ColumbiaX: Machine Learning
Lecture 17

Prof. John Paisley

Department of Electrical Engineering
& Data Science Institute

Columbia University

COLLABORATIVE FILTERING

Object recommendation

Matching consumers to products is an important practical problem.

We can often make these connections using user feedback about subsets of products. To give some prominent examples:

- Netflix lets users to rate movies
- Amazon lets users to rate products and write reviews about them
- Yelp lets users to rate businesses, write reviews, upload pictures
- YouTube lets users like/dislike a videos and write comments

Recommendation systems use this information to help recommend new things to customers that they may like.

Content filtering

One strategy for object recommendation is:

Content filtering: Use known information about the products and users to make recommendations. Create profiles based on

- Products: movie information, price information, product descriptions
- Users: demographic information, questionnaire information

Example: A fairly well known example is the online radio Pandora, which uses the "Music Genome Project."

- An expert scores a song based on hundreds of characteristics
- A user also provides information about his/her music preferences
- Recommendations are made based on pairing these two sources

Collaborative filtering

Content filtering requires a lot of information that can be difficult and expensive to collect. Another strategy for object recommendation is:

Collaborative filtering (CF): Use previous users' input/behavior to make future recommendations. Ignore any *a priori* user or object information.

- CF uses the ratings of similar users to predict my rating.
- CF is a domain-free approach. It doesn't need to know what is being rated, just who rated what, and what the rating was.

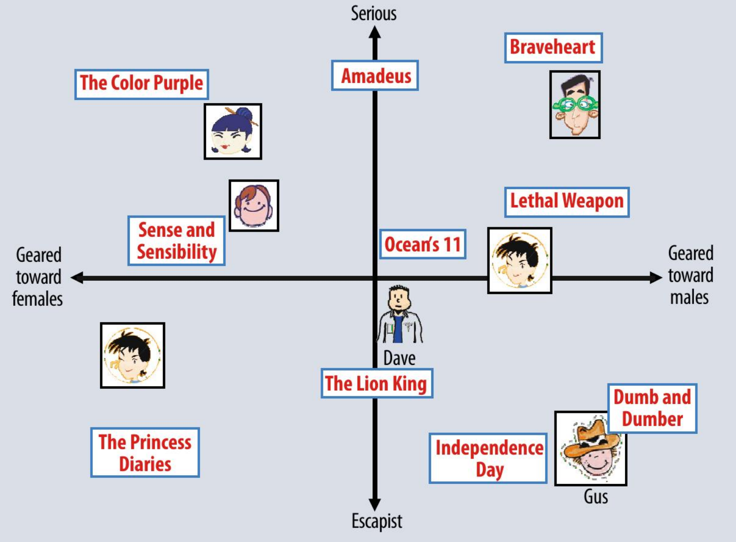
One CF method uses a *neighborhood-based* approach. For example,

1. define a similarity score between me and other users based on how much our overlapping ratings agree, then
2. based on these scores, let others "vote" on what I would like.

These filtering approaches are not mutually exclusive. Content information can be built into a collaborative filtering system to improve performance.

LOCATION-BASED CF METHODS (INTUITION)

Location-based approaches embed users and objects into points in $\mathbb{R}^d$.

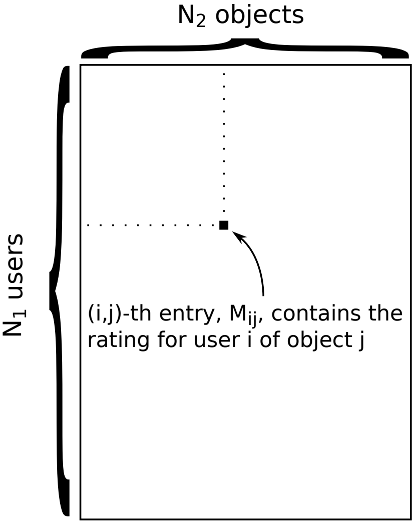


---

1 Koren, Y., Robert B., and Volinsky, C.. "Matrix factorization techniques for recommender systems." Computer 42.8 (2009): 30-37.

Matrix factorization

MATRIX FACTORIZATION



Matrix factorization (MF) gives a way to learn user and object locations.

First, form the rating matrix M:

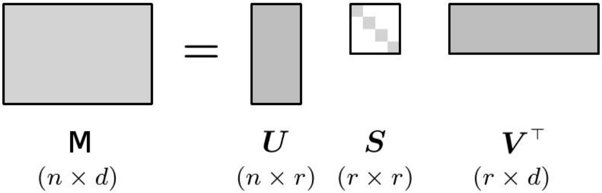
- Contains every user/object pair.
- Will have many missing values.
- The goal is to fill in these missing values.

MF and recommendation systems:

- We have prediction of every missing rating for user i.
- Recommend the highly rated objects among the predictions.

SINGULAR VALUE DECOMPOSITION

Our goal is to factorize the matrix M. We've discussed one method already.

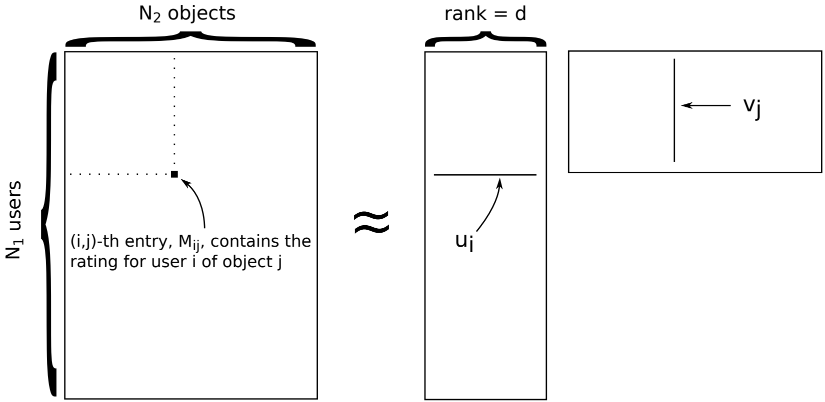


Singular value decomposition: Every matrix M can be written as $M = USV^T$, where $U^TU = I$, $V^TV = I$ and S is diagonal with $S_{ii} \geq 0$.

$r = \text{rank}(M)$. When it's small, M has fewer "degrees of freedom."

Collaborative filtering with matrix factorization is intuitively similar.

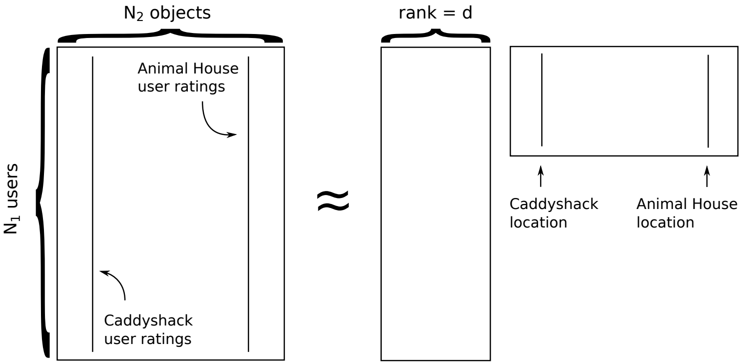
MATRIX FACTORIZATION



We will define a model for learning a low-rank factorization of M. It should:

1. Account for the fact that most values in M are missing
2. Be low-rank, where $d \ll \min\{N_1, N_2\}$ (e.g., $d \approx 10$)
3. Learn a location $u_i \in \mathbb{R}^d$ for user i and $v_j \in \mathbb{R}^d$ for object j

LOW-RANK MATRIX FACTORIZATION

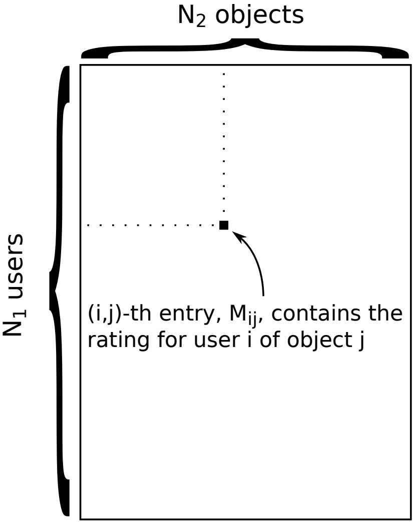


Why learn a low-rank matrix?

- We think that many columns should look similar. For example, movies like *Caddyshack* and *Animal House* should have **correlated** ratings.
- Low-rank means that the N_1-dimensional columns don't "fill up" $\mathbb{R}^{N_1}$.
- Since $> 95\%$ of values may be missing, a low-rank restriction gives hope for filling in missing data because it models correlations.

PROBABILISTIC MATRIX FACTORIZATION

SOME NOTATION



- Let the set Ω contain the pairs (i, j) that are observed. In other words,

$$\Omega = \{(i, j) : M_{ij} \text{ is measured}\}.$$

  So $(i, j) \in \Omega$ if user i rated object j.

- Let Ω_{u_i} be the index set of objects rated by user i.

- Let Ω_{v_j} be the index set of users who rated object j.

Generative model

For N_1 users and N_2 objects, generate

$$\textbf{User locations:} \quad u_i \sim N(0, \lambda^{-1}I), \quad i = 1, \dots, N_1$$

$$\textbf{Object locations:} \quad v_j \sim N(0, \lambda^{-1}I), \quad j = 1, \dots, N_2$$

Given these locations the distribution on the data is

$$M_{ij} \sim N(u_i^T v_j, \sigma^2), \quad \text{for each } (i,j) \in \Omega\,.$$

Comments:

- Since M_{ij} is a rating, the Gaussian assumption is clearly wrong.
- However, the Gaussian is a convenient assumption. The algorithm will be easy to implement, and the model works well.

Model inference

Q: There are many missing values in the matrix M. Do we need some sort of EM algorithm to learn all the u's and v's?

- Let M_o be the part of M that is observed and M_m the missing part. Then

$$p(M_o|U,V) = \int p(M_o, M_m|U,V)dM_m.$$

- Recall that EM is a ***tool*** for maximizing $p(M_o|U,V)$ over U and V.
- Therefore, it is only needed when
  1. $p(M_o|U,V)$ is hard to maximize,
  2. $p(M_o, M_m|U,V)$ is easy to work with, and
  3. the posterior $p(M_m|M_o,U,V)$ is known.

A: If $p(M_o|U,V)$ doesn't present any problems for inference, then no.

(Similar conclusion in our MAP scenario, maximizing $p(M_o,U,V)$.)

Model inference

To test how hard it is to maximize $p(M_o, U, V)$ over U and V, we have to

1. Write out the joint likelihood
2. Take its natural logarithm
3. Take derivatives with respect to u_i and v_j and see if we can solve

The joint likelihood of $p(M_o, U, V)$ can be factorized as follows:

$$p(M_o, U, V) = \underbrace{\left[\prod_{(i,j)\in\Omega} p(M_{ij}|u_i, v_j)\right]}_{\text{conditionally independent likelihood}} \times \underbrace{\left[\prod_{i=1}^{N_1} p(u_i)\right]\left[\prod_{j=1}^{N_2} p(v_j)\right]}_{\text{independent priors}}.$$

By definition of the model, we can write out each of these distributions.

Log joint likelihood and MAP

The MAP solution for U and V is the maximum of the log joint likelihood

$$U_{\text{MAP}}, V_{\text{MAP}} = \arg\max_{U,V} \sum_{(i,j)\in\Omega} \ln p(M_{ij}|u_i, v_j) + \sum_{i=1}^{N_1} \ln p(u_i) + \sum_{j=1}^{N_2} \ln p(v_j)$$

Calling the MAP objective function $\mathcal{L}$, we want to maximize

$$\mathcal{L} = -\sum_{(i,j)\in\Omega} \frac{1}{2\sigma^2}\|M_{ij} - u_i^T v_j\|^2 - \sum_{i=1}^{N_1} \frac{\lambda}{2}\|u_i\|^2 - \sum_{j=1}^{N_2} \frac{\lambda}{2}\|v_j\|^2 + \text{constant}$$

The squared terms appear because all distributions are Gaussian.

MAXIMUM A POSTERIORI

To update each u_i and v_j, we take the derivative of $\mathcal{L}$ and set to zero.

$$\begin{aligned}
\nabla_{u_i}\mathcal{L} &= \sum_{j\in\Omega_{u_i}} \frac{1}{\sigma^2}(M_{ij} - u_i^T v_j)v_j - \lambda u_i = 0 \\
\nabla_{v_j}\mathcal{L} &= \sum_{i\in\Omega_{v_j}} \frac{1}{\sigma^2}(M_{ij} - v_j^T u_i)u_i - \lambda v_i = 0
\end{aligned}$$

We can solve for each u_i and v_j individually (therefore EM isn't required),

$$\begin{aligned}
u_i &= \Big(\lambda\sigma^2 I + \textstyle\sum_{j\in\Omega_{u_i}} v_j v_j^T\Big)^{-1}\Big(\textstyle\sum_{j\in\Omega_{u_i}} M_{ij} v_j\Big) \\
v_j &= \Big(\lambda\sigma^2 I + \textstyle\sum_{i\in\Omega_{v_j}} u_i u_i^T\Big)^{-1}\Big(\textstyle\sum_{i\in\Omega_{v_j}} M_{ij} u_i\Big)
\end{aligned}$$

However, we can't solve for all u_i and v_j at once to find the MAP solution. Thus, as with K-means and the GMM, we use a coordinate ascent algorithm.

MAP inference coordinate ascent algorithm

Input: An incomplete ratings matrix M, as indexed by the set Ω. Rank d.

Output: N_1 user locations, $u_i \in \mathbb{R}^d$, and N_2 object locations, $v_j \in \mathbb{R}^d$.

Initialize each v_j. For example, generate $v_j \sim N(0, \lambda^{-1}I)$.

for each iteration **do**

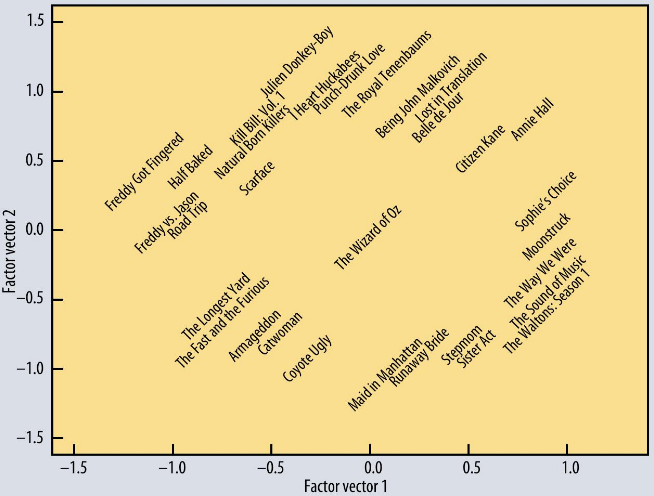
- **for** $i = 1, \ldots, N_1$ **update user location**

$$u_i = \left(\lambda\sigma^2 I + \textstyle\sum_{j\in\Omega_{u_i}} v_j v_j^T\right)^{-1}\left(\textstyle\sum_{j\in\Omega_{u_i}} M_{ij} v_j\right)$$

- **for** $j = 1, \ldots, N_2$ **update object location**

$$v_j = \left(\lambda\sigma^2 I + \textstyle\sum_{i\in\Omega_{v_j}} u_i u_i^T\right)^{-1}\left(\textstyle\sum_{i\in\Omega_{v_j}} M_{ij} u_i\right)$$

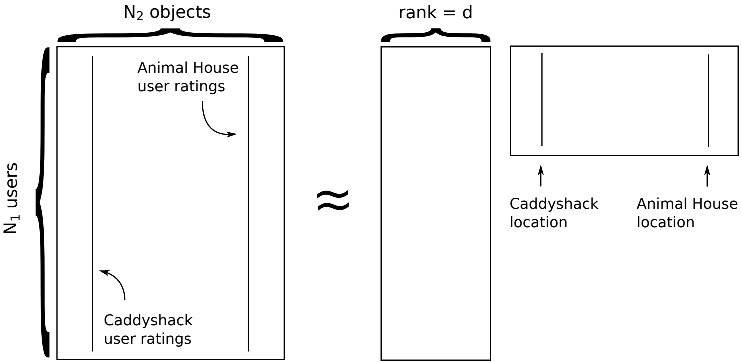
Predict that user i rates object j as $u_i^T v_j$ rounded to closest rating option



Hard to show in $\mathbb{R}^2$, but we get locations for movies and users. Their relative locations captures relationships (that can be hard to explicitly decipher).

[1] Koren, Y., Robert B., and Volinsky, C.. "Matrix factorization techniques for recommender systems." Computer 42.8 (2009): 30-37.

Algorithm output for movies

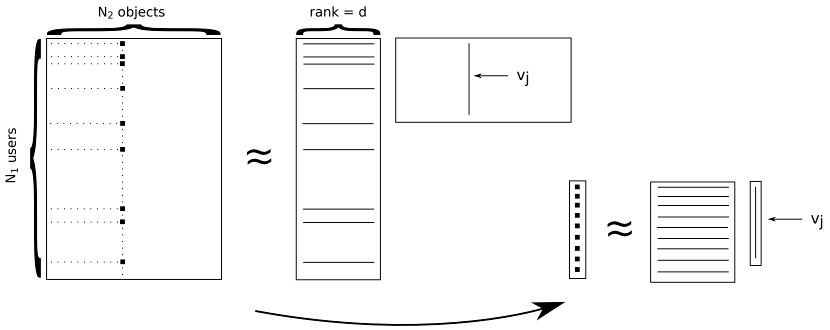


Returning to *Animal House* (j) and *Caddyshack* (j'), it's easy to understand the relationship between their locations v_j and $v_{j'}$:

- For these two movies to have similar rating patterns, their respective v's must be similar (i.e., close to each other in $\mathbb{R}^d$).
- The same holds for users who have similar tastes across movies.

MATRIX FACTORIZATION AND RIDGE REGRESSION

MATRIX FACTORIZATION AND RIDGE REGRESSION



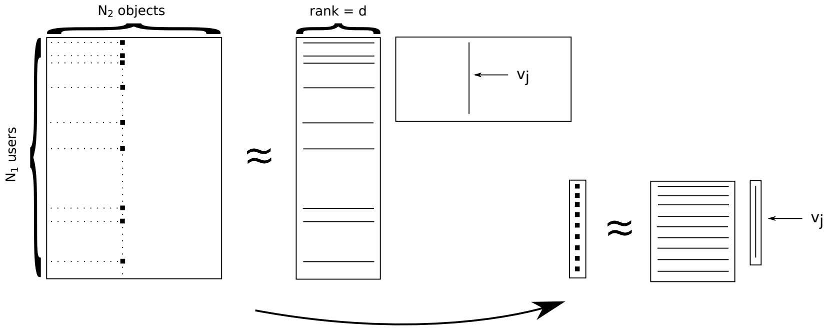
There is a close relationship between this algorithm and ridge regression.

- Think from the perspective of object location v_j.
- Minimize the sum squared error $\frac{1}{\sigma^2}(M_{ij} - u_i^T v_j)^2$ with penalty $\lambda \|v_j\|^2$.
- This is ridge regression for v_j, as the update also shows:

$$v_j = \left(\lambda\sigma^2 I + \sum_{i \in \Omega_{v_j}} u_i u_i^T\right)^{-1} \left(\sum_{i \in \Omega_{v_j}} M_{ij} u_i\right)$$

- So this model is a set of $N_1 + N_2$ coupled ridge regression problems.

Matrix factorization and least squares



We can also connect it to least squares.

- Remove the Gaussian priors on u_i and v_j. The update for, e.g., v_j is then

$$v_j = \Big(\sum_{i \in \Omega_{v_j}} u_i u_i^T\Big)^{-1} \Big(\sum_{i \in \Omega_{v_j}} M_{ij} u_i\Big)$$

- This is the least squares solution. It requires that every user has rated at least d objects and every object is rated by at least d users.
- This probably isn't the case, so we see why a prior is *necessary* here.