Recommender System

-Quantifying the World-

Lecturer: Darren Homrighausen, PhD

Overview

Providing personalized recommendations is important in an online marketplace

It benefits both buyers and sellers:

- buyers are shown interesting products that they might not have found on their own
- products get more exposure beyond the sellers own marketing efforts

In this class we review some of the methods used for making recommendations

(Such as at Etsy. Reference: Rob Hall)

Overview

Computing recommendations basically consists of two stages:

1. Build a model of users' interests based on a matrix of historic data

   (For example, their past purchases or their favorite listings)

2. Compute recommendations by finding a set of items for each user which approximately maximizes an estimate of his/her interest.

   (The model of users and items can be also used in other ways, such as finding users with similar interests, items which are similar from a taste perspective, items which complement each other and could be purchased together, etc.)

Matrix Factorization

At Etsy, they deal with "implicit feedback"

(observe only the indicators of users interactions with items)

This is in contrast to "explicit feedback"

(users give ratings (e.g. 3 of 5 stars) to items they have experienced)

This implicit feedback can be represented as a binary matrix

- the elements are ones if a user liked the item
- a zero if he/she did not

  (Zeros only indicate a user has not expressed an interest so far. This may be due to disinterest or not having seen that item yet while browsing)

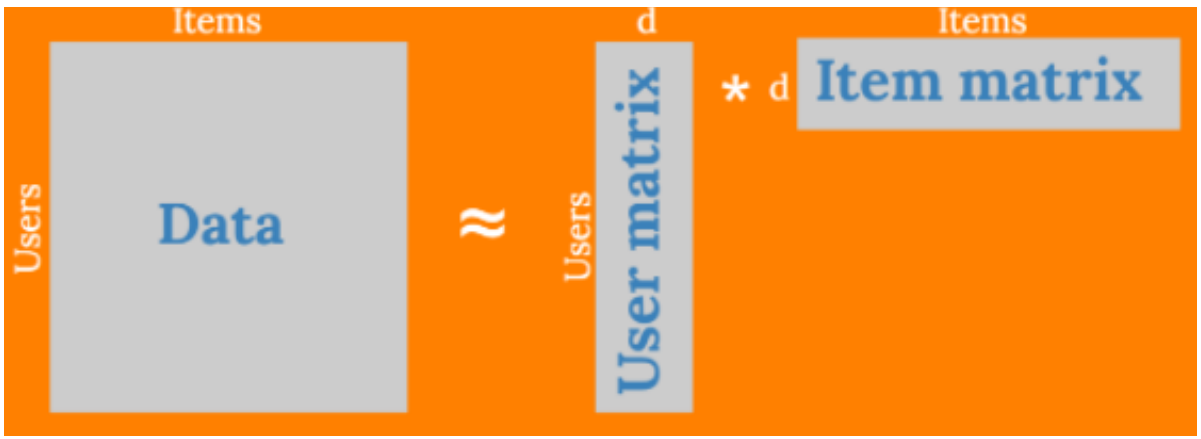
Users					...
Alice	1	1	0	0	
Bob	0	0	1	1	
Corey	1	0	1	0	
...					

Matrix Factorization

The underpinning assumption that matrix factorization models make is that the affinity between a user and an item is explained by a low-dimensional linear model

(Remember Latent Semantic Analysis?)

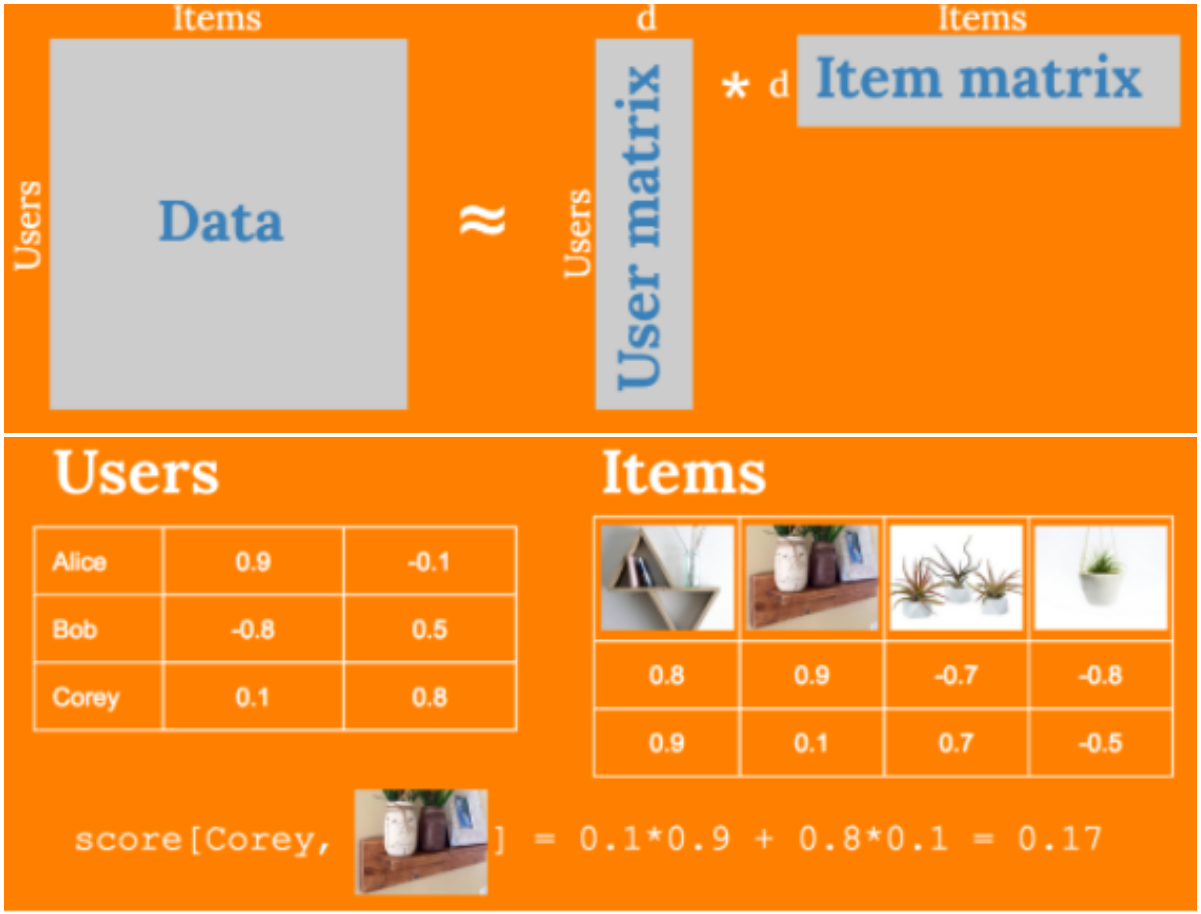
This means that each item and user really corresponds to an unobserved latent factors of some small dimension.

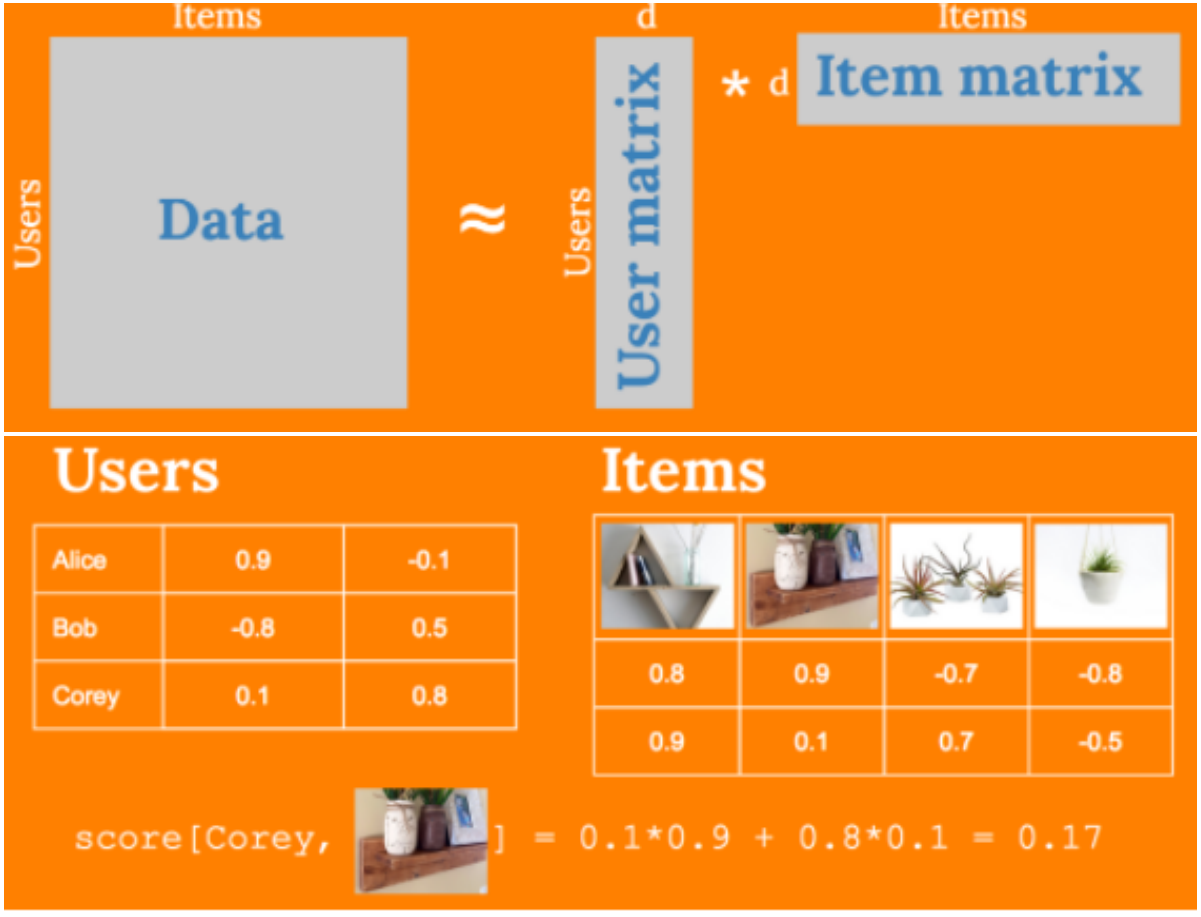


Matrix Factorization

We compute these latent factors so that the inner product will approximate the observed values in the implicit feedback matrix

(i.e., it will be close to one in the case the user favorited that item and close to zero if they did not)





We estimate the User matrix (u) and Item matrix (v) by solving

$$\min_{u,v} \sum_{i,j} (m[i,j] - u[i,] \,\%*\%\, v[j,])^2$$

(Note: In our R notation v = Item matrix$^\top$)

The methods

Methods

We estimate the User matrix (u) and Item matrix (v) by solving

$$\min_{u,v} \sum_{i,j} (m[i,j] - u[i,] \,\%*\%\, v[j,])^2$$

- Alternating least squares: Alternate between least squares for u (User matrix) and v (Item matrix)
- Singular Value Decomposition (SVD): Decompose $m = UDV^T$ via the singular value decomposition

(In both cases, the estimated latent factors are restricted to have d columns)

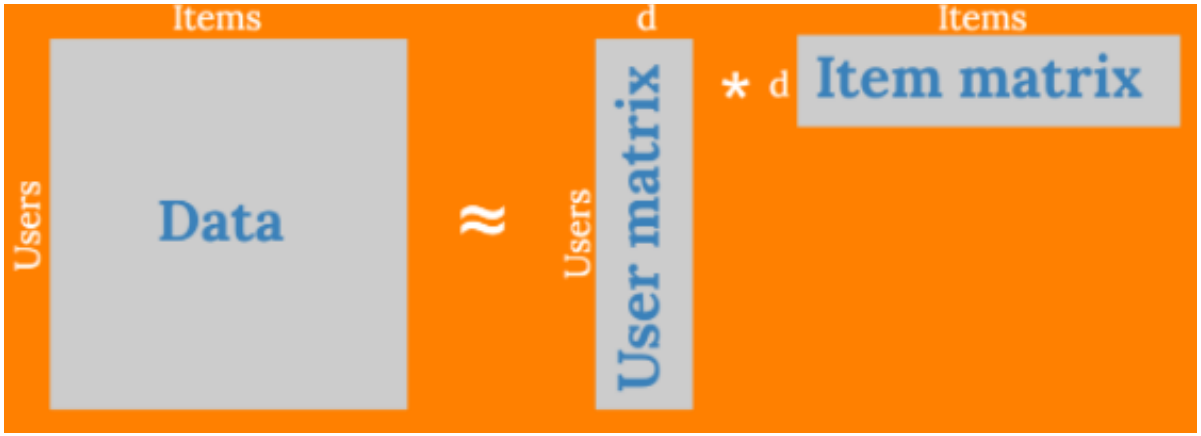
Methods

We estimate the User matrix (u) and Item matrix (v) by solving

$$\min_{u,v} \sum_{i,j} (m[i,j] - u[i,] \,\%*\%\, v[j,])^2$$

- **Alternating least squares:** Alternate between least squares for u (User matrix) and v (Item matrix)
- **Singular Value Decomposition (SVD):** Decompose $m = UDV^T$ via the singular value decomposition

(In both cases, the estimated latent factors are restricted to have d columns)



Alternating least squares

$$\sum_{i,j}(m[i,j]-u[i,] \;\%*\%\; v[j,])^2 = \sum_i \sum_j (m[i,j]-u[i,] \;\%*\%\; v[j,])^2$$

So, for a fixed i,

- $m[i,j] = Y_j$
- $u[i,] = \beta$
- $v[j,] = X_j$

$$\rightarrow \text{Linear regression: } \sum_j (Y_j - X_j^\top \beta)^2$$

So, we estimate $u[i,]$ via least squares of $m[i,]$ onto v

Similarly, we estimate $v[j,]$ via least squares of $m[,j]$ onto u

Then, we iterate between these steps until convergence

(Note that we need to initialize one of these steps with a random matrix)

SVD

Alternating least squares can be expensive to compute due to having to solve a large number of linear regressions

Hence, we can turn to the SVD to get u and v

$$m = UDV^\top \qquad \rightarrow \qquad u = U[, 1 : d], v = V[, 1 : d]$$

(Some implementations put a $D^{1/2}$ in each component: $UDV^\top = (UD^{1/2})(VD^{1/2})^\top$)

Approximate SVD

When m is a large matrix, forming $m = UDV^\top$ is expensive

Some recent work on randomized SVDs can help

(page 121, `http://amath.colorado.edu/faculty/martinss/Pubs/2012_halko_dissertation.pdf`)

In the interest of time, I'll not discuss this too carefully. It is implemented in the short function approx_svdF in the R code

The recommendations

Approximate SVD

Now we have u and v that have d columns

(Here, think of d as very small, like between 2 and 10)

u has the same number of rows as there are users

v has the same number of rows as there are items

We get recommendations by looking at the largest entries in:

$$\hat{m} = u \,\%*\%\, v^\top$$

that don't have a "1" in the original matrix m