Class 9
Latent Factor Models and Matrix Factorization

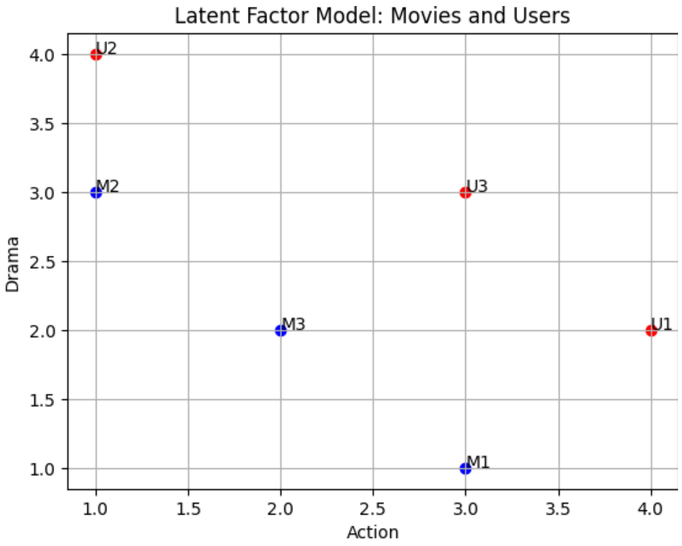
Latent Factor Models

- Latent factor models are a class of statistical models used in various fields such as machine learning, statistics, and social sciences to represent relationships between observed variables and unobserved (latent) factors. These models assume that the observed variables are influenced by underlying, unobservable factors or variables.
- Aims to capture hidden or latent factors that explain observed data patterns.
- In the context of recommendation systems and collaborative filtering, latent factor models are used to uncover underlying factors (preferences, characteristics) that influence user-item interactions.
- The latent factors are not directly observed but are inferred from the observed data through the model.

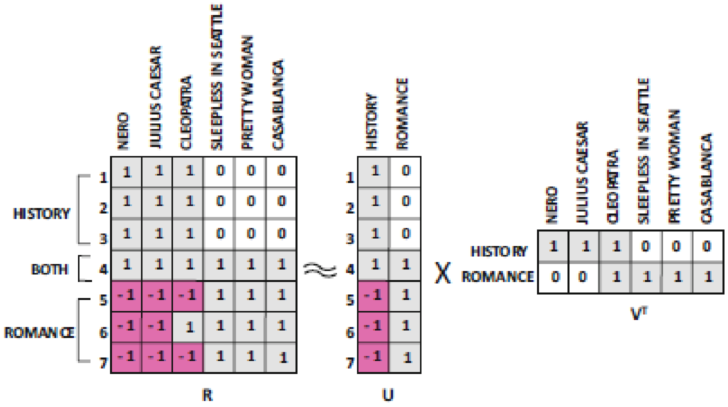
Latent Factor Models (contd...)

- Finds a lower-dimensional space (latent space) where the data can be effectively represented while preserving important information.
- Learns the positions of users and movies in this latent space based on observed ratings.
- By aligning users' preferences with the characteristics of movies in the latent space, the model can predict how users would rate unseen movies.

Geometric Illustration of Latent Factor Models



Low-Rank Intuition of Latent Factor Models

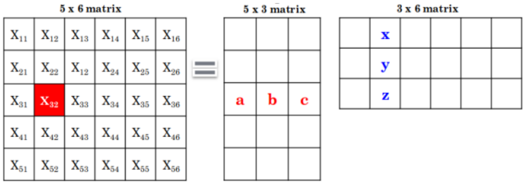


$$
\underbrace{\begin{array}{ll|cccccc}
 & & \text{NERO} & \text{JULIUS CAESAR} & \text{CLEOPATRA} & \text{SLEEPLESS IN SEATTLE} & \text{PRETTY WOMAN} & \text{CASABLANCA} \\
\hline
 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
\text{HISTORY} & 2 & 1 & 1 & 1 & 0 & 0 & 0 \\
 & 3 & 1 & 1 & 1 & 0 & 0 & 0 \\
\text{BOTH} & 4 & 1 & 1 & 1 & 1 & 1 & 1 \\
 & 5 & -1 & -1 & -1 & 1 & 1 & 1 \\
\text{ROMANCE} & 6 & -1 & -1 & 1 & 1 & 1 & 1 \\
 & 7 & -1 & -1 & -1 & 1 & 1 & 1
\end{array}}_{R}
\approx
\underbrace{\begin{array}{c|cc}
 & \text{HISTORY} & \text{ROMANCE} \\
\hline
1 & 1 & 0 \\
2 & 1 & 0 \\
3 & 1 & 0 \\
4 & 1 & 1 \\
5 & -1 & 1 \\
6 & -1 & 1 \\
7 & -1 & 1
\end{array}}_{U}
\times
\underbrace{\begin{array}{l|cccccc}
 & \text{NERO} & \text{JULIUS CAESAR} & \text{CLEOPATRA} & \text{SLEEPLESS IN SEATTLE} & \text{PRETTY WOMAN} & \text{CASABLANCA} \\
\hline
\text{HISTORY} & 1 & 1 & 1 & 0 & 0 & 0 \\
\text{ROMANCE} & 0 & 0 & 1 & 1 & 1 & 1
\end{array}}_{V^T}
$$

Example of rank-2 matrix factorization

Matrix Factorization (MF)

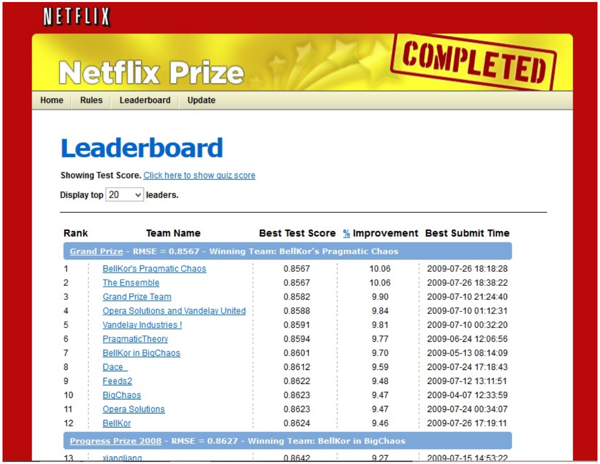
- **MF** – To find two(or more) matrices such that when we multiply them we will get back the original matrix.


5 x 6 matrix
5 x 3 matrix
3 x 6 matrix
X_{11} X_{12} X_{13} X_{14} X_{15} X_{16}
X_{21} X_{22} X_{12} X_{24} X_{25} X_{26}
X_{31} X_{32} X_{33} X_{34} X_{35} X_{36}
X_{41} X_{42} X_{43} X_{44} X_{45} X_{46}
X_{51} X_{52} X_{53} X_{54} X_{55} X_{56}
=
a b c
x
y
z

$$X_{32} = (a, b, c) \cdot (x, y, z) = a * x + b * y + c * z$$

Netflix Prize Competition

The Netflix Prize competition, launched by Netflix in 2006, played a significant role in popularizing Matrix Factorization techniques in the field of recommendation systems.



Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				
13	xiangliang	0.8642	9.27	2009-07-15 14:53:22

https://www.herox.com/blog/839-history-of-crowdsourcing-the-netflix-prize

Collaborative Prediction

- Collaborative prediction is formalized as a simple matrix-completion problem.
- Suppose we have a set of m users (U) and set of n items (M).
- Let $Y = [y_{ij}]$ is $m \times n$ ($|U| \times |M|$) user/item rating matrix.
- Each element $y_{ij} \epsilon \{0, 1, ...R\}$, R is total level of ratings.
- 0 represent the unknown rating.

The goal is to predict the unknown ratings.

	M1	**M2**	**M3**	**M4**
U1	5	3	0	1
U2	4	0	0	1
U3	1	1	0	5
U4	1	0	0	4
U5	0	1	5	4

User-Item Rating matrix (Y)

Matrix Factorization for Collaborative Prediction

- Matrix factorization is a key technique used in latent factor models. It involves decomposing a high-dimensional user-item interaction matrix into lower-dimensional matrices that capture latent factors.
- Decomposes the rating matrix into two lower-dimensional matrices which represent users and items in a latent space characterized by latent factors.
- The product of the matrices approximates the original matrix and helps in making predictions for missing values.
- Assume there are two hidden factors: *Action* and *Romance*.
- Find how much each user likes action movies and romance movies, and how much action and romance each movie contains.
- It is achieved by low-dimensional embedding process.

Mingrui Wu. Collaborative Filtering via Ensembles of Matrix Factorizations. In Knowledge Discovery and Data Mining (KDD), pages 43-47, 2007.

User and Item latent factors of toy example

	Action	*Romance*
U1	2.13	-0.49
U2	1.73	-0.28
U3	1.06	2.01
U4	0.92	1.57
U5	1.06	1.61

User Latent feature matrix

- The values in each cell represent how much the user likes each factor.

	M1	**M2**	**M3**	**M4**
Action	2.18	1.34	1.44	0.90
Romance	-0.64	-0.23	2.07	1.96

Item Latent feature matrix

- The values indicate how much of each factor is present in the movie.

Missing value prediction

- Predict the ratings for missing values by taking the dot product of the user preferences over features and movie factors.

	M1	**M2**	**M3**	**M4**
U1	4.95	2.96	2.05	0.95
U2	3.95	2.38	1.91	1.00
U3	1.02	0.95	5.68	4.89
U4	1.00	0.87	4.57	3.90
U5	1.28	1.05	4.85	4.10

Predicted User-Item Rating matrix

Matrix Factorization for Collaborative Prediction (contd..)

- Formally, given a user-item rating matrix $Y \in \mathbb{R}^{m \times n}$ where m is the number of users and n is the number of items, we find two matrices, $U \in \mathbb{R}^{m \times k}$ and $V \in \mathbb{R}^{n \times k}$, where k is the dimension of the embedding, such that the product is approximately equal to Y on observed entries, i.e., $U \times V^T = \hat{Y} \approx Y$.
- The i^{th} row $\bar{u}_i$ of U is referred to as a user factor and it contains k entries corresponding to the affinity of user i towards the k features.
- Each row $\bar{v}_i$ of V corresponds to an item factor.
- Each rating y_{ij} in Y can be approximately expressed as a dot product of the i^{th} user factor and j^{th} item factor.

$$y_{ij} \approx \bar{u}_i . \bar{v}_j$$

$$y_{ij} \approx \sum_{s=1}^{k} u_{is} . v_{js}$$

$= \sum_{s=1}^{k}$(Affinity of user i to concept s) $\times$ (Affinity of item j to concept s)

Matrix Factorization for Collaborative Prediction (contd..)

- The problem can be formulated as following optimization problem,

$$Minimize\ \mathcal{J}(U,V) = \sum_{(i,j)\in\Omega} Loss(y_{ij}, U_i V_j^T)$$

- In order to avoid overfitting, a regularization term is introduced into the optimization function.

$$Minimize\ \mathcal{J}(U,V) = \sum_{(i,j)\in\Omega} Loss(y_{ij}, U_i V_j^T) + \mathcal{R}(U,V)$$

- The objective is to minimize $\mathcal{J}$,

$$\mathcal{J}(U,V) = \sum_{(i,j)\in\Omega} (y_{ij} - U_i V_j^T)^2 + \lambda(||U||_F + ||V||_F) \qquad (1)$$

where, Ω is the set of observed (i,j) pairs, $\lambda > 0$ is the regularization parameter.

Updation of U, V

One can solve the above optimization function (1) using gradient descent method, by updating U, V, using (2)

- U, V are updated iteratively in Gradient-search as,

$$\begin{aligned} U_{t+1} &= U_t - c\ \tfrac{\partial \mathcal{J}}{\partial U} \\ V_{t+1} &= V_t - c\ \tfrac{\partial \mathcal{J}}{\partial V} \end{aligned} \tag{2}$$

where c is the step size.

Toy Example

	M1	M2	M3	M4
U1	5	3	0	1
U2	4	0	0	1
U3	1	1	0	5
U4	1	0	0	4
U5	0	1	5	4

User-Item Rating matrix (Y)

	M1	M2	M3	M4
U1	4.95	2.96	2.05	0.95
U2	3.95	2.38	1.91	1.00
U3	1.02	0.95	5.68	4.89
U4	1.00	0.87	4.57	3.90
U5	1.28	1.05	4.85	4.10

Predicted User-Item Rating matrix ($\hat{Y}$)

	k_1	k_2
U1	2.13	-0.49
U2	1.73	-0.28
U3	1.06	2.01
U4	0.92	1.57
U5	1.06	1.61

User Latent feature matrix (U)

	M1	M2	M3	M4
k_1	2.18	1.34	1.44	0.90
k_2	-0.64	-0.23	2.07	1.96

Item Latent feature matrix (V)

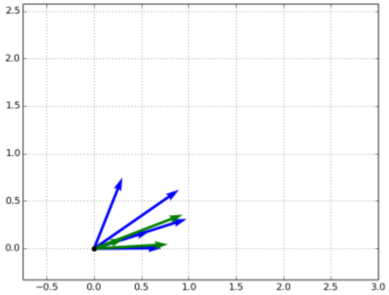
Initial U, V

- If we apply MF (with squared loss) on above shown example by considering the number of latent features as two (i.e., $k = 2$), the following are the $U_{5\times 2}$, $V^T_{2\times 4}$ and the predicted matrix ($\hat{Y}_{5\times 4} = UV^T$) which we get initially with the random initialization of U and V.

$$\underbrace{\begin{pmatrix} 0.5904876 & 0.18520637 \\ 0.89651422 & 0.62040181 \\ 0.97932976 & 0.31239695 \\ 0.71248937 & 0.00531318 \\ 0.2988344 & 0.7485681 \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} 0.93627203 & 0.78030931 & 0.01137135 & 0.31557944 \\ 0.35834808 & 0.04508127 & 0.01842057 & 0.10453503 \end{pmatrix}}_{V^T} =$$

$$\underbrace{\begin{pmatrix} 0.6192254 & 0.78030931 & 0.01137135 & 0.31557944 \\ 0.35834808 & 0.04508127 & 0.01842057 & 0.10453503 \\ 1.0288659 & 0.7782634 & 0.0168908 & 0.3417128 \\ 0.6689878 & 0.5562016 & 0.0081998 & 0.2254024 \\ 0.5480382 & 0.2669297 & 0.0171872 & 0.1725576 \end{pmatrix}}_{\hat{Y}}$$

$$\underbrace{\begin{pmatrix} 0.5904876 & 0.18520637 \\ 0.89651422 & 0.62040181 \\ 0.97932976 & 0.31239695 \\ 0.71248937 & 0.00531318 \\ 0.2988344 & 0.7485681 \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} 0.93627203 & 0.78030931 & 0.01137135 & 0.31557944 \\ 0.35834808 & 0.04508127 & 0.01842057 & 0.10453503 \end{pmatrix}}_{V^T}$$



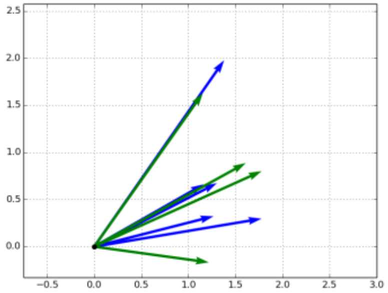
Pictorial representation of latent feature vectors (U - blue color vectors and V - green color vectors) in latent space initially

- Once we calculate the predicted rating matrix (UV^T), calculate the difference between the predicted values and the actual (existing) values and try to minimize the difference recursively by using Gradient Descent (GD).
- GD helps to find out in which direction (U and V) we should go to minimize the error and keep on going iteratively until no more error exist.
- So, for the given toy example, when we apply GD and going on updating U and V iteratively, to minimize the error, the updated U, V, $\hat{Y}$ which were obtained at 1500th iteration are shown below.

$$\underbrace{\begin{pmatrix} 1.77752894 & 0.29656766 \\ 1.16303033 & 0.66145795 \\ 1.30310564 & 0.67387977 \\ 1.26769425 & 0.32239559 \\ 1.38022296 & 1.97844308 \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} 1.77800423 & 1.21778941 & 1.15383339 & 1.6100205 \\ 0.80150418 & -0.16676241 & 1.62889942 & 0.88737676 \end{pmatrix}}_{V^T} =$$

$$\underbrace{\begin{pmatrix} 3.3982 & 2.1152 & 2.5341 & 3.1250 \\ 2.5980 & 1.3060 & 2.4194 & 2.4595 \\ 2.8570 & 1.4745 & 2.6012 & 2.6960 \\ 2.5124 & 1.4900 & 1.9879 & 2.3271 \\ 4.0398 & 1.3509 & 4.8152 & 3.9778 \end{pmatrix}}_{\hat{Y}}$$

$$\underbrace{\begin{pmatrix} 1.77752894 & 0.29656766 \\ 1.16303033 & 0.66145795 \\ 1.30310564 & 0.67387977 \\ 1.26769425 & 0.32239559 \\ 1.38022296 & 1.97844308 \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} 1.77800423 & 1.21778941 & 1.15383339 & 1.6100205 \\ 0.80150418 & -0.16676241 & 1.62889942 & 0.88737676 \end{pmatrix}}_{V^T}$$



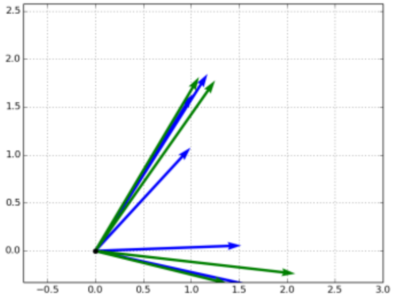
Pictorial representation of latent feature vectors (U - blue color vectors and V - green color vectors) in latent space at 1500th iteration

- The updated U, V, $\hat{Y}$ which were obtained at 3000^{th} iteration are shown below.

$$\underbrace{\begin{pmatrix} 2.11668492 & -0.46805813 \\ 1.52168501 & 0.05662858 \\ 1.01298558 & 1.6322625 \\ 0.99366696 & 1.07722617 \\ 1.17065079 & 1.84492609 \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} 2.08575665 & 1.38337271 & 1.24972628 & 1.08183373 \\ -0.23726548 & -0.3410333 & 1.77726989 & 1.8130988 \end{pmatrix}}_{V^T} =$$

$$\underbrace{\begin{pmatrix} 4.52594 & 3.08779 & 1.81341 & 1.44127 \\ 3.16043 & 2.08575 & 2.00233 & 1.74888 \\ 1.72556 & 0.84468 & 4.16693 & 4.05534 \\ 1.81696 & 1.00724 & 3.15633 & 3.02810 \\ 2.00396 & 0.99027 & 4.74192 & 4.61148 \end{pmatrix}}_{\hat{Y}}$$

$$\underbrace{\begin{pmatrix} 2.11668492 & -0.46805813 \\ 1.52168501 & 0.05662858 \\ 1.01298558 & 1.6322625 \\ 0.99366696 & 1.07722617 \\ 1.17065079 & 1.84492609 \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} 2.08575665 & 1.38337271 & 1.24972628 & 1.08183373 \\ -0.23726548 & -0.3410333 & 1.77726989 & 1.8130988 \end{pmatrix}}_{V^T}$$



Pictorial representation of latent feature vectors (U - blue color vectors and V - green color vectors) in latent space at 3000th iteration

- When we are updating, at some point of time, the error doesn't change, and we get local minima (or sometimes global minima) and the feature vectors obtained here are final vectors (U and V).

- For the above toy example, the user and item latent feature vectors obtained at 4999^{th} iteration are shown below, after which there is no change in the error.

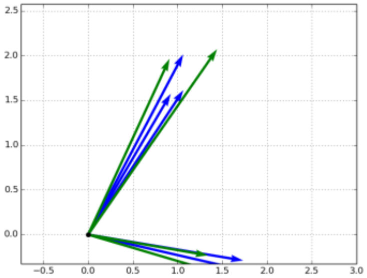
$$\underbrace{\begin{pmatrix} 2.13929538 & -0.49059839 \\ 1.73310635 & -0.28993261 \\ 1.06195536 & 2.01308987 \\ 0.92227177 & 1.57269387 \\ 1.06416843 & 1.61413303 \end{pmatrix}}_{U} \underbrace{\begin{pmatrix} 2.1818626 & 1.34069235 & 1.44112731 & 0.90992741 \\ -0.64414485 & -0.23296697 & 2.07216746 & 1.96481116 \end{pmatrix}}_{V^T} =$$

$$\underbrace{\begin{pmatrix} 4.98367 & 2.98243 & 2.06639 & 0.98267 \\ 3.96816 & 2.39111 & 1.89684 & 1.00734 \\ 1.02032 & 0.95477 & 5.70187 & 4.92164 \\ 0.99923 & 0.87010 & 4.58800 & 3.92925 \\ 1.28213 & 1.05068 & 4.87836 & 4.13978 \end{pmatrix}}_{\hat{Y}}$$

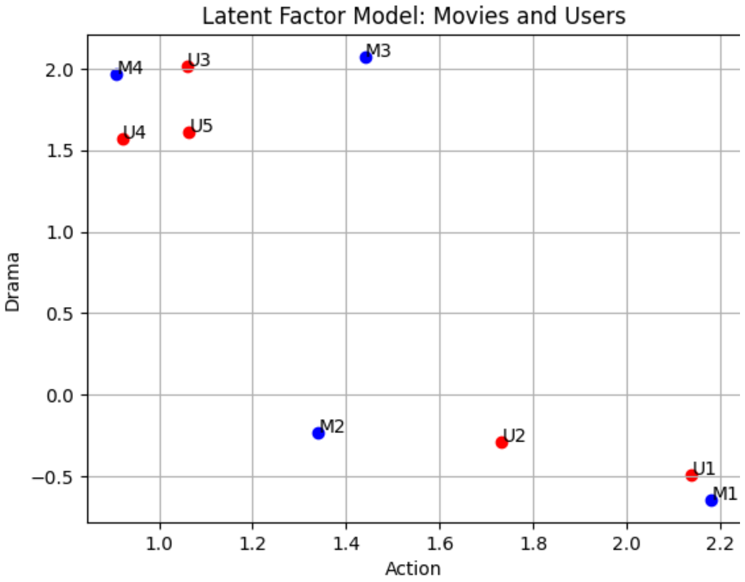
$$\underbrace{\begin{pmatrix} 5 & 3 & 0 & 1 \\ 4 & 0 & 0 & 1 \\ 1 & 1 & 0 & 5 \\ 1 & 0 & 0 & 4 \\ 0 & 1 & 5 & 4 \end{pmatrix}}_{Y}$$

The corresponding vectors are depicted in latent space as shown below.

$$\underset{U}{\begin{pmatrix} 2.13929538 & -0.49059839 \\ 1.73310635 & -0.28993261 \\ 1.06195536 & 2.01308987 \\ 0.92227177 & 1.57269387 \\ 1.06416843 & 1.61413303 \end{pmatrix}} \underset{V^T}{\begin{pmatrix} 2.1818626 & 1.34069235 & 1.44112731 & 0.90992741 \\ -0.64414485 & -0.23296697 & 2.07216746 & 1.96481116 \end{pmatrix}}$$



Pictorial representation of latent feature vectors (U - blue color vectors and V - green color vectors) in latent space at 4999th iteration

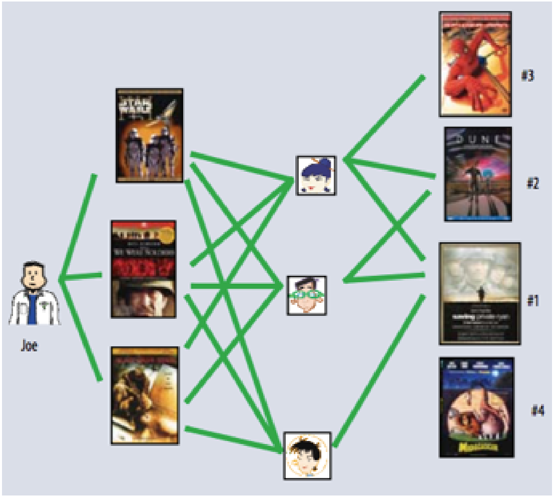


Latent Factor Model: Movies and Users
M4
U3
M3
U4
U5
M2
U2
U1
M1
Drama
Action
2.0
1.5
1.0
0.5
0.0
−0.5
1.0
1.2
1.4
1.6
1.8
2.0
2.2

Neighborhood Models vs Latent Factor Models

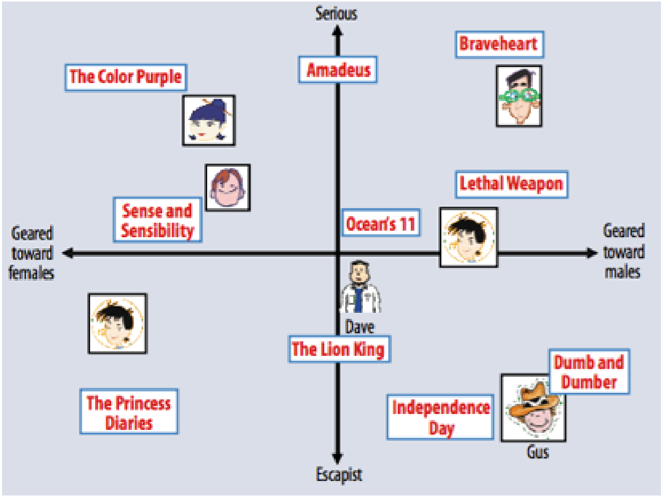
- MF is a latent factor model approach that decomposes the user-item interaction matrix into lower-dimensional matrices representing users and items in a latent space. It learns latent factors that capture underlying patterns and preferences in the data.
- Neighborhood methods, such as k-nearest neighbors (k-NN) in collaborative filtering, rely on finding similar users or items based on their past interactions. These methods use the notion of proximity or similarity metrics to make predictions.
- MF transforms the original sparse user-item matrix into a lower-dimensional latent space. It captures latent factors (e.g., user preferences, item characteristics) that explain user-item interactions.
- Neighborhood methods operate directly on the original user-item matrix or a similarity matrix. They do not transform the data into a lower-dimensional space but rely on pairwise similarities or distances between users or items.

Neighborhood Models vs Latent Factor Models



Koren, Y.; Bell, R.; and Volinsky, C. Matrix factorization techniques for recommender systems. Computer, 2009, pages 30-37.

Neighborhood Models vs Latent Factor Models



Koren, Y.; Bell, R.; and Volinsky, C. Matrix factorization techniques for recommender systems. Computer, 2009, pages 30-37.