Recommending Products: Matrix Factorization

Solution 0: Popularity
Simplest approach: Popularity

What are people viewing now?
- Rank by global popularity

Limitation:
- No personalization

Solution 1: Classification model
What’s the probability I’ll buy this product?

Pros:
- Personalized: Considers user info & purchase history
- Features can capture context: Time of the day, what I just saw,…
- Even handles limited user history: Age of user, …

Cons:
- Features may not be available
- Often doesn’t perform as well as collaborative filtering methods (next)

Solution 2: People who bought this also bought…
Co-occurrence matrix

- People who bought diapers also bought baby wipes

- **Matrix C:**
  - store # users who bought both items $i \& j$
  - $(\# \text{ items} \times \# \text{ items})$ matrix
  
  \[ C = \begin{pmatrix} \text{baby wipes} \\ \text{diapers} \end{pmatrix} \begin{pmatrix} \text{baby wipes} \\ \text{diapers} \end{pmatrix} \]

  - *Symmetric:* $\# \text{ purchasing } i \& j$ same as $\# \text{ for } j \& i$ \( (C_{ij} = C_{ji}) \)

(Weighted) Average of purchased items

User \( \bigcirc \) bought items \{diapers, milk\}
- Compute user-specific score for each item $j$ in inventory by combining similarities:

\[
\text{Score}(\bigcirc, \text{baby wipes}) = \frac{1}{2} (S_{\text{baby wipes}, \text{diapers}} + S_{\text{baby wipes}, \text{milk}})
\]

- Could also weight recent purchases more

Sort \( \text{Score}(\bigcirc, j) \) and find item $j$ with highest similarity
Solution 3: Discovering hidden structure by matrix factorization
Movie recommendation

Users watch movies and rate them

<table>
<thead>
<tr>
<th>User</th>
<th>Movie</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each user only watches a few of the available movies

Matrix completion problem

Rating = Movies

Users

Data: Users score some movies

Rating(u,v) known for black cells
Rating(u,v) unknown for white cells

Goal: Filling missing data?
Suppose we had $d$ topics for each user & movie

- Describe movie $v$ with topics $R_v$
  - How much is it action, romance, drama, ...

- Describe user $u$ with topics $L_u$
  - How much she likes action, romance, drama, ...

- $\text{Rating}(u,v)$ is the product of the two vectors

- **Recommendations**: sort movies user hasn’t watched by $\text{Rating}(u,v)$
Predictions in matrix form

\[ \text{Rating} = L \times R' \approx X_{ij} \text{ known for black cells} \]
\[ X_{ij} \text{ unknown for white cells} \]

Rows index movies

Columns index users

\[ X = \text{Rating} \]

But we don’t know topics of users and movies...

Matrix factorization model: Discovering topics from data

\[ \text{Rating} = L \times R' \approx X_{ij} \text{ known for black cells} \]
\[ X_{ij} \text{ unknown for white cells} \]

Rows index movies

Columns index users

\[ X = \text{Rating} \]

• Only use observed values to estimate “topic” vectors \( \hat{L}_u \) and \( \hat{R}_v \)
• Use estimated \( \hat{L}_u \) and \( \hat{R}_v \) for recommendations

Many efficient algorithms for factorization
Is the problem well posed?

Can we uniquely identify the latent factors?

If \( r_{uv} \) is described by \( L_u, R_v \), what happens if we redefine the “topics” as

Then,

Other (orthonormal) transformations can have the same effect.
Matrix factorization objective

- Minimize mean squared error:
  - (Other loss functions are possible)

- Non-convex objective
Coordinate descent

Goal: Minimize some function $g$

Often, hard to find minimum for all coordinates, but easy for each coordinate

Coordinate descent:
- Initialize $\hat{w} = 0$ (or smartly...)
- while not converged
  - pick a coordinate $j$
  - $\hat{w}_j \leftarrow$

Comments on coordinate descent

How do we pick next coordinate?
- At random ("random" or "stochastic" coordinate descent), round robin, ...

No stepsize to choose!

Super useful approach for many problems
- Converges to optimum in some cases (e.g., "strongly convex")
Coordinate descent for matrix factorization

\[
\min_{L,R} \sum_{(u,v): r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2
\]

- Fix movie factors \( R_v \), optimize for user factors \( L_u \)
- First key insight:

Minimize objective separately for each user

- For each user \( u \):
  \[
  \min_{L_u} \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2
  \]
- Second key insight:
Overall coordinate descent algorithm

\[
\min_{L,R} \sum_{(u,v): r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2
\]

- Fix movie factors, optimize for user factors
  - Independent least-squares over users
    \[
    \min_{L_u} \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2
    \]
- Fix user factors, optimize for movie factors
  - Independent least-squares over movies
    \[
    \min_{R_v} \sum_{u \in U_v} (L_u \cdot R_v - r_{uv})^2
    \]

- System may be underdetermined:
- Converges to
- Choices of regularizers and impact on algorithm:

Training Data → Feature extraction → ML model → \( \hat{y} \) → ML algorithm → Quality metric
Using the results of matrix factorization

- Discover “topics” $R_v$ for each movie $v$
- Discover “topics” $L_u$ for each user $u$
- $Score(u,v)$ is the product of the two vectors $\Rightarrow$ Predict how much a user will like a movie

- Recommendations: sort movies user hasn’t watched by $Score(u,v)$

Example topics discovered from Wikipedia

Application to text data:
Bringing it all together:  
Featurized matrix factorization

Limitations of matrix factorization

• Cold-start problem
  – This model still cannot handle a new user or movie

\[
\text{Rating} = \begin{array}{c}
\includegraphics[width=0.5\textwidth]{rating.png}
\end{array}
\]
Cold-start problem more formally

Consider a new user \( u' \) and predicting that user’s ratings
- No previous observations
- Objective considered so far:

\[
\min_{L,R} \frac{1}{2} \sum_{u,v} (L_u \cdot R_v - r_{uv})^2 + \frac{\lambda_u}{2} ||L||_F^2 + \frac{\lambda_v}{2} ||R||_F^2.
\]

- Optimal user factor:
- Predicted user ratings:

Combining features and discovered topics

- Features capture context
  - Time of day, what I just saw, user info, past purchases, ...
- Discovered topics from matrix factorization capture groups of users who behave similarly
  - Women from Seattle who teach and have a baby

- Combine to mitigate cold-start problem
  - Ratings for a new user from features only
  - As more information about user is discovered, matrix factorization topics become more relevant
Collaborative filtering with specified features

- Create feature vector for each movie (often have this even for new movies):

- Define weights on these features for how much all users like each feature

- Fit linear model:

- Minimize:

Building in personalization

- Of course, users do not have identical preferences
- Include a user-specific deviation from the global set of user weights:

- If we don’t have any observations about a user, use wisdom of the crowd

- As we gain more information about the user, forget the crowd

- Can add in user-specific features, and cross-features, too
Featurized matrix factorization—
A combined approach

Feature-based approach:
- Feature representation of user and movies fixed
- Can address cold-start problem

Matrix factorization approach:
- Suffers from cold-start problem
- User & movie features are learned from data

A unified model:

Blending models

- Squeezing last bit of accuracy by blending models
- Netflix Prize 2006-2009
  - 100M ratings
  - 17,770 movies
  - 480,189 users
  - Predict 3 million ratings to highest accuracy
  - Winning team blended over 100 models
A performance metric for recommender systems

The world of all baby products
Why not use classification accuracy?

- Classification accuracy = \text{fraction of items correctly classified (liked vs. not liked)}

- Here, not interested in what a person does not like

- Rather, how quickly can we discover the relatively few liked items?
  - (Partially) an imbalanced class problem
How many liked items were recommended?

Recall

# liked & shown  
# liked

How many recommended items were liked?

Precision

# liked & shown  
# shown
Maximize recall:
Recommend everything

Recall
# liked & shown
# liked

Resulting precision?

Precision
# liked & shown
# shown
Optimal recommender

Precision-recall curve

- **Input**: A specific recommender system
- **Output**: Algorithm-specific precision-recall curve

- To draw curve, vary threshold on # items recommended
  - For each setting, calculate the precision and recall
Which Algorithm is Best?

- For a given precision, want recall as large as possible (or vice versa)
- One metric: largest area under the curve (AUC)
- Another: set desired recall and maximize precision (precision at k)

Summary of recommender systems
What you can do now...

• Describe the goal of a recommender system
• Provide examples of applications where recommender systems are useful
• Implement a co-occurrence based recommender system
• Describe the input (observations, number of “topics”) and output (“topic” vectors, predicted values) of a matrix factorization model
• Implement a coordinate descent algorithm for optimizing the matrix factorization objective presented
• Exploit estimated “topic” vectors to make recommendations
• Describe the cold-start problem and ways to handle it (e.g., incorporating features)
• Analyze performance of various recommender systems in terms of precision and recall
• Use AUC or precision-at-k to select amongst candidate algorithms