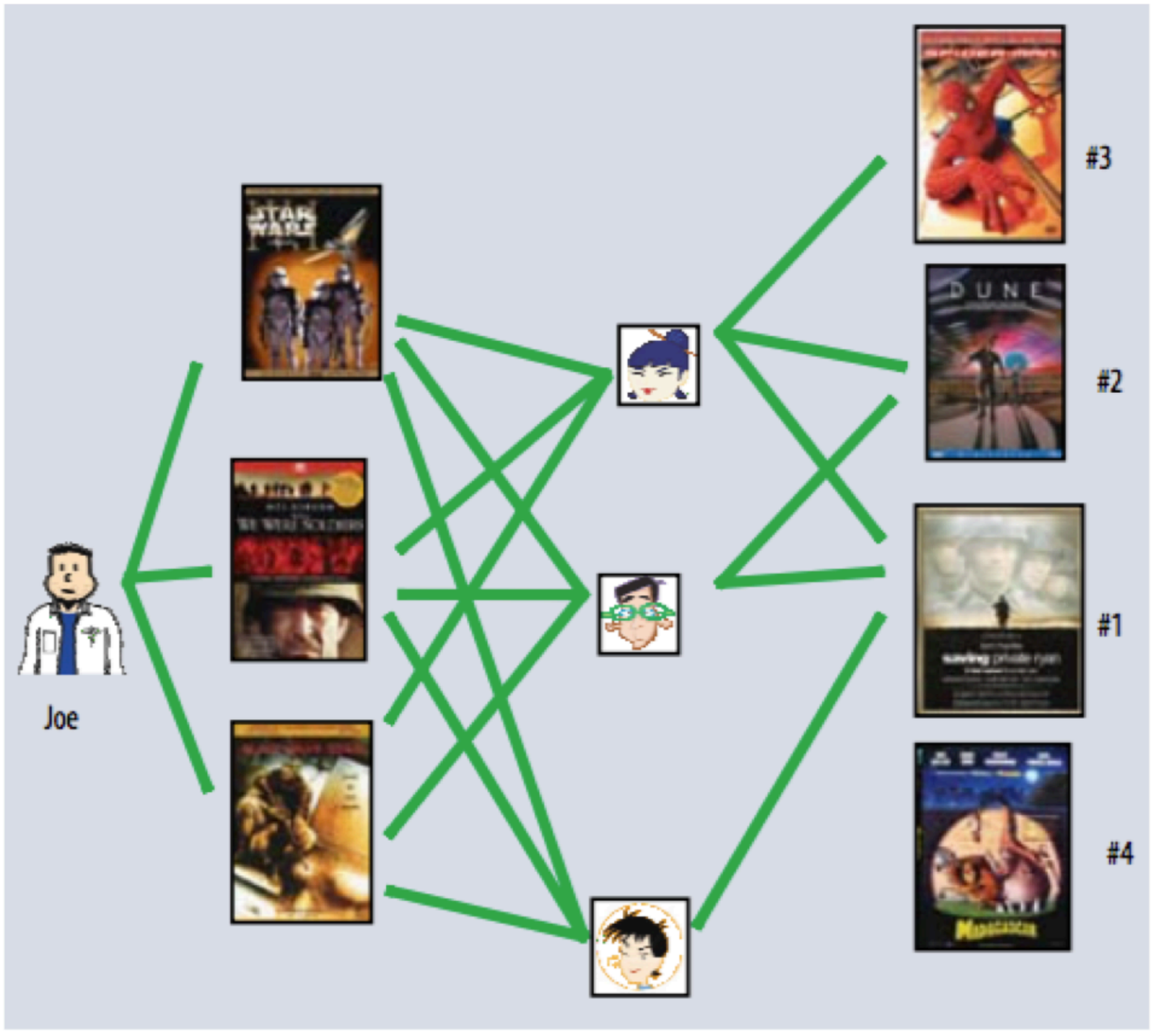
Machine Learning

Lecture 13　Latent Factor Model

1. Neighborhood Methods



In the figure, assume that a green line indicates the movie was **watched**

Algorithm:

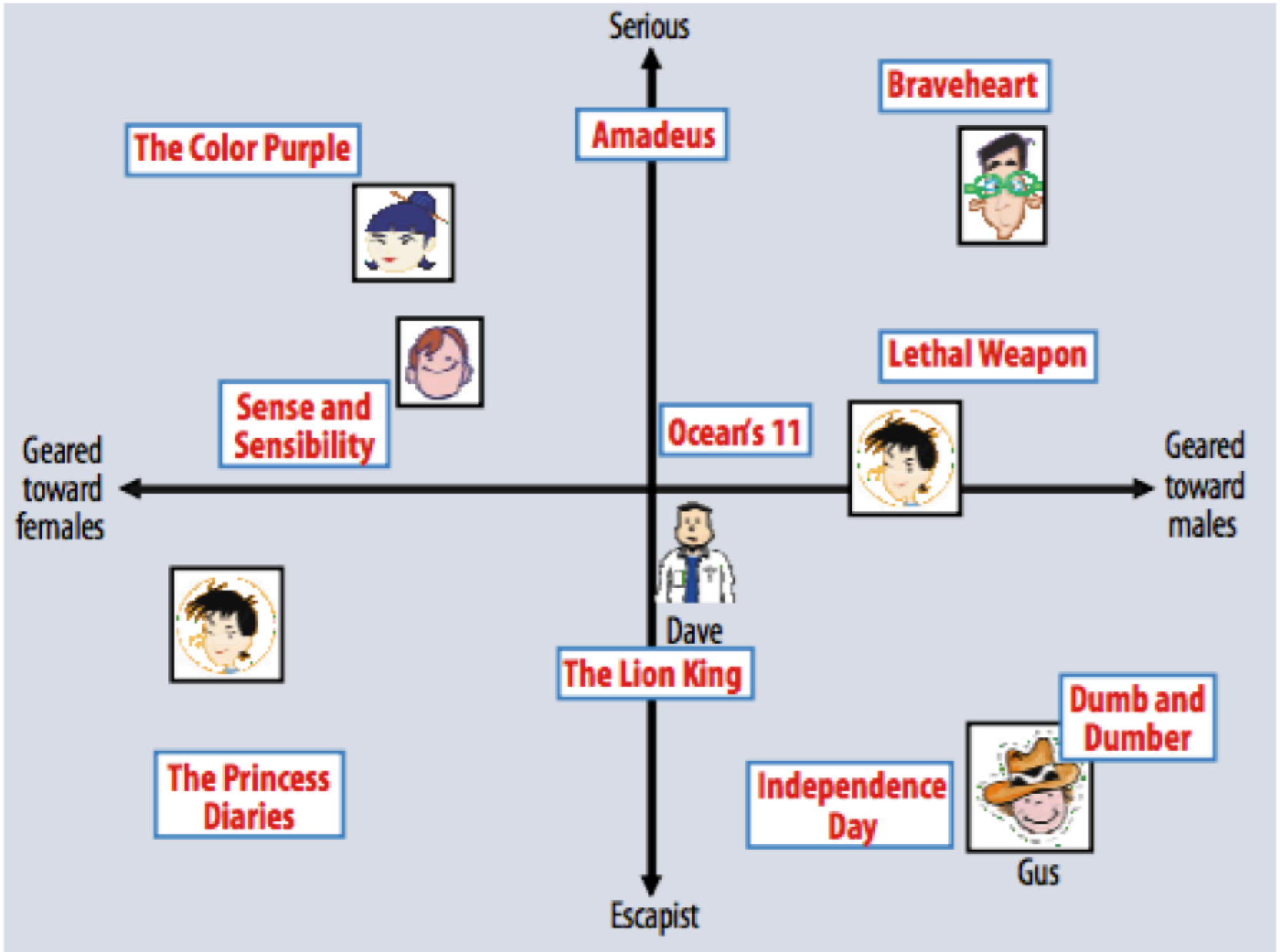
1. **Find neighbors** based on similarity of movie preferences
2. **Recommend** movies that those neighbors watched

Figures from Koren et al. (2009)

Source: https://www.cs.cmu.edu/~mgormley/courses/10601-s17/slides/lecture25-mf.pdf

2. Latent Factor Methods

- Assume that both movies and users live in some **low-dimensional space** describing their properties
- **Recommend** a movie based on its **proximity** to the user in the latent space



Figures from Koren et al. (2009)

Source: https://www.cs.cmu.edu/~mgormley/courses/10601-s17/slides/lecture25-mf.pdf

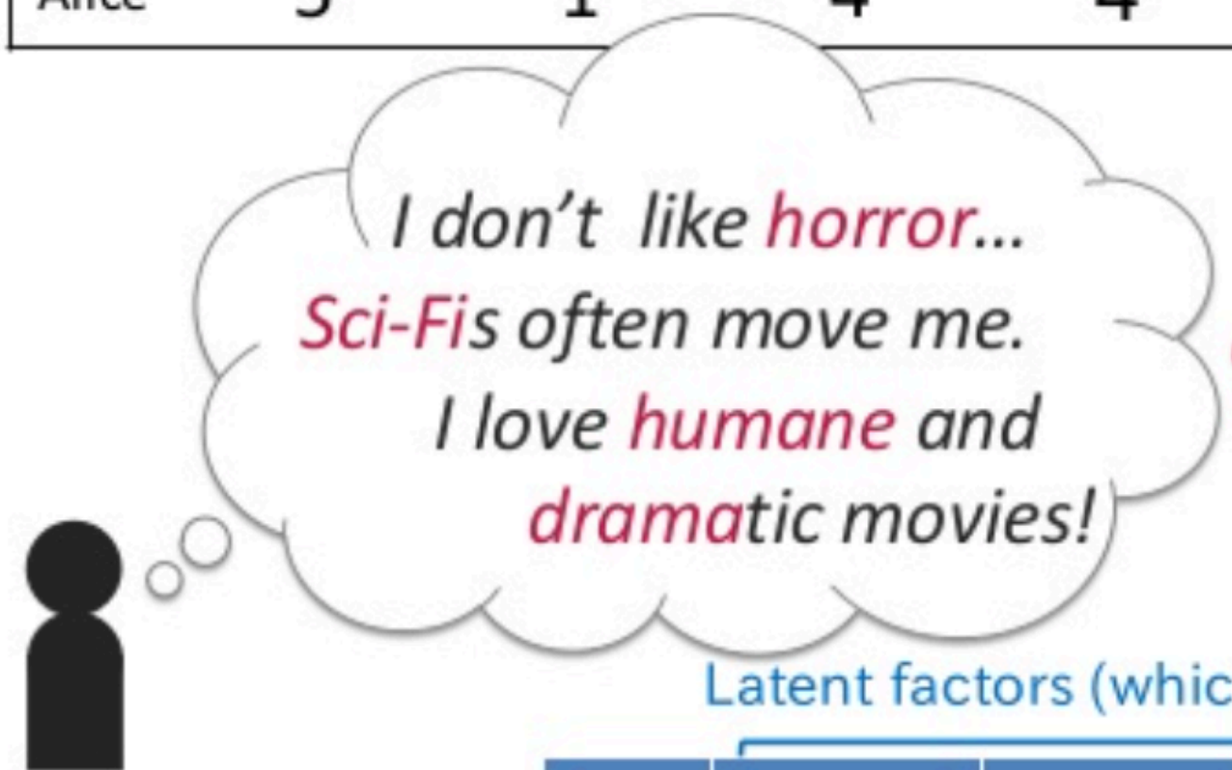
Netflix competition

Image ref: http://blogs.itmedia.co.jp/saito/2009/09/httpjournalmyco.html

Basic Idea

Assumes that latent factors exist in users/items

User	God father	Termin-ator	Money game	Titanic	Back to the future	…	X-men
Alice	5	1	4	4	3	…	2



Latent factors (which cannot be observed)

User	Horror	Sci-Fi	Humanity	Drama	…
Alice	0.3	2.1	6.1	4.7	…

Source: Yusuke Yamamoto

Basic Idea

Assumes that latent factors exist in users/items

User	God father	Termin-ator	Money game	Titanic	Back to the future	…	X-men
Alice	5	1	4	4	3	…	2

Latent factors (which cannot be observed)

User	Horror	Sci-Fi	Humanity	Drama	…
God father	2.3	0.4	4.9	5.7	…

Source: Yusuke Yamamoto

Basic Idea

Assumes that rating scores derive from latent factors of users and items

User	God father	Termin-ator	Money game	Titanic	Back to the future	…	X-men
Alice	5	1	4	4	3	…	2

User	Horror	Sci-Fi	Humanity	Drama	…
God father	2.3	0.4	4.9	5.7	…

×

User	Horror	Sci-Fi	Humanity	Drama	…
Alice	0.3	2.1	6.1	4.7	…

Source: Yusuke Yamamoto

Ideas

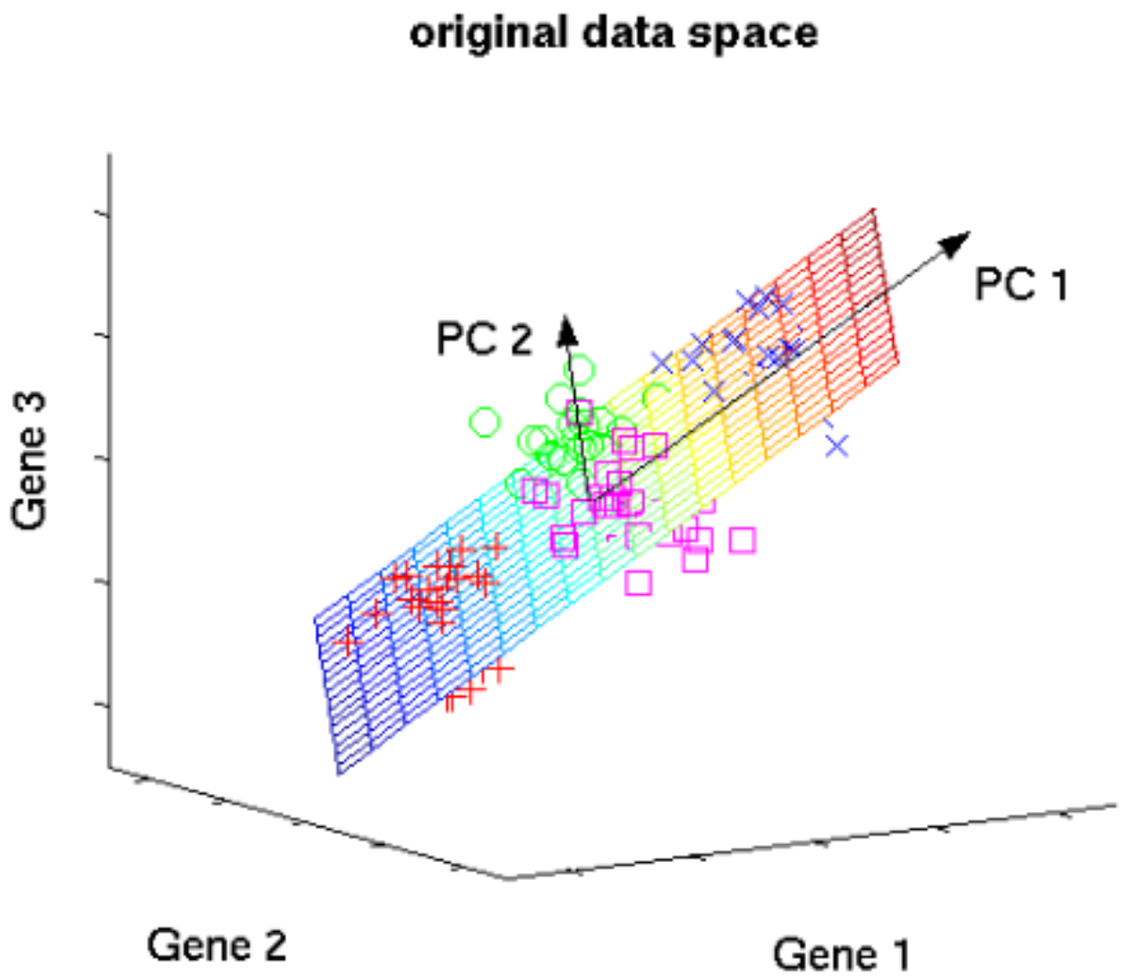
- 考慮評分矩陣R中的所有的值都被填滿的簡單情況。
- 秩 (rank) 為 $k << \min\{m,n\}$的任何 $m \times n$ 矩陣 R 可以用以下rank-k因子的乘積表示：

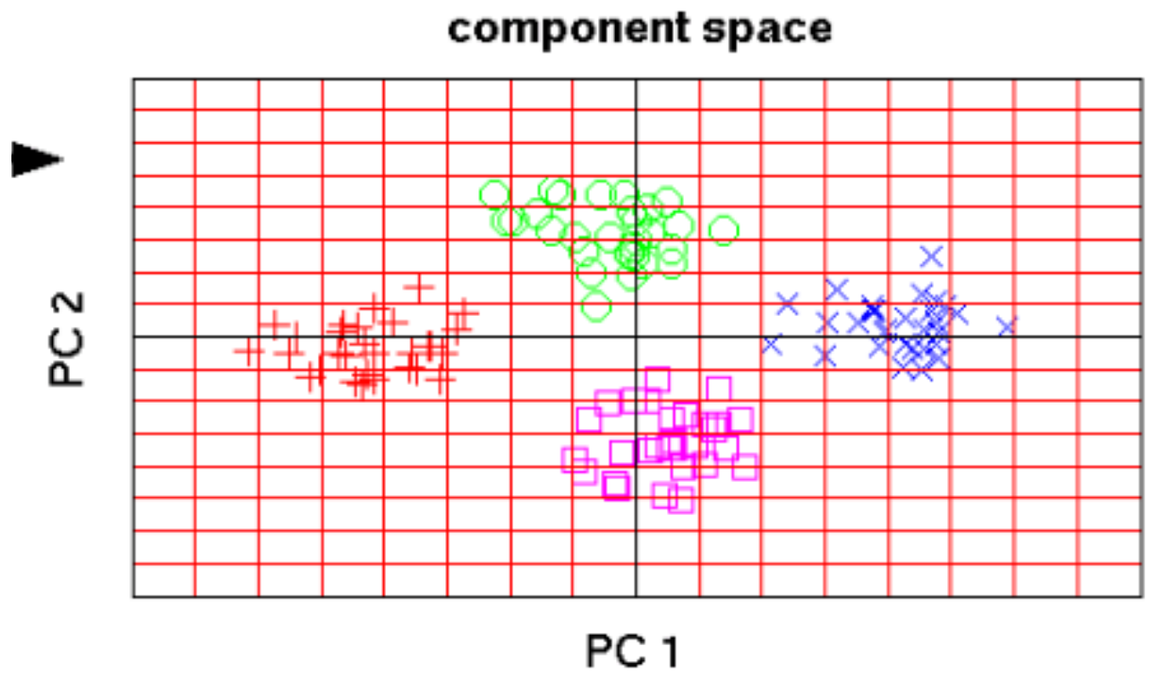
$$R = UV^T$$

其中 U 是一個 $m \times k$ 矩陣，V 是一個 $n \times k$ 矩陣

- 當 rank(R) > k 時，通常可以近似地表示為rank-k因子的乘積：$R \approx UV^T$

Dimension Reduction





Source: https://hackernoon.com/a-laymans-introduction-to-principal-components-2fca55c19fa0

Example

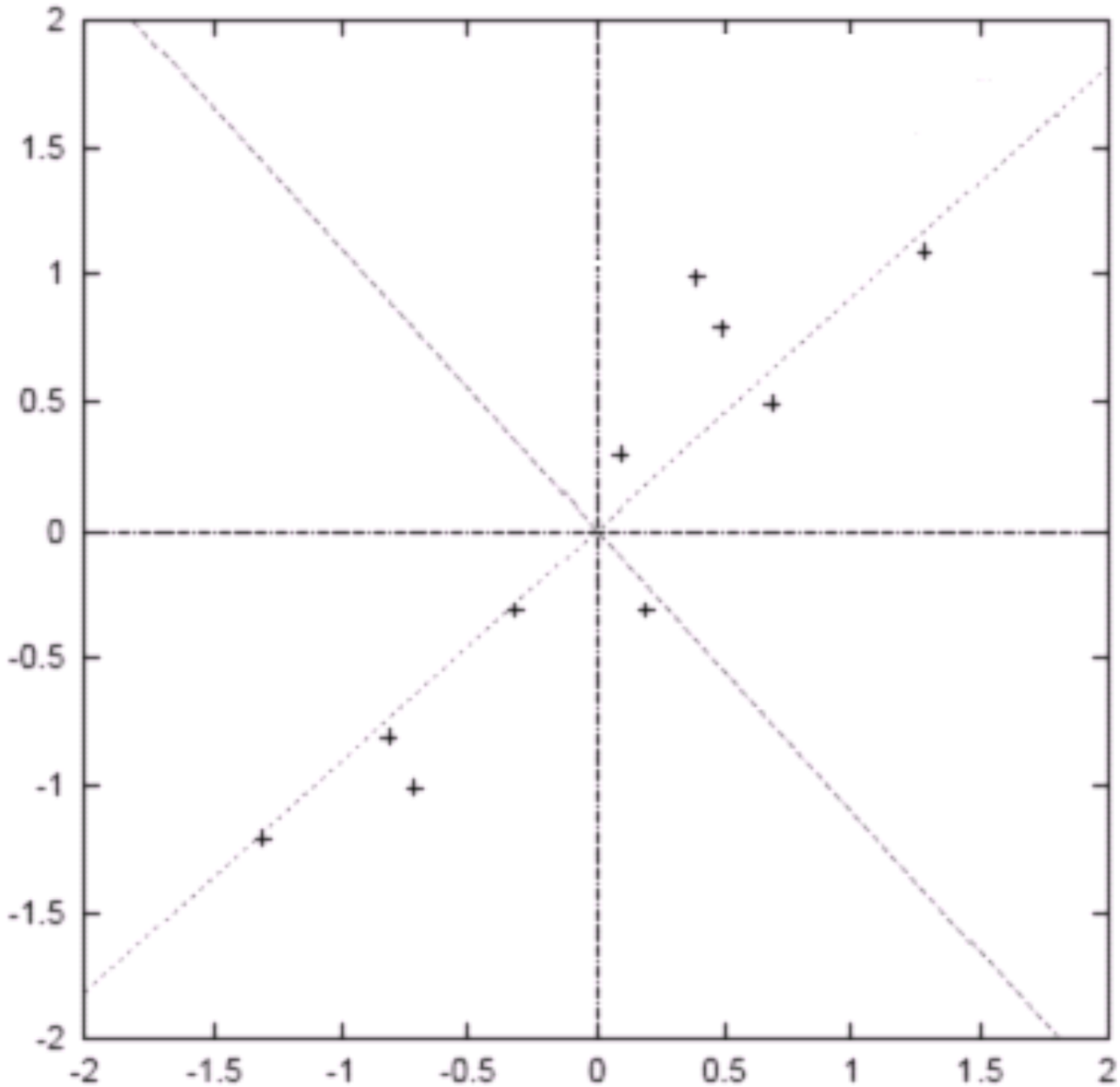
Get their **eigenvectors & eigenvalues**

$$\lambda_1 = 1.2840$$

$$v_1 = \begin{bmatrix} 0.6779 \\ 0.7352 \end{bmatrix}$$

$$\lambda_2 = 0.0491$$

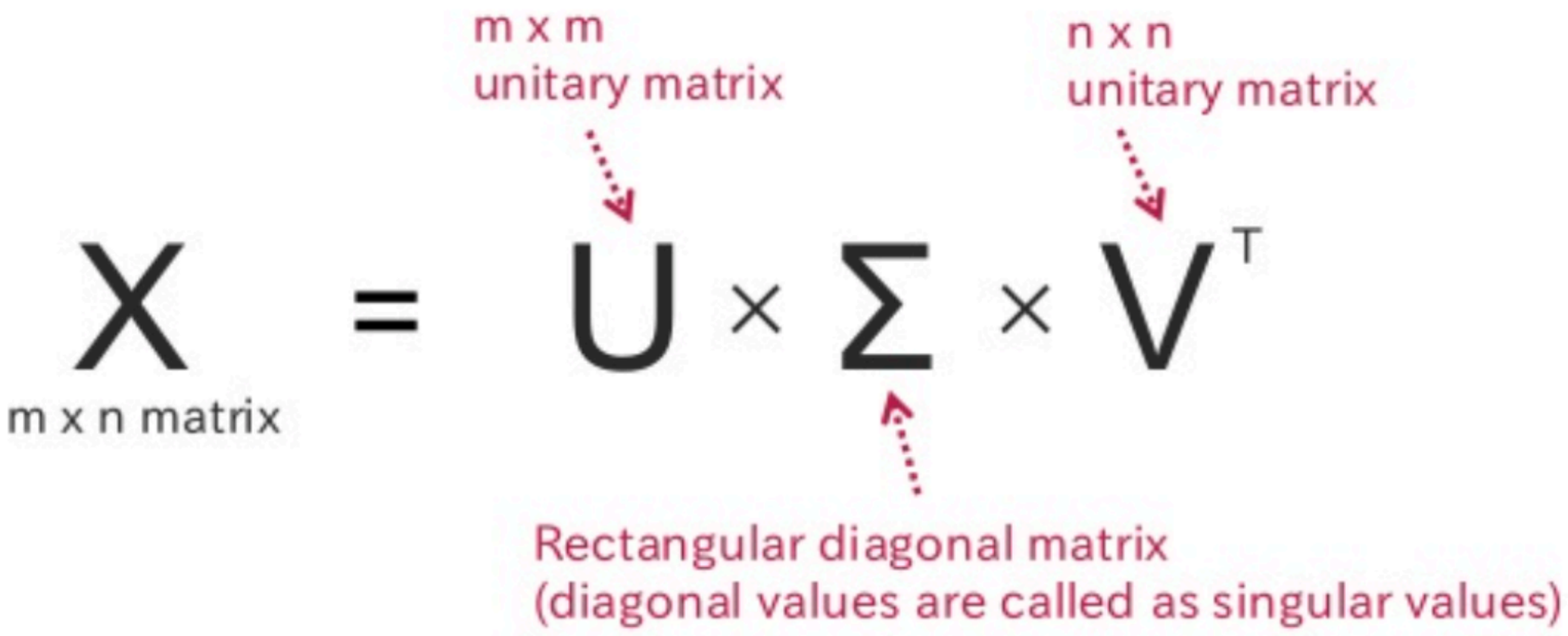
$$v_2 = \begin{bmatrix} -0.7352 \\ 0.6779 \end{bmatrix}$$



SVD

SVD: singular value decomposition

- A famous linear algebra technique for matrix decomposition
- It is often used for dimensionality reduction
- SVD delivers essentially the same result as PCA does



Source: Yusuke Yamamoto

SVD

$$X = U \times \Sigma \times V^{T}$$

$$\begin{bmatrix} 1 & 3 & 3 & 3 & 0 \\ 2 & 4 & 2 & 2 & 4 \\ 1 & 3 & 3 & 5 & 1 \\ 4 & 5 & 2 & 3 & 3 \\ 1 & 1 & 5 & 2 & 1 \end{bmatrix}$$

$$U = \begin{bmatrix} -0.369 & -0.325 & 0.282 & 0.343 & 0.749 \\ -0.459 & 0.448 & -0.339 & -0.584 & 0.364 \\ -0.468 & -0.359 & 0.544 & -0.459 & -0.381 \\ -0.563 & 0.491 & 0.07 & 0.562 & -0.348 \\ -0.341 & -0.569 & -0.71 & 0.118 & -0.202 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 13.368 & 0 & 0 & 0 & 0 \\ 0 & 4.708 & 0 & 0 & 0 \\ 0 & 0 & 2.792 & 0 & 0 \\ 0 & 0 & 0 & 1.586 & 0 \\ 0 & 0 & 0 & 0 & 0.904 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.325 & 0.341 & -0.101 & 0.682 & -0.55 \\ -0.562 & 0.345 & 0.273 & 0.153 & 0.684 \\ -0.468 & -0.642 & -0.577 & 0.125 & 0.141 \\ -0.504 & -0.327 & 0.601 & -0.324 & -0.416 \\ -0.324 & 0.496 & -0.47 & -0.626 & -0.189 \end{bmatrix}$$

Source: Yusuke Yamamoto

Important features of SVD (1/4)

We can approximate a given matrix by focusing on the largest singular values and the vectors of U and V which correspond to the singular values

$$U=\begin{bmatrix} -0.369 & -0.325 & 0.282 & 0.343 & 0.749 \\ -0.459 & 0.448 & -0.339 & -0.584 & 0.364 \\ -0.468 & -0.359 & 0.544 & -0.459 & -0.381 \\ -0.563 & 0.491 & 0.07 & 0.562 & -0.348 \\ -0.341 & -0.569 & -0.71 & 0.118 & -0.202 \end{bmatrix}$$

$$\Sigma=\begin{bmatrix} 13.368 & 0 & 0 & 0 & 0 \\ 0 & 4.708 & 0 & 0 & 0 \\ 0 & 0 & 2.792 & 0 & 0 \\ & & & 1.586 & 0 \\ 0 & 0 & 0 & 0 & 0.904 \end{bmatrix}$$

Largest singular values

$$V=\begin{bmatrix} -0.325 & 0.341 & -0.101 & 0.682 & -0.55 \\ -0.562 & 0.345 & 0.273 & 0.153 & 0.684 \\ -0.468 & -0.642 & -0.577 & 0.125 & 0.141 \\ -0.504 & -0.327 & 0.601 & -0.324 & -0.416 \\ -0.324 & 0.496 & -0.47 & -0.626 & -0.189 \end{bmatrix}$$

Source: Yusuke Yamamoto

Important features of SVD (2/4)

We can approximate a given matrix by focusing on the largest singular values and the vectors of U and V which correspond to the singular values

$$U = \begin{bmatrix} -0.369 & -0.325 \\ -0.459 & 0.448 \\ -0.468 & -0.359 \\ -0.563 & 0.491 \\ -0.341 & -0.569 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 13.368 & 0 \\ 0 & 4.708 \end{bmatrix}$$

Ignore unimportant values!

$$V = \begin{bmatrix} -0.325 & 0.341 \\ -0.562 & 0.345 \\ -0.468 & -0.642 \\ -0.504 & -0.327 \\ -0.324 & 0.496 \end{bmatrix}$$

Source: Yusuke Yamamoto

Important features of SVD (3/4)

We can approximate a given matrix by focusing on the largest singular values and the vectors of U and V which correspond to the singular values

$$U_2 = \begin{bmatrix} -0.369 & -0.325 \\ -0.459 & 0.448 \\ -0.468 & -0.359 \\ -0.563 & 0.491 \\ -0.341 & -0.569 \end{bmatrix}$$

$$\Sigma_2 = \begin{bmatrix} 13.368 & 0 \\ 0 & 4.708 \end{bmatrix}$$

$$V_2 = \begin{bmatrix} -0.325 & 0.341 \\ -0.562 & 0.345 \\ -0.468 & -0.642 \\ -0.504 & -0.327 \\ -0.324 & 0.496 \end{bmatrix}$$

Source: Yusuke Yamamoto

Important features of SVD (4/4)

We can approximate a given matrix by focusing on the largest singular values and the vectors of U and V which correspond to the singular values

$$U_2 \times \Sigma_2 \times V_2^T$$

$$= \begin{bmatrix} 1.08 & 2.24 & 3.29 & 2.98 & 0.84 \\ 2.72 & 4.17 & 1.52 & 2.41 & 3.04 \\ 1.46 & 2.93 & 4.02 & 3.71 & 1.19 \\ 3.24 & 5.03 & 2.04 & 3.04 & 3.59 \\ 0.57 & 1.64 & 3.86 & 3.18 & 0.15 \end{bmatrix} \approx \begin{bmatrix} 1 & 3 & 3 & 3 & 0 \\ 2 & 4 & 2 & 2 & 4 \\ 1 & 3 & 3 & 5 & 1 \\ 4 & 5 & 2 & 3 & 3 \\ 1 & 1 & 5 & 2 & 1 \end{bmatrix} = X$$

Source: Yusuke Yamamoto

SVD

Regarded as zero

	Item1	Item2	Item3	Item4	Item5
Alice	1	3	3	3	?
User1	2	4	2	2	4
User2	1	3	3	5	1
User3	4	5	2	3	3
User4	1	1	5	2	1

To matrix

$$\begin{bmatrix} 1 & 3 & 3 & 3 & 0 \\ 2 & 4 & 2 & 2 & 4 \\ 1 & 3 & 3 & 5 & 1 \\ 4 & 5 & 2 & 3 & 3 \\ 1 & 1 & 5 & 2 & 1 \end{bmatrix} = X$$

Source: Yusuke Yamamoto

Step 1 Run SVD

$$X = U \times \Sigma \times V^{T}$$

Step 2

Focus on important features

$$U_2 \times \Sigma_2 \times V_2^{T}$$

Step 3 Multiply three matrix

$$\begin{bmatrix} 1.08 & 2.24 & 3.29 & 2.98 & \mathbf{0.84} \\ 2.72 & 4.17 & 1.52 & 2.41 & 3.04 \\ 1.46 & 2.93 & 4.02 & 3.71 & 1.19 \\ 3.24 & 5.03 & 2.04 & 3.04 & 3.59 \\ 0.57 & 1.64 & 3.86 & 3.18 & 0.15 \end{bmatrix}$$

Step 4

Check the values which were zero before running SVD

Source: Yusuke Yamamoto

SVD

- Suppose R is a fully specified matrix, $R \approx UV^T$
- Optimization problem: find U and V

Minimize $J = \frac{1}{2}\|R - UV^T\|^2$ Forbenius norm

subject to:

Columns of U are mutually orthogonal

Columns of V are mutually orthogonal

SVD

- Suppose R is a fully specified matrix, $R \approx UV^T$
- Optimization problem: find U and V

$$\text{Minimize} \quad J = \frac{1}{2}\|R - UV^T\|^2 = \frac{1}{2}\sum_{i,j}\left(R_{ij} - \sum_{s=1}^{k} u_{is}v_{js}\right)^2$$

subject to:

Columns of U are mutually orthogonal

Columns of V are mutually orthogonal

Issues

	Ghibli 1	Ghibli 2	Ghibli 3	Ghibli 4
User1	5			
User2	3	4		
User3	2		1	
User4		5		4
User5				5

Zero replacement

$$\begin{bmatrix} 5 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 0 & 5 & 0 & 4 \\ 0 & 0 & 0 & 5 \end{bmatrix}$$

Run SVD

$$U_2, \Sigma_2, V_2$$

Multiply matrix

$$\begin{bmatrix} 3.53 & 1.88 & 0.16 & -0.26 \\ 3.62 & 4.04 & 0.13 & 2.17 \\ 1.44 & 0.76 & 0.07 & -0.12 \\ 2.76 & 5.41 & 0.06 & 4.34 \\ 1.67 & 5.97 & 0 & 5.74 \\ -0.26 & 2.91 & -0.06 & 3.52 \end{bmatrix}$$

Source: Yusuke Yamamoto

Example from: http://smrmkt.hatenablog.jp/entry/2014/08/23/211555

SVD

- The set of observed user-item pairs $S = \{(i, j) : r_{i,j} \text{ is observed}\}$
- Optimization problem: find U and V

$$\text{Minimize} \quad J = \frac{1}{2} \sum_{(i,j) \in S} \left(R_{ij} - \sum_{s=1}^{k} u_{is} v_{js} \right)^2$$

subject to:

Columns of U are mutually orthogonal

Columns of V are mutually orthogonal

Unconstrained matrix factorization

- $R \approx UV^T$
- The set of observed user-item pairs $S = \{(i, j) : r_{i,j} \text{ is observed}\}$
- Optimization problem: find U and V

$$\text{Minimize} \quad J = \frac{1}{2} \sum_{(i,j) \in S} \left(R_{ij} - \sum_{s=1}^{k} u_{is} v_{js} \right)^2$$

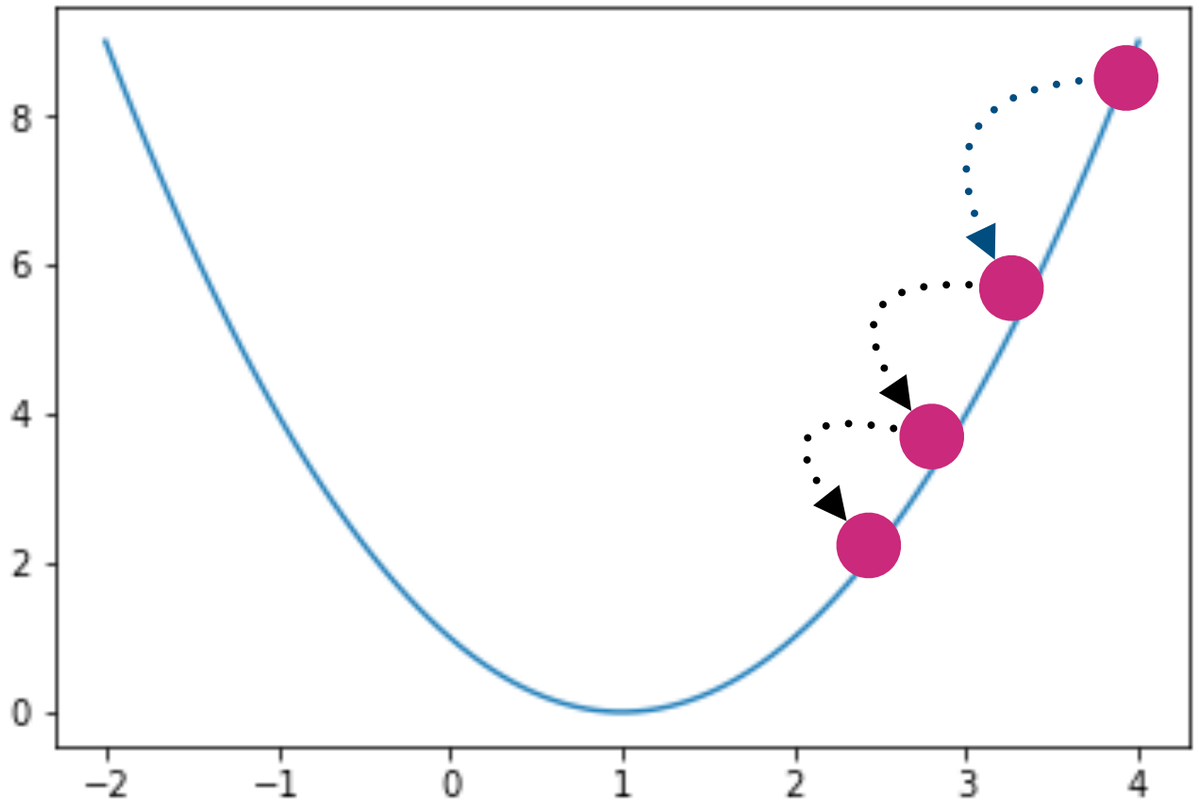
subject to:

No constraints on U and V

Gradient descent (梯度下降法)

Suppose $\eta = 0.1$

$$g(\theta) = (\theta - 1)^2$$



$$\frac{d}{d\theta} g(\theta) = 2\theta - 2$$

Gradient descent (梯度下降法)

$$\theta := \theta - \eta \frac{d}{d\theta} g(\theta)$$

$$= \theta - \eta(2\theta - 2)$$

$$\theta := 4 - 0.1(2 \cdot 4 - 2) = 4 - 0.6 = 3.4$$

$$\theta := 3.4 - 0.1(2 \cdot 3.4 - 2) = 3.4 - 0.48 = 2.92$$

$$\theta := 2.92 - 0.1(2 \cdot 2.92 - 2) = 2.92 - 0.384 = 2.536$$

Gradient descent (梯度下降法)

- Cost function

$$E(\theta_0, \theta_1, \theta_2) = \frac{1}{2} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

- Gradient descent

$$\theta_0 := \theta_0 - \eta \frac{\partial E}{\partial \theta_0}$$

$$\theta_1 := \theta_1 - \eta \frac{\partial E}{\partial \theta_1}$$

$$\theta_2 := \theta_2 - \eta \frac{\partial E}{\partial \theta_2}$$

Gradient descent

Algorithm GD(Ratings Matrix: R, Learning Rate: α)
begin
  Randomly initialize matrices U and V;
  $S = \{(i, j) : r_{ij}$ is observed$\}$;
  while not(convergence) **do**
  begin
    Compute each error $e_{ij} \in S$ as the observed entries of $R - UV^T$;
    for each user-component pair (i, q) **do** $u_{iq}^+ \Leftarrow u_{iq} + \alpha \cdot \sum_{j:(i,j)\in S} e_{ij} \cdot v_{jq}$;
    for each item-component pair (j, q) **do** $v_{jq}^+ \Leftarrow v_{jq} + \alpha \cdot \sum_{i:(i,j)\in S} e_{ij} \cdot u_{iq}$;
    for each user-component pair (i, q) **do** $u_{iq} \Leftarrow u_{iq}^+$;
    for each item-component pair (j, q) **do** $v_{jq} \Leftarrow v_{jq}^+$;
    Check convergence condition;
  end
end

Recommender systems the textbook by Charu C. Aggarwal

Stochastic gradient descent

Algorithm *SGD*(Ratings Matrix: R, Learning Rate: α)
begin
  Randomly initialize matrices U and V;
  $S = \{(i, j) : r_{ij}$ is observed$\}$;
  while not(convergence) **do**
  begin
    Randomly shuffle observed entries in S;
    for each $(i, j) \in S$ in shuffled order **do**
    begin
      $e_{ij} \Leftarrow r_{ij} - \sum_{s=1}^{k} u_{is} v_{js}$;
      for each $q \in \{1 \ldots k\}$ **do** $u_{iq}^{+} \Leftarrow u_{iq} + \alpha \cdot e_{ij} \cdot v_{jq}$;
      for each $q \in \{1 \ldots k\}$ **do** $v_{jq}^{+} \Leftarrow v_{jq} + \alpha \cdot e_{ij} \cdot u_{iq}$;
      for each $q \in \{1 \ldots k\}$ **do** $u_{iq} = u_{iq}^{+}$ and $v_{jq} = v_{jq}^{+}$;
    end
    Check convergence condition;
  end
end

Recommender systems the textbook by Charu C. Aggarwal

Regularization

- The set of observed user-item pairs $S = \{(i, j) : r_{i,j} \text{ is observed}\}$ is small, which can cause **overfitting**.

- Optimization problem: find U and V

$$\text{Minimize} \quad J = \frac{1}{2} \sum_{(i,j) \in S} \left(R_{ij} - \sum_{s=1}^{k} u_{is} v_{js} \right)^2 + \frac{\lambda}{2} \|U\|^2 + \frac{\lambda}{2} \|V\|^2$$

subject to:

No constraints on U and V

Incorporating User and Item Biases

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	7	6	7	4	5	4
User 2	6	7	?	4	3	4
User 3	?	3	3	1	1	?
User 4	1	2	2	3	3	4
User 5	1	?	1	2	3	3

相似度 2：Pearson 相關係數

Q. 計算 User 1 和 3 的 Pearson 相關係數 Pearson(1, 3)

Step 2. 利用這些商品集計算 Pearson 相關係數

$$\textbf{Pearson}(1,3)$$

$$= \frac{(6-5.5)\times(3-2)+(7-5.5)\times(3-2)+(4-5.5)\times(1-2)+(5-5.5)\times(1-2)}{\sqrt{(6-5.5)^2+(7-5.5)^2+(4-5.5)^2+(5-5.5)^2}\cdot\sqrt{(3-2)^2+(3-2)^2+(1-2)^2+(1-2)^2}}$$

$$= 0.894$$

	Item 2	Item 3	Item 4	Item 5		平均
User 1	6	7	4	5	−	5.5
User 2						
User 3	3	3	1	1	−	2

→

	Item 2	Item 3	Item 4	Item 5
User 1	0.5	1.5	-1.5	-0.5
User 2				
User 3	1	1	-1	-1

預測

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	平均	Pearson(*i*, 3)
User 1	7	6	7	4	5	4	5.5	0.894
User 2	6	7	?	4	3	4	4.8	0.939
User 3	?	3	3	1	1	?	2	1.0

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in P_u(j)} \mathbf{Sim}(u, v) \cdot (r_{vj-\mu_v})}{\sum_{v \in P_u(j)} |\mathbf{Sim}(u, v)|}$$

$$\hat{r}_{31} = 2 + \frac{1.5 \times 0.894 + 1.2 \times 0.939}{0.894 + 0.939} \approx 3.35$$

$$\hat{r}_{36} = 2 + \frac{-1.5 \times 0.894 + 0.8 \times 0.939}{0.894 + 0.939} \approx 0.86$$

Incorporating User and Item Biases

- o_i : the general bias of user *i* to rate items
- p_j : the bias in the ratings of item *j*
- The main change to the original latent factor model is that a part of the (i, j)th rating is explained by $o_i + p_j$ and the remainder by the (i, j)th entry of UV^T
- The predicted value of the rating of entry (i, j) is

$$\hat{r}_{ij} = o_i + p_j + \sum_{s=1}^{k} u_{is} \cdot v_{js}$$

Incorporating User and Item Biases

- Optimization problem: find U and V

Minimize

$$J = \frac{1}{2} \sum_{(i,j) \in S} \left(r_{ij} - \hat{r}_{ij} \right)^2 + \frac{\lambda}{2} \left(\|U\|^2 + \|V\|^2 + \sum_i o_i^2 + \sum_j p_j^2 \right)$$

$$= \frac{1}{2} \sum_{(i,j) \in S} \left(r_{ij} - o_i - p_j - \sum_{s=1}^{k} u_{is} v_{js} \right)^2 + \frac{\lambda}{2} \left(\sum_{i=1}^{m} \sum_{s=1^k} u_{is}^2 + \sum_{j=1}^{n} \sum_{s=1^k} v_{js}^2 + \sum_i o_i^2 + \sum_j p_j^2 \right)$$

Surprise

- Surprise (Simple Python RecommendatIon System Engine)

- *built-in* datasets (Movielens, Jester)

- Provide various ready-to-use prediction algorithms such as baseline algorithms, neighborhood methods, matrix factorization-based, and many others. Also, various similarity measures (cosine, MSD, pearson...) are built-in.

Python

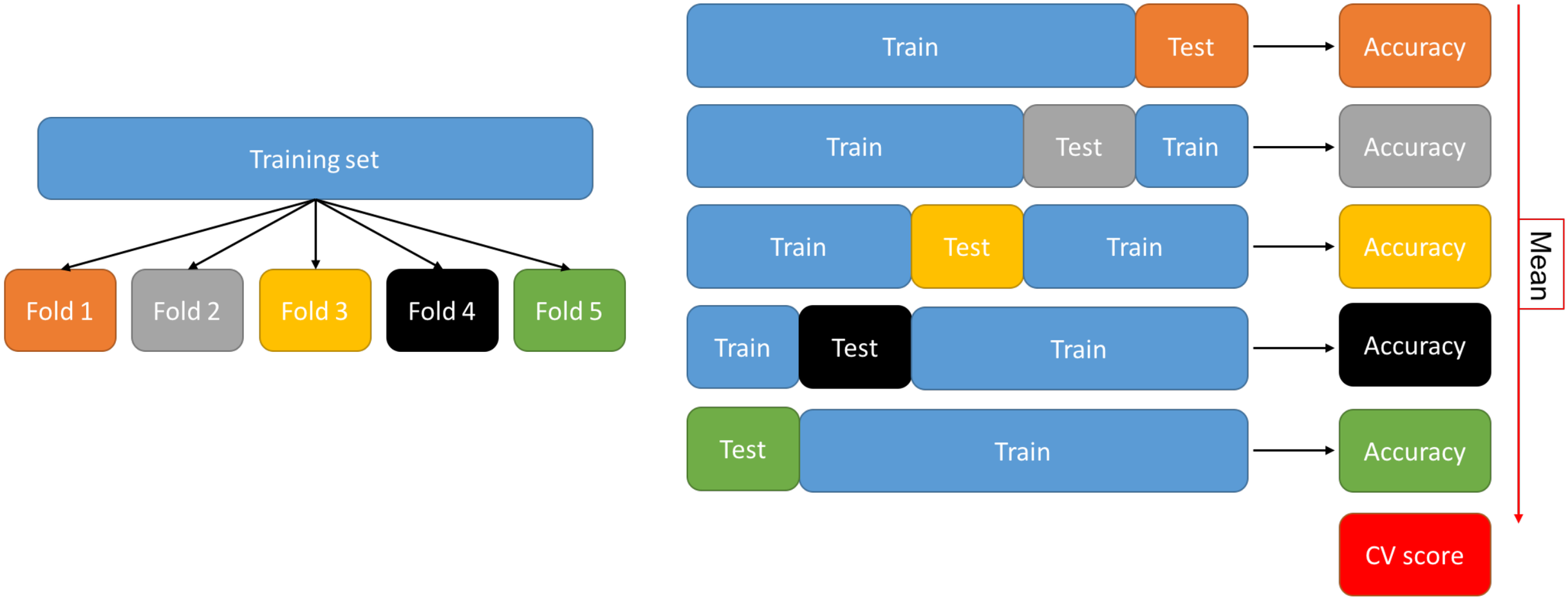
```python
from surprise import SVD
from surprise import Dataset
from surprise.model_selection import cross_validate

# Load the movielens-100k dataset (download it if needed).
data = Dataset.load_builtin('ml-100k')

# Use the famous SVD algorithm.
algo = SVD()

# Run 5-fold cross-validation and print results.
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

cross-validation



Source: https://aldro61.github.io/microbiome-summer-school-2017/sections/basics/

Python

```python
from surprise import SVD
from surprise import Dataset
from surprise.model_selection import cross_validate

# Load the movielens-100k dataset (download it if needed).
data = Dataset.load_builtin('ml-100k')

# Use the famous SVD algorithm.
algo = SVD()

# Run 5-fold cross-validation and print results.
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9411  0.9292  0.9295  0.9365  0.9430  0.9359  0.0057
MAE (testset)     0.7426  0.7321  0.7339  0.7390  0.7426  0.7381  0.0044
Fit time          6.00    6.06    6.20    6.02    6.08    6.07    0.07
Test time         0.32    0.23    0.17    0.23    0.27    0.25    0.05
```

http://surpriselib.com

MAE & RMSE

- MAE (Mean Absolute Error, 平均絶對値誤差)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- RMSE (Root Mean Square Error)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

MAE & RMSE

	$y - \hat{y}$	$\lvert y - \hat{y} \rvert$	$(y - \hat{y})^2$
1	-5	5	25
2	-3	3	9
3	1	1	1
4	5	5	25
5	8	8	64

MAE: 4.4

RMSE: 4.98

MAE & RMSE

	$y - \hat{y}$	$\lvert y - \hat{y} \rvert$	$(y - \hat{y})^2$
1	-5	5	25
2	-3	3	9
3	1	1	1
4	5	5	25
5	8	8	64

MAE: 4.4

RMSE: 4.98

	$y - \hat{y}$	$\lvert y - \hat{y} \rvert$	$(y - \hat{y})^2$
1	-5	5	25
2	-3	3	9
3	1	1	1
4	5	5	25
5	100	100	10000

MAE: 22.8

RMSE: 44.86

Matrix Factorization

- 優點：
  - 解決稀疏性問題
    - 空間複雜度低 （不需儲存龐大的相似度矩陣）
- 缺點：冷啟動問題、解釋性較差