relation between that user, item group 1 and tag group 1

ig, tg: item group, tag group
Model - Tensor Reconstruction

Predict the value of element $a_{ijk}$:

- $i$: user
- $j$: item
- $k$: tag

Input:
- $X_i$: a slice of $X$
- $Y_j$: a slice of $Y$
- $Z_k$: a slice of $Z$
Model - Tensor Reconstruction

Predict the value of element $a_{ijk}$:

- **Input:**
  - $X_{i,:}$, a slice of $X$
  - $Y_{j,:}$, a slice of $Y$

- **Step 1:**
  - Matrix product

- **Output:**
  - $X_{i,:}$, a slice of $X$
  - $Y_{j,:}$, a slice of $Y$

- **Labels:**
  - User $i$
  - Item $j$
  - Tag $k$
Model - Tensor Reconstruction

Predict the value of element $a_{ijk}$:

**input:**

$X_{i,:}, a slice of X$

$Y_{j,:}, a slice of Y$

**step 1:** matrix product

$Z_{k,:}, a slice of Z$

**step 2:** transpose
Model - Tensor Reconstruction

Predict the value of element $a_{ijk}$:

- **Step 1:** Matrix product
- **Step 2:** Transpose
- **Step 3:** Dot product of the matrices

Prediction of the element $a_{ijk}$
The prediction of $a_{ijk}$ is defined as

$$a_{ijk} = \sum_{p=1}^{D_1} \sum_{q=1}^{D_2} \sum_{r=1}^{D_3} x_{iqr} \cdot y_{jrp} \cdot z_{kpq},$$

where $D_1 = \#\text{user groups}$, $D_2 = \#\text{item groups}$, $D_3 = \#\text{tag groups}$.

The order of $X_{i::}$, $Y_{j::}$ and $Z_{k::}$ makes no difference:

$$a_{ijk} = \langle (X_{i::} \cdot Y_{j::})^T, Z_{k::} \rangle$$

$$= \langle (Y_{j::} \cdot Z_{k::})^T, X_{i::} \rangle$$

$$= \langle (Z_{k::} \cdot X_{i::})^T, Y_{j::} \rangle.$$
Model - Limitations of DTT

The time complexity to calculate

\[ a_{ijk} = \sum_{p=1}^{D_1} \sum_{q=1}^{D_2} \sum_{r=1}^{D_3} x_{iqr} \cdot y_{jrp} \cdot z_{kpq} \]

is \( O(D_1 D_2 D_3) \), where \( D_1 = \# \text{user groups} \), \( D_2 = \# \text{item groups} \), \( D_3 = \# \text{tag groups} \).

The space complexity of DTT is \( O(ID_2D_3 + JD_3D_1 + KD_1D_2) \), where \( I = \# \text{users} \), \( J = \# \text{items} \), \( K = \# \text{tags} \).

It takes too much time and contains too many parameters.
Model - Fast DTT

Decompose each slice of factor tensors into two smaller matrices:
Model - Fast DTT

\[ a_{ijk} = \sum_{p=1}^{D_1} \sum_{q=1}^{D_2} \sum_{r=1}^{D_3} x_{iqr} \cdot y_{jrp} \cdot z_{kpq} \]

can be rewritten as:

\[ a_{ijk} = \sum_{p=1}^{D_1} \sum_{q=1}^{D_2} \sum_{r=1}^{D_3} \sum_{u=1}^{d_1} x_{iqr}^{(l)} \cdot x_{iru}^{(r)} \sum_{v=1}^{d_2} y_{jrv}^{(l)} \cdot y_{jpv}^{(r)} \sum_{w=1}^{d_3} z_{kpw}^{(l)} \cdot z_{kqw}^{(r)} \cdot \]

After re-organization of the formula, we can calculate \( a_{ijk} \) in \( O(D_1d_2d_3+D_2d_3d_1+D_3d_1d_2) \),
and generally \( d_1,d_2,d_3 << D_1, D_2, D_3 \).
# Model - Comparison between Fast DTT, TD and CP

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Complexity</th>
<th>Space Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Predict</td>
</tr>
<tr>
<td>Fast DTT</td>
<td>$O(Dd^2NT)$</td>
<td>$O(Dd^2)$</td>
</tr>
<tr>
<td>TD</td>
<td>$O(D^3NT)$</td>
<td>$O(D^3)$</td>
</tr>
<tr>
<td>CP</td>
<td>$O(DNT)$</td>
<td>$O(D)$</td>
</tr>
</tbody>
</table>

N: #triplets in training set  
T: #iterations  
I = J = K, $D_1 = D_2 = D_3 = D$, $d_1 = d_2 = d_3 = d$, $d << D$
Experiments - Experimental Setup

- Conduct experiments on synthetic datasets and real tag annotation datasets.
- Optimize the pair-wise ranking function Bayesian Personalized Ranking (BPR) for personalized tag recommendation.
- Exploit Stochastic Gradient Descent (SGD) to optimize the objective function.
- Use information retrieval metrics: MAP and NDCG.
Experiments - Comparative Methods

- **TD-based models**
  - RTF (Rendle et al, KDD 2009)
  - HOSVD (Lathauwer et al, SIAM journal 2000)

- **CP-based models**
  - PITF (Rendle and Schmidt-Thieme, WSDM 2010)
  - CP (Carroll and Chang, Psychometrika 1970)

- **Popularity-based model**
  - Popularity (ranks the tags based on the frequency that item i and the tags co-occur in the training set)
Experiments - Synthetic Datasets

Goal: evaluate different tensor factorization models’ capacity of expression.

<table>
<thead>
<tr>
<th>Users</th>
<th>Items</th>
<th>Tags</th>
<th>Training triplets</th>
<th>Test triplets</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>5000</td>
<td>1000</td>
</tr>
</tbody>
</table>

Users, items and tags are randomly divided into 10 user groups, 10 item groups, 10 tag groups, respectively.

We assume users in the i-th user group always annotate items in the j-th item group with tags in the g(i,j)-th tag group, where g(i,j) is generated randomly.
Experiment - Performance on Synthetic Datasets

![Graph showing performance metrics for different methods.](image)
Experiments - Real Datasets

- Three real datasets: Delicious (283K triplets), Last.fm (162K triplets) and Movielens (27K triplets)
  - From the 2-nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011)

- Removed infrequent users, items and tags until every user, item and tag occurred in at least 5 triplets
  - Use a core-based approach (Stumme et al, Ai Communications 2008)
# Experiments - Performance on Real Datasets

<table>
<thead>
<tr>
<th>category</th>
<th>algorithm</th>
<th>#feature</th>
<th>Delicious</th>
<th>Last.fm</th>
<th>Movielens</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>D</td>
<td>d</td>
<td>MAP</td>
<td>NDCG@5</td>
<td>MAP</td>
</tr>
<tr>
<td>DTT</td>
<td>Fast DTT</td>
<td>64</td>
<td>1</td>
<td>0.111</td>
<td>0.240</td>
</tr>
<tr>
<td>TD</td>
<td>RTF</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>HOSVD</td>
<td>64</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CP</td>
<td>PITF</td>
<td>64</td>
<td>-</td>
<td>0.106</td>
<td>0.240</td>
</tr>
<tr>
<td></td>
<td>CP</td>
<td>64</td>
<td>-</td>
<td>0.104</td>
<td>0.238</td>
</tr>
<tr>
<td>popularity</td>
<td>popularity</td>
<td>-</td>
<td>-</td>
<td>0.100</td>
<td>0.158</td>
</tr>
</tbody>
</table>
The prediction quality of the models with $d = 1$ is comparable with the models with $d = 2$ on all the real datasets. Thus, our model is near linear.
Conclusion

• DTT: Decompose a tensor into factor tensors rather than factor matrices
  • More natural for the representation of the higher-order relations
  • Has strong capacity of expression

• Fast DTT: Decompose each slice of factor tensors into two smaller matrices
  • Near linear and practical for large scale real problems
Thank you!