Machine Learning for Large-Scale Data Analysis and Decision Making
80-629-17A

Recommender Systems
— Week #10
Today

- Machine learning for recommender system
- Use case of ML with practical constraints

1. Motivation/Introduction to the field
2. Models
   - Matrix factorization (Probablistic Matrix factorization)
   - Modeling additional information (covariates)
   - Practice with pySpark
3. Evaluation
Introduction

• Recommendation task:

  • Suggest items of interest to users
    • Items: movies, books, articles, humans
    • Users: humans
Is it worth our attention?

- Recommendation is the next search
- Search finds items (given a query)
- Recommendation finds items of interest
Google

stock prices
stock prices
stock price history
stock price calculator
stock prices live
• Claim they’re responsible for 4% of US marriages
Machine learning for recommender systems?

- Several different opportunities for machine learning
  1. Learn good user models
  2. Gather most useful data
  3. Learn methods aware of the final objective
• Imagine
  • The data are user ratings
  • Task: Recommend items the user will like

• How do we set it up as a machine learning problem?
• Imagine
  • The data are user ratings
  • Task: Recommend items the user will like

• How do we set it up as a machine learning problem?

• Task: What do we learn? What do we predict? What is the model?
• Performance measure: How do we evaluate the results?
• Experience: How does our model interact with data?
Framework for recommendation problems

Data
- User preferences
- User/Item features

Representation of user preferences
- Model

Recommendations
- E.g. Top-N recommendations
Framework for recommendation problems

1. Task:
   • What do we predict?
   • Regression, Classification, Ranking

Data

- User preferences
- User/Item features

Representation of user preferences

- Model

Recommendations

- E.g. Top-N recommendations
Recommendations

- Ranking models
  - Have to consider a group of items.
    \[ f : (u, i_1, i_2, \ldots, i_m) \rightarrow (r_1, r_2, \ldots, r_m) \]
    - user u's unseen items
    - rank of each item
  - Computationally more expensive

- Score models
  - For each user:
    1. Predict scores of all unseen items \[ f : (u, i) \rightarrow \mathbb{R} \]
    2. Rank items (show top-K)
Framework for recommendation problems

Data

- User preferences
- User/Item features

Representation of user preferences

- Model

Predict Missing Ratings
Score Prediction as regression/classification

$$\begin{bmatrix}
3 & - & \ldots & 0 \\
- & 0 & \ldots & - \\
\vdots & \vdots & \ddots & \vdots \\
2 & - & \ldots & - \\
\end{bmatrix}_{\text{users} \times \text{items}} \xrightarrow{f} \begin{bmatrix}
3 & 2 & \ldots & 0 \\
1 & 0 & \ldots & 3 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 2 & \ldots & 2 \\
\end{bmatrix}$$
Collaborative filtering (CF)

- Work horse used in many recommender systems
- Assumption:
  - Users with past similar preferences will have similar future preferences
CF- Neighborhood approaches

1. For each user, find other users with similar past preferences

2. Predict that user’s missing preferences as the weighted combination of its neighbors’ preferences

\[
\begin{bmatrix}
3 & - & \cdots & 0 \\
- & 0 & \cdots & - \\
\vdots & \ddots & \ddots & \cdots \\
2 & - & \cdots & - \\
\end{bmatrix}_{\text{users} \times \text{items}}
\]
CF- Neighborhood

1. Find distance between every pair of users (or items)

2. Predict missing scores using a user’s neighbors

\[
\text{Dist}(u, u') = \frac{(S_u^o)^\top S_{u'}^o}{\|S_u^o\| \|S_{u'}^o\|}
\]

\[
\hat{S}_{uj} = \sum_{u'} \text{Dist}(u, u') S_{u'j}^o, \quad \forall u' \text{ that have rated } j
\]
CF- Neighborhood approaches

• Non-parametric approach

• A user is represented by a weighted combination of its neighbors

• New users can change one’s recommendations

• Different distance functions to capture different effects

• Ratings vs. clicks

• Could consider additional information

• Works well empirically

• Building similarity matrix can be slow (offline)

• Not probabilistic
CF - Matrix factorization

Model. \( S_{ui} := \theta_u^T \beta_i \)

Parameters. \( \theta_u, \forall u, \beta_i, \forall i \)

Objective. \( \sum_u \sum_i (S_{ui} - \hat{S}_{ui})^2 \)

- Assumption: the observation matrix is low-rank
- Estimates user and item representations
- \( k \) is a hyperparameter \( k << \min(\|Users\|, \|Items\|) \)
Model fitting

Objective. \[ \sum_u \sum_i (S_{ui} - \hat{S}_{ui})^2 \]
Model fitting

Objective.  \[ \sum_u \sum_i (S_{ui} - \hat{S}_{ui})^2 \]

- Joint parameter optimization
- Gradient descent: \( (\nabla \theta, \nabla \beta) \)
Model fitting

Objective. \[ \sum_u \sum_i (S_{ui} - \hat{S}_{ui})^2 \]

- Joint parameter optimization
- Gradient descent: \((\nabla \theta, \nabla \beta)\)
- Alternative optimization
  1. Fix \(\theta\), update \(\beta\)
  2. Fix \(\beta\), update \(\theta\)
- Each step can be solved using least squares
Model fitting

Objective. \[ \sum_u \sum_i (S_{ui} - \hat{S}_{ui})^2 \]

- Joint parameter optimization
- Gradient descent: \((\nabla \theta, \nabla \beta)\)
- Alternative optimization
  1. Fix \(\theta\), update \(\beta\)
  2. Fix \(\beta\), update \(\theta\)
- Each step can be solved using least squares

For every user/item fit a linear regression model

\[
S_{ui} = \theta_u^T \beta_i = \sum_k \theta_{uk} \beta_{ik} = \sum_k \theta_{u1} \beta_{i1} + \theta_{u2} \beta_{i2} + \ldots + \theta_{up} \beta_{ip}
\]
ALS in Spark

1. Fix $\theta$, update $\beta$

$$\begin{bmatrix}
\theta \\
\beta
\end{bmatrix} \approx \begin{bmatrix}
\theta \\
\beta
\end{bmatrix}$$
ALS in Spark

1. Fix $\theta$, update $\beta$
Demo
User's highly rated movies

E.T. the Extra-Terrestrial (Children's, Drama)
Full Metal Jacket (Action, Drama, War)
Three Colors: Red (Drama)
Breaker Morant (Drama, War)
Shakespeare in Love (Comedy, Romance)
Shadowlands (Drama, Romance)
Rob Roy (Drama, Romance, War)
The Verdict (Drama)
A Little Princess (Children's, Drama)
Leaving Las Vegas (Drama, Romance)

User's weights for 100 components

Expected user weights

Components

Top movies recommended for the user

Casablanca (Drama, Romance, War)
Breakfast at Tiffany's (Drama, Romance)
Amadeus (Drama)
When Harry Met Sally... (Comedy, Romance)
American Beauty (Comedy, Drama)
Fargo (Crime, Drama, Thriller)
The Right Stuff (Drama)
Gandhi (Drama)
Apocalypse Now (Drama, War)
Toy Story (Children's, Comedy, Animation)

[Gopalan et al.’14]
Laurent Charlin — 80-629

6K Users
4K Movies
1M Ratings

Gopalan et al.’14

480K Users
17.7K Movies
100M Ratings

80K Users
260K Sci. articles
100M Ratings

[Goapan et al.'14]
Probabilistic matrix factorization
Probabilistic matrix factorization

\[(S_{ui} - \hat{S}_{ui})^2\]

Minimizing mean squared error is equivalent to maximizing likelihood under Gaussian noise.
Probabilistic matrix factorization

Gaussian Matrix Factorization

[Salakhutdinov et al. '08]

\[(S_{ui} - \hat{S}_{ui})^2\]

Minimizing mean squared error is equivalent to maximizing likelihood under Gaussian noise.

\[S_{ui} \sim \mathcal{N}(\theta_u^T \beta_i, \sigma)\]

[wikipedia]
Probabilistic matrix factorization

Gaussian Matrix Factorization

\[ \theta_u \sim \mathcal{N}(a, b) \]
\[ \beta_i \sim \mathcal{N}(c, d) \]
\[ S_{ui} \sim \mathcal{N}(\theta_u^T \beta_i, \sigma) \]

Minimizing mean squared error is equivalent to maximizing likelihood under Gaussian noise

[Salakhutdinov et al. '08]
Probabilistic matrix factorization

Gaussian Matrix Factorization

[Salakhutdinov et al. '08]

\[ \|eta_i\|_2 \]
\[ \|	heta_u\|_2 \]
\[ (S_{ui} - \hat{S}_{ui})^2 \]

\[ \theta_u \sim \mathcal{N}(a, b) \]
\[ \beta_i \sim \mathcal{N}(c, d) \]
\[ S_{ui} \sim \mathcal{N}(\theta_u^T \beta_i, \sigma) \]

Minimizing mean squared error is equivalent to maximizing likelihood under Gaussian noise
Probabilistic matrix factorization

Gaussian Matrix Factorization

\[
\begin{align*}
\|\beta_i\|_2 & \\
\|\theta_u\|_2 & \\
(S_{ui} - \hat{S}_{ui})^2 & \\
\theta_u & \sim \mathcal{N}(a, b) \\
\beta_i & \sim \mathcal{N}(c, d) \\
S_{ui} & \sim \mathcal{N}(\theta_u^T \beta_i, \sigma)
\end{align*}
\]

Minimizing mean squared error is equivalent to maximizing likelihood under Gaussian noise

Poisson Matrix Factorization

\[
\begin{align*}
\theta_u & \sim \text{Gamma}(a, b) \\
\beta_i & \sim \text{Gamma}(c, d) \\
S_{ui} & \sim \text{Poisson}(\theta_u^T \beta_i)
\end{align*}
\]
Probabilistic matrix factorization

Gaussian Matrix Factorization

\[ \parallel \beta_i \parallel_2 \]
\[ \parallel \theta_u \parallel_2 \]
\[ (S_{ui} - \hat{S}_{ui})^2 \]
\[ \theta_u \sim \mathcal{N}(a, b) \]
\[ \beta_i \sim \mathcal{N}(c, d) \]
\[ S_{ui} \sim \mathcal{N}(\theta_u^T \beta_i, \sigma) \]

Minimizing mean squared error is equivalent to maximizing likelihood under Gaussian noise

Poisson Matrix Factorization

\[ \theta_u \sim \text{Gamma}(a, b) \]
\[ \beta_i \sim \text{Gamma}(c, d) \]
\[ S_{ui} \sim \text{Poisson}(\theta_u^T \beta_i) \]

• Poisson factorization is correct
• Gaussian factorization is incorrect
• In practice MF typically gives better performance than PF

[wikipedia]
[wikipedia]
Extensions

- We have seen models for user ratings/scores
Extensions

- We have seen models for user ratings/scores
- In several domains additional information exists
Extensions

• We have seen models for user ratings/scores

• In several domains additional information exists
  
  • Users:
    
    • demographic information
Extensions

- We have seen models for user ratings/scores
- In several domains additional information exists
  - Users:
    - demographic information
  - Items:
    - content (e.g., movie genre, book text)
Extensions

• We have seen models for user ratings/scores

• In several domains additional information exists

  • Users:
    • demographic information

  • Items:
    • content (e.g., movie genre, book text)

  • Users & items:
    • timestamps, session information
Extensions

- We have seen models for user ratings/scores
- In several domains additional information exists
  - Users:
    - demographic information
  - Items:
    - content (e.g., movie genre, book text)
  - Users & items:
    - timestamps, session information

Models:

1. A specific model for every type of information
   - A. Social network
   - B. Time information
2. Generic models
A. Social network

- Data: user ratings and users’ friends
- Assume:
  1. Friends influence your preferences
  2. Different levels of trusts for different friends

Model. $S_{ui} := \theta_u^T \beta_i$
A. Social network

• Data: user ratings and users’ friends

• Assume:

  1. Friends influence your preferences

  2. Different levels of trusts for different friends

Model. \( S_{ui} := \theta_u^T \beta_i + \sum_{u' \in N(u)} \tau_{un} S_{u'i} \)
A. Social network

- Data: user ratings and users’ friends

- Assume:
  1. Friends influence your preferences
  2. Different levels of trusts for different friends

Model. \( S_{ui} := \theta_u^T \beta_i + \sum_{u' \in N(u)} \tau_{un} S_{u'i} \)

- The rating of \( u' \) on item \( i \)
- How much \( u \) “trusts” \( u' \)
B. Timestamps

- **Data: user ratings over time**

<table>
<thead>
<tr>
<th>Monday</th>
<th>Tuesday</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>[3 \quad 0 \quad \ldots \quad 0]</td>
<td>[-2 \quad \ldots\quad 2]</td>
<td>...</td>
</tr>
<tr>
<td>[-0 \quad \ldots\quad \ldots]</td>
<td>[-2 \quad \ldots\quad 2]</td>
<td>...</td>
</tr>
<tr>
<td>[\vdots \quad \ldots \quad \ldots]</td>
<td>[\vdots \quad \ldots\quad \ldots]</td>
<td>...</td>
</tr>
<tr>
<td>[2 \quad \ldots \quad \ldots]</td>
<td>[\quad \ldots\quad \ldots]</td>
<td>...</td>
</tr>
</tbody>
</table>
B. Timestamps

• Data: user ratings over time

\[
\begin{bmatrix}
3 & - & \cdots & - & 0 \\
- & 0 & \cdots & - \\
\vdots & \vdots & \ddots & \vdots \\
2 & - & \cdots & -
\end{bmatrix}
\quad \begin{bmatrix}
- & 2 & \cdots & - \\
- & - & \cdots & 2 \\
\vdots & \vdots & \ddots & \vdots \\
- & - & \cdots & -
\end{bmatrix}
\quad \cdots
\]

• Assume that user preferences evolve smoothly over time (t = 0, 1, \ldots, T)

\[
\begin{bmatrix}
\theta^0 \\
\beta^0
\end{bmatrix}
\quad \begin{bmatrix}
\theta^1 \\
\beta^1
\end{bmatrix}
\quad \cdots
\quad \begin{bmatrix}
\theta^T \\
\beta^T
\end{bmatrix}
\]
B. Timestamps

• Data: user ratings over time

\[
\begin{bmatrix}
3 & \cdots & 0 \\
0 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
2 & \cdots & 2 \\
\end{bmatrix}
\begin{bmatrix}
-2 & \cdots & -2 \\
-1 & \cdots & -1 \\
\vdots & \ddots & \vdots \\
0 & \cdots & 0 \\
\end{bmatrix}
\begin{bmatrix}
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\end{bmatrix}
\]

• Assume that user preferences evolve smoothly over time ( \( t = 0, 1, \ldots, T \))

\[
\|\theta^t - \theta^{t-1}\| \\
\theta^t \sim \mathcal{N}(\theta^{t-1}, \sigma^t)
\]

\[
\begin{bmatrix}
\theta^0 \\
\beta^0 \\
\theta^1 \\
\beta^1 \\
\vdots \\
\theta^T \\
\beta^T \\
\end{bmatrix}
\]
Factorization Machines (FM)

- Generic framework to model “all” additional information

\[
\begin{bmatrix}
3 & \cdots & 0 \\
- & 0 & \cdots \\
\vdots & \ddots & \vdots \\
2 & \cdots & -
\end{bmatrix}
\]
Factorization Machines (FM)

• Generic framework to model “all” additional information
Factorization Machines (FM)

- Generic framework to model “all” additional information

```
3   -   ...    0
-   0   ...   -
... ...
2   -   ...   -
```

Encode each categorical variable using a series of indicator variables

(User 1, Item 1, 3)
(User 1, Item m, 0)
(User 2 Item 2, 0)
(User n, Item 1, 2)
Factorization Machines (FM)

- Generic framework to model “all” additional information

 Encode each categorical variable using a series of indicator variables

\[
\begin{bmatrix}
3 & - & \cdots & 0 \\
- & 0 & \cdots & - \\
\vdots & \ddots & \ddots & \vdots \\
2 & - & \cdots & - \\
\end{bmatrix}
\]

(User 1, Item 1, 3)
(User 1, Item m, 0)
(User 2 Item 2, 0)
(User n, Item 1, 2)

User \rightarrow Item

\[(1 0 \ldots 0, 1 0 \ldots 0, 3)\]
\[(1 0 \ldots 0, 0 0 \ldots 1, 0)\]
\[(0 1 \ldots 0, 0 1 \ldots 1, 0)\]
\[(0 0 \ldots 1, 1 0 \ldots 0, 2)\]
Factorization Machines (FM)

- Generic framework to model “all” additional information

\[
\begin{bmatrix}
3 & - & \cdots & 0 \\
- & 0 & \cdots & - \\
\vdots & \vdots & \ddots & \vdots \\
2 & - & \cdots & -
\end{bmatrix}
\]

Encode each categorical variable using a series of indicator variables

\[
\begin{align*}
&\text{(User 1, Item 1, 3)} & (1 & 0 & \cdots & 0, 1 & 0 & \cdots & 0, 3) \\
&\text{(User 1, Item m, 0)} & (1 & 0 & \cdots & 0, 0 & 0 & \cdots & 1, 0) \\
&\text{(User 2 Item 2, 0)} & (0 & 1 & \cdots & 0, 0 & 1 & \cdots & 1, 0) \\
&\text{(User n, Item 1, 2)} & (0 & 0 & \cdots & 1, 1 & 0 & \cdots & 0, 2)
\end{align*}
\]

Model. \( S_{ui} := w_0 + \sum_{j=0}^{n} w_i x_i + \sum_{j=0}^{n} \sum_{j'=0}^{n} \theta_j^\top \theta_{j'} x_j x_{j'} \)
Factorization Machines (FM)

- Generic framework to model “all” additional information

\[
\begin{bmatrix}
3 & \cdots & 0 \\
-0 & \cdots & -0 \\
\vdots & \ddots & \vdots \\
2 & \cdots & -0
\end{bmatrix}
\]

Encode each categorical variable using a series of indicator variables:

- (User 1, Item 1, 3)
- (User 1, Item m, 0)
- (User 2 Item 2, 0)
- (User n, Item 1, 2)

Model. \( S_{ui} := w_0 + \sum_{j=0}^{n} w_i x_i + \sum_{j=0}^{n} \sum_{j'=0}^{n} \theta_j^\top \theta_{j'} x_j x_{j'} \)

- Features added to the data (extra columns) are “automatically” used in the model
- Modelling extra information implies adding the feature

\[
\begin{bmatrix}
1 & 0 & \cdots & 0, 3 \\
1 & 0 & \cdots & 0, 1, 0 \\
0 & 1 & \cdots & 1, 0 \\
0 & 0 & \cdots & 1, 0, 2
\end{bmatrix}
\]
Evaluation metrics

• Score prediction

• Mean squared error: \( \frac{1}{|\text{users}|} \sum_u \sum_i |S_{ui} - \hat{S}_{ui}|^2 \)

• Mean absolute error: \( \frac{1}{|\text{users}|} \sum_u \sum_i (S_{ui} - \hat{S}_{ui})^2 \)

• Ranking

• Precision, recall

• Average rank, mean average precision

• NDCG
Precision/Recall

From wikipedia
Precision/Recall

Precision := \frac{TP}{TP + FP}

Recall := \frac{TP}{TP + FN}

From wikipedia
Precision/Recall

Precision := \frac{TP}{TP + FP} \quad \text{Recall} := \frac{TP}{TP + FN}

- Consider only the top items

From wikipedia
Precision/Recall

Precision := \( \frac{TP}{TP + FP} \)  
Recall := \( \frac{TP}{TP + FN} \)

- Consider only the top items

Precision@K := \( \frac{|\{\text{relevant items}\} \cap \{\text{item ranked in top K}\}|}{|K|} \)

Recall@K := \( \frac{|\{\text{relevant items}\} \cap \{\text{item ranked in top K}\}|}{|\{\text{relevant items}\}|} \)
Precision/Recall

Precision := \frac{TP}{TP + FP}  \quad \text{Recall := } \frac{TP}{TP + FN}

- Consider only the top items

\begin{align*}
\text{Precision@K} & := \frac{|\{\text{relevant items}\} \cap \{\text{item ranked in top K}\}|}{|K|} \\
\text{Recall@K} & := \frac{|\{\text{relevant items}\} \cap \{\text{item ranked in top K}\}|}{|\{\text{relevant items}\}|}
\end{align*}

- Other metrics such as NDCG further discount later ranked items
Administrative matters

• Private account on notebooks
  • Required if you want to work outside the University
  • Talk see me (you need to set your password)
  • I have decommissioned the previous notebooks (from the labs), let me know if you want your data (email)

• Poster session

• Printing
  A0 or A1 format (between $25 and $35)

Coop HEC