Recommendation System

Hung Le

University of Victoria

March 6, 2019

Recommendation System

A system capable of predicting users' responses given options. Two representative examples:

- News recommendation: offering news articles to users given their interests.
- Online shopping: suggest products to users to buy given their purchase history. Amazon is a typical example.
- Movie recommendation: suggest movies to users given their watching history and genre of interests. Netflix is a typical example.

Two Approaches

- Content based approach: classify items into "topics" based on their content, then recommend to users based on their topics of preference.
- Collaborative filtering based approach: examine the similarity measure between users and/or items. Items recommended to an user are preferred by other similar users.

Recommendation systems in the real world combine both approaches to produce the best result.

An example

Amazon Recommendation System in a 2003 paper[1] (cited more than 5000 times according to Google scholar).

AmazonAlg(a customer C who bought item I_1)
    foreach item I_2
        Compute the similarity between I_1 and I_2
    Recommend most similar items of I_1 to C.

Figure: Amazon recommedation system

[1] https://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf

Utility Matrix

Utility matrix is a (spare) matrix where each cell is one of two types:

- Known cells: the value of each cell is the preference of each user (row) given to the item (column).
- Unknown cells: we do not know about the preference of the user given to the item.

The goal of the recommendation system is to predict the values of unknown cells.

Utility Matrix - An example

HP stands for *Harry Potter*, TW stands for *Twilight*, and SW stands for *Star Wars*.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C				2	4	5	
D		3					3

Figure: A utility matrix representing ratings of movies on a 1–5 scale

We would like to know:

- Would user A like SW2? The data supports that A may not like SW2.

Long Tail Phenomenon

Illustrate the difference between recommendation in physical world and online world.

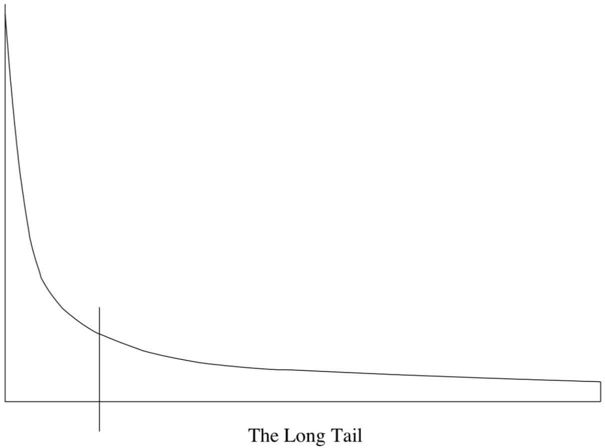

Figure: y-axis represents the popularity of items

Long Tail Phenomenon - An example[2]

Touching the Void is a book (about mountain climbing) published in 1988, which were soon forgotten after its publication. Another book named *Into Thin Air* published a decade later also about mountain climbing. Amazon recommendation system saw few people buy the two books together and started recommend *Touching the Void* to users. *Touching the Void* then outsold *Into Thin Air* more than two to one.

[2] `https://www.wired.com/2004/10/tail/`

Constructing Utility Matrix

- Ask users to give ratings. However, most people are unwilling to provide ratings and the obtained ratings biased toward the ones who provided.
- Infer preferences from users' behavior. Typically the rating has only one value: 1 means the user likes the item.

Content Based Recommendation

The first step is to build a *profile* for each item, which is the set of features of the item. For example a movie has:

- The set of actors of the movie.
- The director.
- The published year.
- The genre.

Content Based Recommendation

The first step is to build a *profile* for each item, which is the set of features of the item. For example a movie has:

- The set of actors of the movie.
- The director.
- The published year.
- The genre.

There are types of items whose features are not clear: *documents* and *images*.

Documents and Images

- Each document can be represented by the set of words that characterize the topic of the document.
- Image: each pixel contains a very little information about the whole image. We can represent an image by its *tags*, a set of words that describe the image.

Representing Item Profiles

We typically represent each item by a vector.

- For discrete values like actors of the movie, or words of documents, create a component for each actor/word and then put 1 if the actor/word appears in the movie/document and 0 otherwise.
- For continuous values, like average rating, can be represented by a single component containing the value.

If the vector contains both 0/1 values and continuous values, we may want to scale components when calculating similarity measure.

User Profiles

Create vectors with the same components that describe users' preferences.

- What we have: the utility matrix representing the connection between users and items. Values of the known cells could be all 1 or arbitrary numbers.
- What we can do: for each user, we can aggregate vectors of the items correspond to known cells of the utility matrix, possibly weighted by the value of the cell.

Recommendation

Given user profiles and item profiles (which are vectors), we can:

- Compute the (cosine) similarity between users and items to predict an user's preference to the item.

Recommendation by Machine Learning

Treat the recommendation problem as a machine learning problem given item profiles and utility matrices.
Idea: build a machine learning model (a classifier or a regression model) separately for each user, then use the model to predict users' preferences.

Collaborative Filtering

We only use the utility matrix to do recommendation. Two approaches:

- Similarity based: regard rows/columns of the utility matrix as vectors representation of users/items. The recommendation is then based on computing the similarity between vectors.
- Dimensionality reduction: Factorize the utility matrix M into the product of two long, thin matrices U and V. Then U and V are used to recover the unknown cells of M.

Similarity based

This approach is further divided into two:

- User based: users as central object for recommendation. For each user:
  - Find a set of similar users (using Jaccard or cosine) and recommend to the user items viewed by similar users.
  - Items can be ranked by the number of similar people viewed it (boolean case) or average rating (rating case).

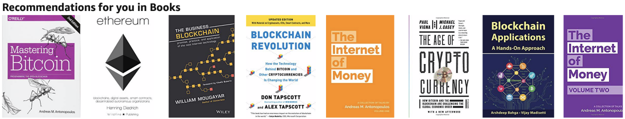

Figure: User-based recommendation on Amazon

Similarity based (Cont.)

This approach is further divided into two:

- Item based: items as central object for recommendation. For each item, we compute the set of similar items, ranked by similarity.

Figure: Item-based recommendation on Amazon

Dimensionality Reduction

- Factorize the utility matrix M into the product of two long, thin matrices U and V.
- U and V are used to recover the unknown cells of M.

$$\begin{aligned}
\underbrace{\begin{bmatrix} 1 & 2 & \\ 5 & & 1 \\ 4 & 3 & \end{bmatrix}}_{M} &= \underbrace{\begin{bmatrix} -0.13 & 1.83 \\ 1.87 & 0.5 \\ 1.2 & 1.42 \end{bmatrix}}_{U} \cdot \underbrace{\begin{bmatrix} 2.48 & 1.11 & 0 \\ 0.73 & 1.17 & 2 \end{bmatrix}}_{V} \\
&= \underbrace{\begin{bmatrix} 1 & 2 & {\color{red}3.66} \\ 5 & {\color{red}2.67} & 1 \\ 4 & 3 & {\color{red}2.83} \end{bmatrix}}_{M'}
\end{aligned} \tag{1}$$

Root-Mean-Square Error - RMSE

Let $P(M) = (i, j) : M[i, j]$ is known. RMSE of an UV-decomposition of M is:

$$\mathrm{RMSE}(U, V) = \sqrt{\frac{\sum_{(i,j)\in P(M)} (U[i] \cdot V[j] - M[i, j])^2}{|P(M)|}} \tag{2}$$

where $U[i]$ is the i-th row vector of U and $V[j]$ is the j-th column vector of V.

RMSE - An example

$$M = \begin{bmatrix} 1 & 2 & \\ 5 & & 1 \\ 4 & 3 & \end{bmatrix} \qquad U = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 1 \end{bmatrix} \qquad V = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{3}$$

RMSE - An example

$$M = \begin{bmatrix} 1 & 2 & \\ 5 & & 1 \\ 4 & 3 & \end{bmatrix} \qquad U = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 1 \end{bmatrix} \qquad V = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{3}$$

- $P(M) = \{(1,1),(1,2),(2,1),(2,3),(3,1),(3,2)\}$
- $U[1]V[1] = 3$, $U[1]V[2] = 4$, $U[2]V[1] = 3$, $U[2]V[3] = 4$, $U[3]V[1] = 2$, $U[3]V[2] = 3$

RMSE - An example

$$M = \begin{bmatrix} 1 & 2 & & \\ 5 & & & 1 \\ 4 & 3 & & \end{bmatrix} \qquad U = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 1 \end{bmatrix} \qquad V = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{3}$$

- $P(M) = \{(1,1),(1,2),(2,1),(2,3),(3,1),(3,2)\}$
- $U[1]V[1] = 3$, $U[1]V[2] = 4$, $U[2]V[1] = 3$, $U[2]V[3] = 4$, $U[3]V[1] = 2$, $U[3]V[2] = 3$

$$\mathrm{RMSE(U,V)} = \sqrt{\frac{2^2 + 2^2 + 2^2 + 3^2 + 2^2 + 0^2}{6}} = \sqrt{\frac{25}{6}} = 2.04 \tag{4}$$

Finding UV-decomposition

Idea: find UV-decomposition incrementally; at each step, we find a column of U (or a row of V), given other elements, so that the resulting matrix has minimum RMSE.

UV-decomposition - An Example

Given:

$$M = \begin{bmatrix} 1 & 2 & & \\ 5 & & 1 & \\ 4 & 3 & & \end{bmatrix} \tag{5}$$

and we start of with a random U, V, say:

$$U = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 1 \end{bmatrix} \qquad V = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{6}$$

Now, let's find a new column, say column 1, of U. Denote

$$U = \begin{bmatrix} x_1 & 2 \\ x_2 & 1 \\ x_3 & 1 \end{bmatrix} \qquad V = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{7}$$

UV-decomposition - An Example (Cont.)

$$UV = \begin{bmatrix} x_1 & 2 \\ x_2 & 1 \\ x_3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} = \begin{bmatrix} x_1 + 2 & 2x_1 + 2 & x_1 + 4 \\ x_2 + 1 & 2x_2 + 1 & x_2 + 2 \\ x_3 + 1 & 2x_3 + 1 & x_3 + 2 \end{bmatrix} \tag{8}$$

and the RMSE is:

$$\sqrt{\frac{(x_1+1)^2 + (2x_1)^2 + (x_2-4)^2 + (x_2+1)^2 + (x_3-3)^2 + (2x_3-2)^2}{6}}$$
$$= \sqrt{\frac{(5x_1^2 + 2x_1 + 2x_2^2 - 6x_2 + 5x_3^2 - 14x_3 + 31)}{6}} \tag{9}$$

which is minimized when:

$$x_1 = -0.2 \qquad x_2 = 1.5 \qquad x_3 = 1.4 \tag{10}$$

UV-decomposition - An Example (Cont.)

Thus

$$U = \begin{bmatrix} -0.2 & 2 \\ 1.5 & 1 \\ 1.4 & 1 \end{bmatrix} \qquad V = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{11}$$

and the RMSE is 1.66 (it is 2.04 previously).

UV-decomposition - An Example (Cont.)

Thus

$$U = \begin{bmatrix} -0.2 & 2 \\ 1.5 & 1 \\ 1.4 & 1 \end{bmatrix} \qquad V = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{11}$$

and the RMSE is 1.66 (it is 2.04 previously).

Next, we optimize for, say, the 2nd column of U. Repeat the above procedure:

$$U = \begin{bmatrix} -0.2 & x_1 \\ 1.5 & x_2 \\ 1.4 & x_3 \end{bmatrix} \qquad V = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{12}$$

We will end up with:

$$U = \begin{bmatrix} -0.2 & 1.8 \\ 1.5 & 0.5 \\ 1.4 & 1.4 \end{bmatrix} \qquad V = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{13}$$

with RMSE $= 1.57$

UV-decomposition - An Example (Cont.)

Now, we optimize for the 1st row of V:

$$U = \begin{bmatrix} -0.2 & 1.8 \\ 1.5 & 0.5 \\ 1.4 & 1.4 \end{bmatrix} \qquad V = \begin{bmatrix} y_1 & y_2 & y_3 \\ 1 & 1 & 2 \end{bmatrix} \tag{14}$$

$$UV = \begin{bmatrix} -0.2y_1 + 1.8 & -0.2y_2 + 1.8 & -0.2y_3 + 3.6 \\ 1.5y_1 + 0.5 & 1.5y_2 + 0.5 & 1.5y_3 + 1 \\ 1.4y_1 + 1.4 & 1.4y_2 + 1.4 & 1.4y_3 + 2.8 \end{bmatrix} \tag{15}$$

and RMSE is:

$$\sqrt{\frac{4.25y_1^2 - 21.1y_1 + 2y_2^2 - 4.4y_2 + 2.25y_3^2 + 30.25}{6}} \tag{16}$$

which is minimize when:

$$y_1 = 2.48 \quad y_2 = 1.01 \qquad y_3 = 0 \tag{17}$$

UV-decomposition - An Example (Cont.)

Now, we optimize for the 1st row of V:

$$U = \begin{bmatrix} -0.2 & 1.8 \\ 1.5 & 0.5 \\ 1.4 & 1.4 \end{bmatrix} \qquad V = \begin{bmatrix} 2.48 & 1.01 & 0 \\ 1 & 1 & 2 \end{bmatrix} \tag{18}$$

and the RMSE is 0.52 (it is 1.57 previously).

UV-decomposition - An Example (Cont.)

Repeat the process for several times, we obtain:

$$\underbrace{\begin{bmatrix} 1 & 2 & \\ 5 & & 1 \\ 4 & 3 & \end{bmatrix}}_{M} = \underbrace{\begin{bmatrix} -0.13 & 1.83 \\ 1.87 & 0.5 \\ 1.2 & 1.42 \end{bmatrix}}_{U} \cdot \underbrace{\begin{bmatrix} 2.48 & 1.11 & 0 \\ 0.73 & 1.17 & 2 \end{bmatrix}}_{V}$$
$$= \underbrace{\begin{bmatrix} 1 & 2 & 3.66 \\ 5 & 2.67 & 1 \\ 4 & 3 & 2.83 \end{bmatrix}}_{M'} \tag{19}$$

Finding the k-th column of U

Assume that we wan to factorize $M_{n \times m}$ into:

$$M_{n \times m} = U_{n \times d} \times V_{d \times m} \tag{20}$$

We start off with a random U and V. Now suppose that we want to find k-th column of U, denoted by $[x_1, \ldots, x_n]^T$. Other columns of U are known (and V is also known).

Finding the k-th column of U (Cont.)

Let $U[i]^{-k}$ be the row vector of U with k-th element removed and $V[j]^{-k}$ be the column vector of V with k-th element removed.

$$(UV)[i,j] = U[i] \cdot V[j] = U[i]^{-k} \cdot V[j]^{-k} + x_i V[j,k] \tag{21}$$

Let given user i, let $N(i)$ be the set of items that i rates. That is:

$$j \in N(i) \Leftrightarrow M[i,j] \neq \emptyset \tag{22}$$

Finding the k-th column of U (Cont.)

The term of RMSE^2 that involves x_i only is:

$$\begin{aligned}\mathrm{SSE}_i &= \sum_{j \in N(i)} (U[i]^{-k} \cdot V[j]^{-k} + x_i V[j,k] - M[i,j])^2 \\ &= \sum_{j \in N(i)} V[j,k]^2 x_i^2 + 2(\sum_{j \in N(i)} (U[i]^{-k} \cdot V[j]^{-k} - M[i,j]) V[j,k]) x_i + C \end{aligned} \tag{23}$$

where C is a constant that does not depend on any $x_1, \ldots, x_n$.

Finding the k-th column of U (Cont.)

The term of RMSE^2 that involves x_i only is:

$$\begin{aligned}\mathrm{SSE}_i &= \sum_{j \in N(i)} (U[i]^{-k} \cdot V[j]^{-k} + x_i V[j,k] - M[i,j])^2 \\ &= \sum_{j \in N(i)} V[j,k]^2 x_i^2 + 2\Big(\sum_{j \in N(i)} (U[i]^{-k} \cdot V[j]^{-k} - M[i,j]) V[j,k]\Big) x_i + C\end{aligned} \tag{23}$$

where C is a constant that does not depend on any $x_1, \ldots, x_n$. SSE_i is minimized when:

$$x_i = -\frac{\sum_{j \in N(i)} (U[i]^{-k} \cdot V[j]^{-k} - M[i,j]) V[j,k]}{\sum_{j \in N(i)} V[j,k]^2} \tag{24}$$

Finding the k-th row of V

Let $y_1, y_2, \ldots, y_m$ be k-th row of V that we want to find, given other rows of V. By a similar derivation to find k-th column of U, we end up with:

$$y_j = -\frac{\sum_{i \in N(j)} (U[i]^{-k} \cdot V[j]^{-k} - M[i,j]) U[i,k]}{\sum_{i \in N(j)} U[i,k]^2} \tag{25}$$

for $j = 1, 2, \ldots, m$. Where $N(j)$ is the set of users that rate item j.

UV-decomposition algorithm

```
UVDecomposition(M, n, m, d)
    Initialize U_{n×d}, V_{d×m} randomly
    repeat T times
        for each k ← 1 to d
            for each i ← 1 to n
                Compute x[i] be as in Equation 24
            for each i ← 1 to n
                U[i, k] ← x[i]
        for each k ← 1 to d
            for each j ← 1 to m
                Compute y[j] be as in Equation 25
            for each j ← 1 to m
                V[k, j] ← y[j]
    return U, V
```

Running time: $O(Td|\#M|)$ where $|\#M|$ is the number of known cells of M.