Large-scale Factorization Machines for Recommendation Systems

Erheng Zhong*, Yue Shi#, Nan Liu$, Suju Rajan+

* Yahoo, CA, USA (Now at Baidu Research)
# Yahoo, CA, USA (Now at Facebook)
$ Yahoo, CA, USA (Now at Google)
+ Yahoo, CA, USA (Now at Criteo Labs)

(Presented by Shuguang Han, University of Pittsburgh)

The work was done when authors and the presenter were at Yahoo! Labs.
Background

- Information Overload
- Pull mode - Search (Retrieval)
  - Most effective with well articulated information needs
- Push mode - Recommendation (Discovery)
  - Most useful with unclear information need

Recommend items (e.g., news) to users based on their historical interactions
Challenges in Building Recommendation Systems

● Tackling the high complexity of data
  ○ Item features: taxonomy, tags, etc.
  ○ Explicit/Implicit user feedback: rating, like/unlike, click/skip, etc.
  ○ Context: time, location, etc.
  ○ ......

● Handling different tasks
  ○ User Profiling
  ○ Content Enrichment
  ○ Personalized Ranking
  ○ ......

● Numerous Approaches: ~2M research articles on Google Scholar
Scalable Factorization Machines

- Applicable to a broad range of ML problems at Yahoo
  - Personalization use cases: User-item/Item-Item collaborative filtering, Context-aware collaborative filtering, User profiling
  - Other use cases: Topic modeling for large document corpus, Personalized and query-dependent MLR, ADs CTR prediction

- Each problem solved by an independent implementation
  - Ranking: GBDT, Logistic Regression
  - Content enrichment: topic modeling, clustering and classification
  - Recommendation: latent factor model or neighbor-based model

- Most implementations do not lend itself to scalable solutions
  - As a function of number of parameters in the ML model
  - As a function of amount of data that can be processed by the model

Factorization machine is a general representation of a number of ML models that lends itself to:

- Shared model implementation
- Implicitly modeling interaction features
- Distributed learning
- Without sacrificing performance
What is a Factorization Machine?

- A generic relevance modeling framework using high dimensional sparse features
- Polynomial regression with factorized high-order coefficients
- Each entity is a vector of latent factors, the relevance is modeled by inner-product

![Diagram of Factorization Machine](image)

General formulation

\[
\hat{y}(x) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle x_i x_j
\]

\[
w_{j,j'} = \sum_{f=1}^{k} v_{j,k} v_{j',k}
\]
Data Representation in Factorization Machines

<table>
<thead>
<tr>
<th>Feature vector (x)</th>
<th>Response (y)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>Click</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>Skip</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>Skip</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>Click</td>
</tr>
<tr>
<td>( x_5 )</td>
<td>Click</td>
</tr>
<tr>
<td>( x_6 )</td>
<td>Skip</td>
</tr>
</tbody>
</table>

Indices
- Tom
- Jack
- Alice

User
- Age
- Male
- Female

Response can be binary or float

Indices
- User Features
- Item Features

Side Information
- User
- Item
Scale Up Factorization Machines

- **Challenge:** Big Data + Big Model
  - FM: Terascale models with tens of billions of parameters

- **Solution:** Data Parallelism + Model Parallelism + Mini-batch Algorithm
  - Data Parallelism: (by Map/Reduce)
    - Training data sharded among worker nodes
  - Model Parallelism: (by Parameter Server)
    - Parameters sharded among parameter server nodes
  - Mini-batch Algorithm:
    - Only a small subset of global parameters is needed at any time
    - Reduce collision chance of updating on servers
    - Model parameters are updated slightly in every step, enables relaxing model consistency requirements (Hogwild!, Feng Niu, 2011)
General Distributed Learning Framework

Data Parallelism

- Mappers
  - Partition
  - Learning

- Reducers
  - Partition
  - Learning

Model Parallelism

Server

- Fetch Model
- Model Learning
- Push Model

- Communication Cost
- Collision of Updating on Servers
More Efficient Implementation
Localized Model Parallelism

<table>
<thead>
<tr>
<th></th>
<th>U1</th>
<th>U2</th>
<th>U3</th>
<th>I1</th>
<th>I2</th>
<th>I3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reducer-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reducer-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reducer-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parameter Servers

- I1
- I2
- I3
More Efficient Implementation
Distributed Coordinate Descent

Reducer-1

Reducer-2

Reducer-3

Parameter Servers
Offline Experiments: Public Datasets

- Rating Prediction/Basic Matrix Factorization

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Ratings</th>
<th>Users</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movielens</td>
<td>10M</td>
<td>71,567</td>
<td>10,681</td>
</tr>
<tr>
<td>Netflix</td>
<td>100M</td>
<td>480,189</td>
<td>17,770</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RMSE</th>
<th>MyMediaLite</th>
<th>LibFM</th>
<th>DFM</th>
<th>DFM-SCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Movielens</td>
<td>0.871</td>
<td>0.879</td>
<td>0.889</td>
<td>0.883</td>
</tr>
<tr>
<td>Netflix</td>
<td>0.911</td>
<td>0.909</td>
<td>0.929</td>
<td>0.918</td>
</tr>
</tbody>
</table>
User Profiling: Offline Results

- **Dataset:** 3 Months Data for Training, 1 Week Data for Testing

<table>
<thead>
<tr>
<th>Events</th>
<th>Users</th>
<th>Items (news)</th>
<th>Item Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4B</td>
<td>64M</td>
<td>54M</td>
<td>500K</td>
</tr>
</tbody>
</table>

- **Setup for user profiling experiment**
  - represent each user as a vector
  - represent each item (news) as a vector
  - Ranking news based on similarity of two vectors
  - The ground-truth is user’s true click about each news

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>DFM-SCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean Average Precision</td>
<td>0.62360</td>
<td>0.66278</td>
</tr>
</tbody>
</table>
User Profiling: Online Evaluation on Yahoo Homepage

- Using Distributed Factorization Machines to build user profiles as a replacement for production profiles
User Profiling: Online Evaluation on Yahoo Homepage

- Using Distributed Factorization Machines to **expand** production profiles (combine baseline with newly-learned representations) for mildly engaged users.
User-Item CF: App Recommendation

- Dataset/Metrics: 1 Month Data for Training, 1 Week Data for Testing

<table>
<thead>
<tr>
<th>Usages</th>
<th>Users</th>
<th>Apps</th>
<th>MAP@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6B</td>
<td>451M</td>
<td>4.5M</td>
<td>0.132</td>
</tr>
</tbody>
</table>

- Examples

<table>
<thead>
<tr>
<th>Top-5 frequently opened apps</th>
<th>Top-5 recommendations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Snapchat, 9GAG, Flow, Scribd, Hyper Square</td>
<td>Vine, Lara Croft, YouCam, Makeup, Kamcord</td>
</tr>
<tr>
<td>Mini Ninjas, Dog hotel 3D, Super Stickman Golf 2, Extreme Bike Trip, Funny Riddles</td>
<td>Pixel Gun 3D, Bike Race Android Free, Subway Surfers Android, SWA, Dude Perfect 2</td>
</tr>
</tbody>
</table>
User-Item CF: Scalability on Yahoo Recommends Dataset

- **Scalability**

<table>
<thead>
<tr>
<th>Events</th>
<th>Users</th>
<th>Items</th>
<th>Clients</th>
<th>Servers</th>
<th>#Factors</th>
<th>Model Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4B</td>
<td>562.5M</td>
<td>88.2M</td>
<td>100</td>
<td>25</td>
<td>50</td>
<td>306GB</td>
</tr>
</tbody>
</table>

It takes less than 3 hours to build a converged model with the whole dataset.
Q & A

purlin.zhong@gmail.com
One Example: User-Item Collaborative Filtering

User Factors

Item Factors

Parameter server

User/Item Interactions

SCD

Model Fetch

Model Learning

Update User Model

Push Item Model
Learning of Factorization Machines

- Objective Function
  \[
  \min_{\theta} \sum \ell(y, \hat{y}(x)) + \lambda R(\theta)
  \]

  \(\ell\) represents:

  - Square Loss
    \[
    (y - \hat{y}(x))^2
    \]
    Rating, etc.

  - Logistic Loss
    \[
    y \log (\sigma(\hat{y}(x))) + (1-y) \log (1-\sigma(\hat{y}(x)))
    \]
    Like/Dislike, etc.

- Gradient-based Method

  \[
  \theta \leftarrow \theta + \epsilon \left( \frac{\partial}{\partial \theta} \ell(y, \hat{y}) + \lambda \frac{\partial}{\partial \theta} R(\theta) \right)
  \]