Introduction to Data Science

RECSYS MATRIX FACTORIZATIONS

BRIAN D'ALESSANDRO

LATENT TASTE FACTORS

We generally observe some order and structure to the types of items people consume/view.

Crime Thrillers

Artistic Dramas

Handsome Leading Men

LATENT TASTE FACTORS

What drives these observations? In other words, why do we do watch what we watch?

- Genre (comedy vs. romance)
- Audience (children vs. adults)
- Style (quirky vs. serious)
- Depth of Character
- Presence of certain actors

We want a computational way to be able to extract these properties and not rely on human curation.

BEYOND COLLABORATIVE FILTERING

The Netflix recommendation system contest in the mid-aughts ushered in a new paradigm for making recommendations.

Given a user-item matrix, decompose:

$$A = X \Sigma Y^T$$

Instead of creating user taste, or item similarity neighborhoods, we can predict a user's rating on an item by uncovering the latent dimensions of the ratings matrix.

FACTORIZING THE RATINGS MATRIX

We simplify the factorization to:

$$A = XY^T$$

X Each element X_{ij} of X represents how much user i has an affinity for the latent movie dimension X_j.

Y Each column of Y represents a latent movie dimension. The value Y_{ij} tells us how much movie i can be described by the latent movie dimension j.

Because we don't follow the standard SVD procedure, with a middle diagonal matrix of singular values, X and Y in this decomposition will not consist of orthonormal vectors.

LATENT TASTE FACTORS FROM MF

When uncovering latent factors, we usually only want a subset k, where k<<min(M,N)

X_k

	LF 1	LF 1	LF 3
User 1	-0.7	-0.8	-0.4
User 2	0.2	-0.6	1.0
User 3	-0.8	-0.1	-0.8
User 4	0.4	0.3	-0.1
…			
User N	-0.3	1.0	0.4

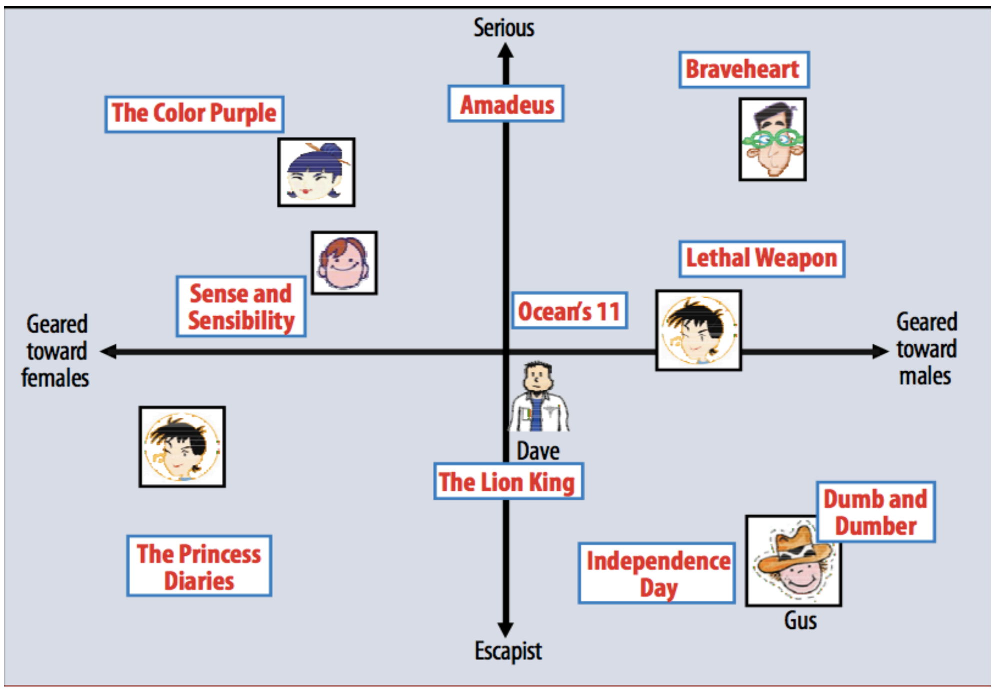
Y_k^T

	Item 1	Item 2	Item 3	…	Item M
LF 1	-0.4	0.6	0.8	…	-0.8
LF 2	-0.8	-0.5	-0.7	…	-0.4
LF 3	0.1	0.9	0.7	…	-0.7

$$A = X_k Y_k^T$$

LATENT FACTORS REVEAL MOVIE CLUSTERS

An example set of movies and how they load on two latent variables.



By plotting the first two movie (item) factors against each other we see movies cluster along two dimensions that have semantic meaning.

Note that the meaning itself is due to human interpretation of the clusters, and not the MF itself.

Source: "Matrix Factorization Techniques for Recommender Systems" Bell, Koren, Volinsky

LEARNING THE FACTORIZATION

We can set this recommendation problem up as a supervised learning problem.

Minimize the sum of squares predictions of observed ratings.

$$\min_{x_\star, y_\star} \sum_{r_{u,i} \text{ is known}} (r_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2)$$

We regularize the components of U and V to avoid over-fitting

LEARNING WITH IMPLICIT FEEDBACK

In many (if not most) cases we don't have explicit user ratings of an item. We only know if a user consumed or viewed the item.

To adapt to this, we introduce a new variable:

$$p_{ui} = \begin{cases} 1 & \textit{if user consumed item i} \\ 0 & \textit{if user did not consume item i} \end{cases}$$

Because consumption isn't always just binary (i.e., how many seconds did they listen to song or watch video), we can create a weighting factor.

$$c_{ui} = 1 + \alpha r_{ui}$$

- r_{ui} is a non-zero observed consumption amount
- α is a weight that we set

Then we redefine our loss function using p_{ui} and c_{ui}

$$\min_{x_\star, y_\star} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

Source: "Collaborative Filtering for Implicit Feedback Datasets", Hu, Koren & Volinsky

TRAINING THE MF: ALTERNATING LEAST SQUARES

Algorithm 1 ALS for Matrix Completion

Initialize X, Y

repeat

for $u = 1 \ldots n$ **do**

$$x_u = \Big(\sum_{r_{ui} \in r_{u*}} y_i y_i^\intercal + \lambda I_k \Big)^{-1} \sum_{r_{ui} \in r_{u*}} r_{ui} y_i$$

end for

for $i = 1 \ldots m$ **do**

$$y_i = \Big(\sum_{r_{ui} \in r_{*i}} x_u x_u^\intercal + \lambda I_k \Big)^{-1} \sum_{r_{ui} \in r_{*i}} r_{ui} x_u$$

end for

until convergence

Given an objective function:

$$\min_{X,Y} \sum_{r_{ui}\ observed} (r_{ui} - x_u^\intercal y_i)^2 + \lambda \Big(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \Big)$$

Pick a parameter vector (X or Y) to hold constant.

Set the derivative w.r.t. the non-constant parameter to zero and solve for it.

Do the same for the other parameters.

Repeat until convergence.

TRAINING THE MF: STOCHASTIC GRADIENT DESCENT

Algorithm 2 Streaming ALS using SGD

for *new* r_{ui} **do**

$$x_u \leftarrow x_u - \alpha(r_{ui} - x_u^{\intercal} y_i) y_i + \lambda x_u$$

$$y_i \leftarrow y_i - \alpha(r_{ui} - x_u^{\intercal} y_i) x_u + \lambda y_i$$

end for

Given an objective function:

$$\min_{X,Y} \sum_{r_{ui}\, observed} (r_{ui} - x_u^{\intercal} y_i)^2 + \lambda\left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2\right)$$

For each record in the data:

- Compute the derivative of the loss function w.r.t. to each parameter in X and Y
- Modify the parameter by a magnitude proportional to the negative of the Loss gradient
- Continue until desired convergence.

ALS vs SGD?

- SGD is generally faster and easier to implement than ALS
- ALS easier to parallelize, so can be faster with access to a large cluster
- ALS may be faster if the data lacks sparsity, as SGD would have to loop through each instance separately
- The better algorithm depends on the data and problem. Often the only way to tell is to test.

Sources:
http://cs229.stanford.edu/proj2014/Christopher%20Aberger,%20Recommender.pdf
https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf

THE RATING PREDICTION

The rating prediction for user i on item j is then an inner product between the the user's preferences for each latent factor and the item's strength on that factor.

y_{jt} indicates how much factor t describes item j

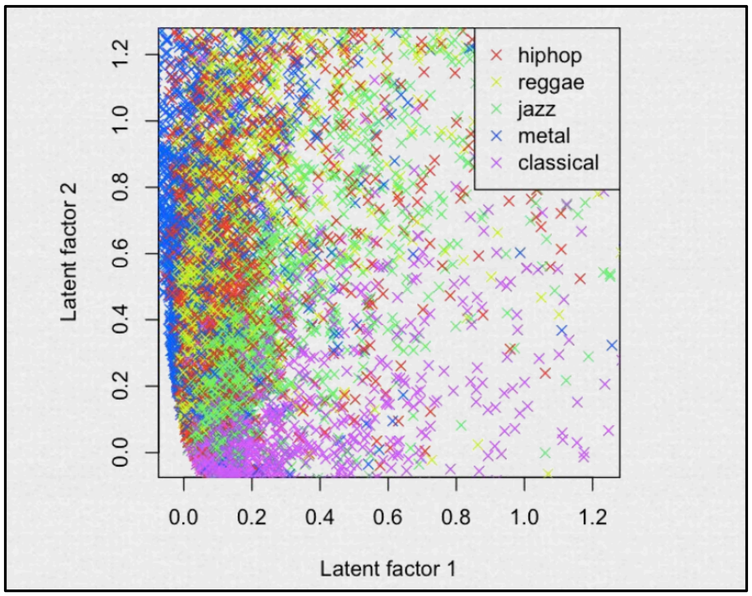
$$r_{ij} = x_{i1} * y_{j1} + x_{i2} * y_{j2} + \ldots + x_{ik} * y_{jk} = \Sigma_{t=1,k}\, x_{it} * y_{jt}$$

x_{it} indicates how user i prefers factor *t*

GOING BEYOND PREDICTIONS

The learned vectors X_k & Y_k can be used as embedding vectors on the user/items respectively. These vectors can then be used in other applications

Clustering: we can build user/item clusters for general insights or for making recommendations



Nearest Neighbors: we can use the vectors to find most similar users/items for the purpose of recommendations