Recommended Products: Matrix Factorization
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 $i$ & $j$
  - (# items x # items) matrix
    - Symmetric: # purchasing $i$ & $j$ same as # for $j$ & $i$ ($C_{ij} = C_{ji}$)

(Weighted) Average of purchased items

User bought items \{diapers, milk\}
- Compute user-specific score for each item $j$ in inventory by combining similarities:
  $$\text{Score}(\text{user}, \text{baby wipes}) = \frac{1}{2} (S_{\text{baby wipes, diapers}} + S_{\text{baby wipes, milk}})$$
- Could also weight recent purchases more

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

Diagram:
- Training Data → Feature extraction
- Feature extraction → ML model
- ML model → Quality metric
- Quality metric → ML algorithm
- ML algorithm → ŵ
- ŵ → ML model
- ML model → ŷ
- ŷ → ŷ
- ŷ → ŷ
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>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each user only watches a few of the available movies

Matrix completion problem

Data: Users score some movies

\[
\text{Rating}(u,v) \text{ known for black cells} \\
\text{Rating}(u,v) \text{ unknown for white cells}
\]

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, ...
  
  \[
  R_v = \begin{bmatrix} 0.3 & 0.1 & 1.5 & \cdots \end{bmatrix}
  \]

- Describe user $u$ with topics $L_u$
  - How much she likes action, romance, drama, ...
  
  \[
  L_u = \begin{bmatrix} 2.5 & 0 & 0.8 & \cdots \end{bmatrix}
  \]

- $\text{Rating}(u, v)$ is the product of the two vectors
  \[
  \begin{align*}
  R_v &= \begin{bmatrix} 0.3 & 0.1 & 1.5 & \cdots \end{bmatrix} \\
  L_u &= \begin{bmatrix} 2.5 & 0 & 0.8 & \cdots \end{bmatrix} \\
  L_u' &= \begin{bmatrix} 0 & 3.5 & 0.01 & \cdots \end{bmatrix}
  \end{align*}
  \]
  \[
  \text{Rating}(u, v) = 0.3 \times 2.5 + 0 + 1.5 \times 0.8 + \cdots 
  \approx 7.2
  \]
  \[
  \text{Rating}(u, v) = 0 + 0.01 + 3.5 \times 1.5 + 0.01 + \cdots 
  \approx 0.8
  \]

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

Matrix factorization model: Discovering topics from data

- 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

$$\hat{L}_u = cL_u \quad \hat{R}_v = \frac{1}{c} R_v$$

Then,

$$\hat{L}_u \cdot \hat{R}_v = cL_u \cdot \frac{1}{c} R_v = c \frac{1}{c} (L_u \cdot R_v) = L_u \cdot R_v = r_{uv}$$

Other (orthonormal) transformations can have the same effect.

(Other trans. have same effect... orthonormal trans.)

(can’t uniquely identify $L_u$, $R_v$) don’t interpret individually, only product

---

Training Data

Feature extraction

ML model

ML algorithm

Quality metric

$x, \hat{y}$

$y, \hat{w}$
Matrix factorization objective

Rating = X

- Minimize mean squared error:
  - (Other loss functions are possible)
    \[
    \min_{L,R} \sum_{u,v,i} \left( L_{uv} R'_{uv} - r_{uv} \right)^2
    \]
    
  - Non-convex objective subject to convergence to local mode
Coordinate descent

Goal: Minimize some function $g$

$$g(w) = g(w_0, w_1, \ldots, w_D)$$

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 $j$  
$$\hat{w}_j \leftarrow \min_{w_j} g(w_0, \ldots, \hat{w}_j, \ldots, w_D)$$

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:

\[
\min_{L_u \ldots L_n} \sum_u \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2
\]

\[
= \sum_u \min_{L_u} \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2 
\]

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: Looks like linear regression!

\[
\min_{w} \sum_{i=1}^{N} (w \cdot h(x_i) - y_i)^2
\]

Opt. w/ grad. desc.
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
  \[
  \min_{L_u} \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2 + \lambda_u \|L\|
  \]

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

- System may be underdetermined: use regularization
- Converges to local optima
- Choices of regularizers and impact on algorithm:
  - \(L_2: \|L\|_2^2 \rightarrow \text{ridge}\)
  - \(L_1: \|L\|_1 \rightarrow \text{lasso}\)
Using the results of matrix factorization

- Discover “topics” $R_v$ for each movie $v$
- Discover “topics” $L_u$ for each user $u$

$\text{Score}(u,v)$ is the product of the two vectors $R_v$ and $L_u$

Predict how much a user will like a movie

Recommend movies $v$ and $w$ with the highest score $\text{Score}(u,v)$

Example topics discovered from Wikipedia
Bringing it all together:
Featurized matrix factorization

Limitations of matrix factorization

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

Rating =

\[ X_{ij} \text{ known for black cells} \]
\[ X_{ij} \text{ unknown for white cells} \]
Cold-start problem more formally

Consider a new user $u'$ and predicting that user's ratings
- No previous observations
- Objective considered so far:
  $$\min_{L,R} \frac{1}{2} \sum_{(u,v) \in \mathcal{T}} (L_u \cdot R_v - r_{uv})^2 + \frac{\lambda_u}{2} ||L||_F^2 + \frac{\lambda_v}{2} ||R||_F^2$$
- Optimal user factor:
  $$L_{u'} = 0$$
- Predicted user ratings:
  $$\text{always predict: } r_{uv} = 0 \quad \forall v$$

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):
  \[ \phi(v) = \langle \text{genre, year, director}, \ldots \rangle \]

- Define weights on these features for how much all users like each feature
  \[ w \in \text{vector of same length} \]

- Fit linear model:
  \[ r_{uv} \approx w \cdot \phi(u) \]

- Minimize:
  \[ \min_w \sum_{u} (w \cdot \phi(u) - r_{uv})^2 + \lambda ||w||^2 \]

Building in personalization

- Of course, users do not have identical preferences

- Include a user-specific deviation from the global set of user weights:
  \[ r_{uv} = (w + w_u) \cdot \phi(u) \]

- If we don’t have any observations about a user, use wisdom of the crowd
  \[ \text{Initialize } w_u = 0 \Rightarrow r_{uv} \approx w \cdot \phi(u) \]

- As we gain more information about the user, forget the crowd
  \[ w_u \text{ more informed (personalization)} \]

- Can add in user-specific features, and cross-features, too
  \[ \phi(u) = \langle \text{age, gender, education} \rangle \]
  \[ \phi(u,v) = \langle \ldots \phi(u) \ldots, \ldots \phi(v) \ldots \rangle \]
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:
\[
\hat{r}_{uv} = \mathbf{u}_v \mathbf{R} \mathbf{v} + (\mathbf{u}_v \mathbf{w}_u) \phi(u,v)
\]

Solve via coord. desc., grad. desc., etc.

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