使用HAWQ和MADlib开发推荐应用

Hong Wu: hwu@pivotal.io
Ruilong Huo: rhuo@pivotal.io
Meetup @Shanghai, 2016/04/28
Agenda

- RecSys Review
- Recommendation with HAWQ
- Demo
- Q&A
Agenda

- RecSys Review
What is RecSys

- Recommendation
- More than search engine
  - explore the world without query
- Personalization
  - discover items based on preference
- Self-evolving
Industry Trends
System Design

- Content-based recommendation
  - features from item itself
- Collaborative filtering
  - user-user similarity & item-item similarity
- Model-based recommendation
  - factor model
System Design

- Content-based recommendation
  - features from item itself
- Collaborative filtering
  - user-user similarity & item-item similarity
- Model-based recommendation
  - factor model
Agenda

- RecSys Review
- Recommendation with HAWQ
System Overview

**Preparation**
Map original interactive data into user-item-rating tuples. Import into HAWQ Database.

**Latent Factor Model**
Use `lmf_igd_run` function in MADLib to train user factors and item factors.

**Similarity**
Apply CF strategy to compute similarity.

**Recommendation**
Three ways to return (see later slides).

---

**Model Pipeline**

1. **Import Training Data**
2. **Matrix Factorization**
3. **Compute Item-Item Similarities**
4. **Make Recommendation**

![Diagram showing the model pipeline with icons for Hadoop, MADlib, and HAWQ.]

A personalized Web, made for you.
Preparation

- User data
- Item data
- Interactive data
Preparation

- User data
- Item data
- Interactive data
  - \{\text{user, item}\}: preference weight
Latent Factor Model

- **Matrix factorization**
  - model each user-item as a vector of factors

\[ y_{ij} \sim \sum_k u_{ik} v_{jk} = u_i' v_j \]

\[ y_{M \times N} \sim U_{M \times K} V_{K \times N} \]

- factor vector of user \( i \)
- factor vector of item \( j \)

- \( K \ll M, N \)
- \( M = \) number of users
- \( N = \) number of items
Latent Factor Model

Rating Matrix (N x M)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>User Feature Matrix (F x N)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f₁</td>
<td>1</td>
<td>-4</td>
<td>1</td>
</tr>
<tr>
<td>f₂</td>
<td>-2</td>
<td>0</td>
<td>-3</td>
</tr>
<tr>
<td>f₃</td>
<td>0</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

Movie Feature Matrix (F x M)

<table>
<thead>
<tr>
<th>Movie Title</th>
<th>f₁</th>
<th>f₂</th>
<th>f₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drama</td>
<td>-1</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>Train de Vie</td>
<td>4</td>
<td>-4</td>
<td>1</td>
</tr>
<tr>
<td>Comedy</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Matrix Factorization in MADlib

SELECT madlib.lmf_igd_run(output, input, row, col, value, row_dim, col_dim, max_rank, stepsize, iterations, tolerance);

— result table to store model
— input table
— user attribute name
— item attribute name
— preference weight
— the number of users
— the number of items
— the number of factors
— learning rate
— training rounds
Similarity

- We have user factors $U$ and item factors $V$
- Apply collaborative filtering strategy
  - compute item-item similarities
  - less computation, large scale
  - good diversity vs original CF
CREATE TABLE item_sim AS (  
SELECT ifacA.iid,  
    ifacB.iid,  
    madlib.cosine_similarity(ifacA.rating,  
    ifacB.rating) as sim  
FROM ifactor AS ifacA,  
    ifactor AS ifacB  
WHERE ifacA.iid < ifacB.iid  
    sim DESC LIMIT K  
);
Recommendation

- Use item similarity directly

radio FM

book like-like
Recommendation

• Use user factor and item factor directly
  • dot product

<table>
<thead>
<tr>
<th></th>
<th>Troy (Action)</th>
<th>The Godfather (Crime)</th>
<th>Titanic (Romantic)</th>
<th>Forrest Gump (Comedy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fahime</td>
<td>5</td>
<td>∅</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Musi</td>
<td>5</td>
<td>∅</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Hamza</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Paul</td>
<td>4</td>
<td>∅</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Adam</td>
<td>1</td>
<td>2</td>
<td></td>
<td>5</td>
</tr>
</tbody>
</table>
Recommendation

- Use item similarity to predict rating
- weighted sum

<table>
<thead>
<tr>
<th></th>
<th>Troy (Action)</th>
<th>The Godfather (Crime)</th>
<th>Titanic (Romantic)</th>
<th>Forrest Gump (Comedy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fahime</td>
<td>5</td>
<td>⌀</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Musi</td>
<td>5</td>
<td>⌀</td>
<td>⌀</td>
<td>1</td>
</tr>
<tr>
<td>Hamza</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>Paul</td>
<td>4</td>
<td>⌀</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Adam</td>
<td>1</td>
<td>2</td>
<td>⌀</td>
<td>5</td>
</tr>
</tbody>
</table>
Agenda

- RecSys Review
- Recommendation with HAWQ
- Demo
Try it Out

- [https://github.com/xunzhang/xz_talk/tree/master/develop_recsys_with_hawq](https://github.com/xunzhang/xz_talk/tree/master/develop_recsys_with_hawq)
Why Choose HAWQ to Build Machine Learning Application

• Simple & Efficient
  • high level abstraction, no transfer cost
  • focus more on application’s own logic
  • dataflow thinking
• High scalability & performance
  • distributed machine learning with MADlib
  • parallel query engine with HAWQ
Summary

• Brief intro of RecSys
• A hybrid model pipeline based on HAWQ
Agenda

• RecSys Review
• Recommendation with HAWQ
• Demo
• Q&A
Thanks
Questions?
Backup Slide

Solution of CF in large scale: AB_similarity.

Storage disaster of dot product based on factor model: Build Balltree index.