

Recommendation System :

협업 필터링을 중심으로

김민환

카이스트 전산학과
응용알고리즘 연구실

제 8회 ROSAEC 워크샵

Table of Contents

- Recommendation in Everyday Life
- Kinds of Recommendation System
 - Content-based Approach
 - Collaborative Filtering
- TF-IDF and Similarity Measures
- Matrix Factorization
- Comparison and Conclusion

Recommendation in Everyday Life

- 언젠가 떠올렸던 의문
 - 죽을 때까지 몇 권의 책을 읽을 수 있을까?
 - 일주일에 1권이라면, 1년에 52권, 앞으로 50년 동안 같은 페이스로 읽는다면..
 - 평생 읽어도 2500여권에 지나지 않는다!



Recommendation in Everyday Life

- 정보 홍수의 시대
 - 즐길 수 있는 다양한 작품들과 구매할 수 있는 온갖 상품들
 - 기호에 맞는 취사 선택이 중요
- 온라인에서의 추천 시스템

amazon.com: Why is this recommended for you?

amazon.com

Help | Close window

Recommended for You

Body by Science
Our Price: **\$9.99**
Used & new from \$9.99

Rate this item
 ☆☆☆☆☆
 I own it
 Not interested

See all buying options

Because you purchased...

The Black Swan: Second Edition: The Impact of the Highly Improbable: With a new section: "On Robustness and Fragility" (Kindle Edition)

☆☆☆☆☆
 This was a gift
 Don't use for recommendations

Help | Close window

NETFLIX

Carine Curry | Your Account & Help

Watch Instantly | Browse DVDs | Your Queue | Suggestions For You

Home | Genres | New Arrivals | Starz Play | Instantly to your TV | Help

You recently watched: National Geographic: Guns, Germs a... ☆☆☆☆☆ | **Play Next**

Critically-acclaimed Foreign Dramas

Based on your interest in...

Let the Right One In
 AMELIE
 Play
 ☆☆☆☆☆

Mongol
 Play
 ☆☆☆☆☆

Europa Europa
 Play
 ☆☆☆☆☆

Nowhere in Africa
 Play
 ☆☆☆☆☆

Thrillers

Your taste preferences created this row.

The Boondock Saints
 Play
 ☆☆☆☆☆

The Horsemen
 Play
 ☆☆☆☆☆

Law Abiding Citizen
 Play
 ☆☆☆☆☆

Armored
 Play
 ☆☆☆☆☆

- 비즈니스에서의 중요성

- Netflix : 대여되는 영화의 2/3가 추천으로부터 발생
- Google News : 38% 이상의 조회가 추천에 의해 발생
- Amazon : 판매의 35% 가 추천으로부터 발생
- 곧, 소비자의 잠재 욕구를 발견하여 판매 기회를 발굴

- Netflix Prize (~2009)

Netflix에서 주관하는 경연대회로,

영화 선호도를 가장 잘 예측하는 협업 필터링 알고리즘에 수상 (US\$1,000,000)

- 본질적으로 객체 사이의 잠재적인 관계를 파악

- 추천할 수 있는 것들
 - News Articles, Tags, Online Mates, Restaurants
 - Courses in e-Learning, Movies, Books, Various Goods
 - Research Papers, Citations ..



Kinds of Recommendation System

- Content-based

- 사용자 혹은 상품의 내용을 이용
- Natural Language Processing, Information Retrieval 등 분야의 기법을 사용
- 예를 들면,
사용자가 영화 '스파이더맨'을 보았다면 RS는 '스파이더맨'에 대한 설명 (heroes, adventure, ..) 을 참고 하여 비슷한 영화를 찾아 추천

- Collaborative Filtering

- 사용자의 평가 내역을 이용
- User-based : 비슷한 사용자를 찾는다
Item-based : 비슷한 항목을 찾는다
- 유사 정도를 측정하는 척도 이용, Matrix Factorization 기법을 사용
- 예를 들면,
사용자 A가 스파이더맨(5), 로미오와줄리엣(1), 헐크(4)의 평가를 하였고,
사용자 B는 스파이더맨(4), 로미오와줄리엣(1) 의 평가를 하였다면
사용자 A와 B는 비슷한 취향을 가진 것으로 생각할 수 있고
사용자 B에게 영화 '헐크'를 추천해 줄 수 있다.

TF-IDF

- TF-IDF : Term Frequency – Inverse Document Frequency
- Content-based 접근에서 사용할 수 있는 표현
- 문서를 수학적 표현으로 나타내고자

- 일상에서나 온라인에서 수집할 수 있는 문서들은 비정형
 - 문서의 길이, 사용되는 단어, ..

- 문서 데이터를 일관된 표현으로 나타내고 싶다
 - 자주 사용하는 표현: 벡터 형식
 - 벡터 형식의 표현을 위해,
Dictionary를 준비 (모든 문서를 훑어 한번 이상 사용되는 단어를 수집)
 - 벡터의 값을 정하는 방법은 다양
 - 단어의 존재 여부 – Binary Representation, 단어 등장 횟수
 - Mutual Information, Entropy, Information Gain, ..

TF-IDF

- 예) word count

Grand Master Turing once dreamed that he was a machine.
When he awoke he exclaimed:

“I don't know whether I am Turing dreaming that I am a machine, or a machine dreaming that I am Turing!”

- The Tao of Programming, 2.3

	grand	master	Turing	I	am	..
Doc_1	1	1	3	4	3	..

TF-IDF

- Keyword-based Vector Space Model with basic TF-IDF weighting

- $D = \{d_1, d_2, \dots\}, T = \{t_1, t_2, \dots\}, d_j = \{w_{1j}, w_{2j}, \dots\}$

- $TF - IDF(t_k, d_j) = TF(t_k, d_j) \times \log \frac{N}{n_k}$

- N = 전체 문서 개수
 n_k = 단어 t_k 가 적어도 한번 이상 등장한 문서 개수

- $TF(t_k, d_j) = \frac{f_{k,j}}{\max_z f_{z,j}}$

- $f_{k,j}$ = 문서 d_j 에서 단어 t_k 등장 빈도 수
 $\max_z f_{z,j}$ = 문서 d_j 에서 가장 많이 등장한 단어의 빈도 수

- $w_{k,j} = \frac{TF-IDF(t_k, d_j)}{\sqrt{\sum_{s=1}^{|T|} TF-IDF(t_s, d_j)^2}} \rightarrow \text{weight 값이 } [0,1] \text{ 사이에 위치하도록}$

TF-IDF

- TF-IDF Weighting 특징
 - 한 문서 안에서 여러 번 등장하는 단어는 적게 등장하는 단어보다 중요 (TF)
 - 전체적으로 드문 단어는 자주 등장하는 단어보다 중요하게 생각 (IDF)
 - 짧은 문서가 긴 문서보다 더 중요 (Normalization)
- 이와 같은 표현을 통해,
두 문서 사이의 유사 정도를 측정할 수 있다

$$\bullet \text{sim}(d_i, d_j) = \frac{\sum_k w_{k,i} \times w_{k,j}}{\sqrt{\sum_k w_{k,i}^2} \times \sqrt{\sum_k w_{k,j}^2}}$$

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

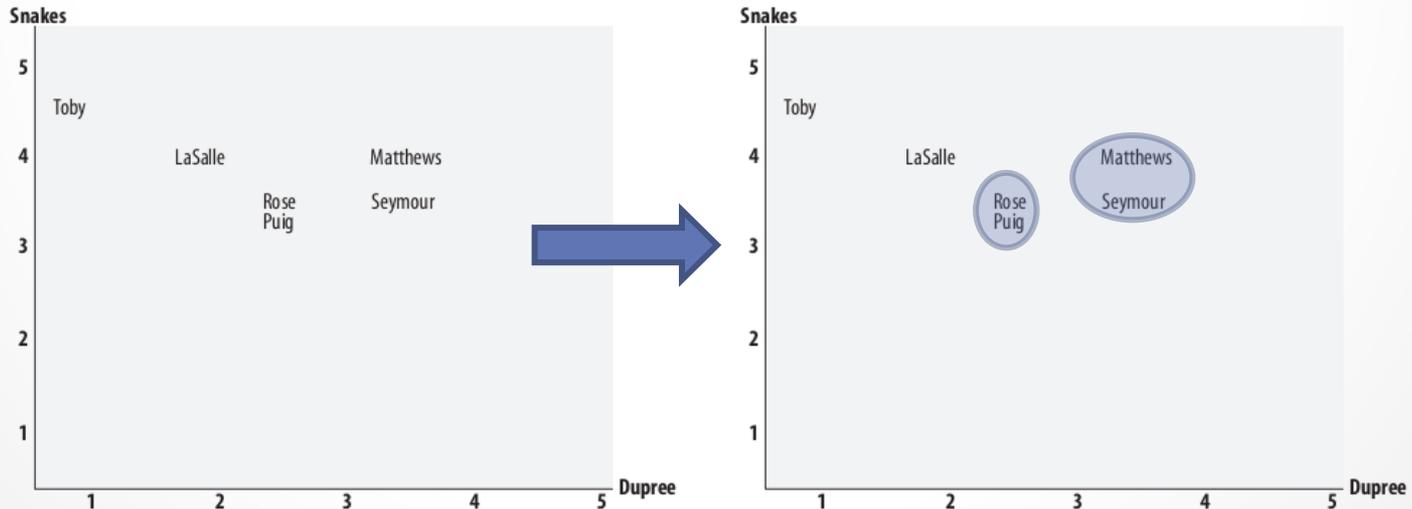

Similarity Measures

- Hooray! 이제 Content-based 방법의 원리를 알았다!
 - 영화 스파이더맨 설명 = {0.32, 0.12, .. }, 배트맨 설명 = ..
항목 간의 유사도를 측정하여 비슷한 항목을 추천
- 유사성은 추천 시스템과 댈래야 댈 수 없는 것
- Collaborative Filtering 역시 유사성을 고려한다
- 아주 간단한 유사성 척도에 기반한 Collaborative Filtering 소개
 - 참고: 집단지성 프로그래밍, 토비 세가란, 한빛 미디어
 - Euclidean Distance, Pearson Correlation

Similarity Measures

- 데이터셋
 - 영화 평가 정보
- Euclidean Distance
 - Snakes on a Plane 과 You, Me and Dupree를 평가한 사람 가운데 취향이 비슷한 사람은?

```
# A dictionary of movie critics and their ratings of a small
# set of movies
critics={'Lisa Rose': {'Lady in the Water': 2.5, 'Snakes on a Plane': 3.5,
    'Just My Luck': 3.0, 'Superman Returns': 3.5, 'You, Me and Dupree': 2.5,
    'The Night Listener': 3.0},
    'Gene Seymour': {'Lady in the Water': 3.0, 'Snakes on a Plane': 3.5,
    'Just My Luck': 1.5, 'Superman Returns': 5.0, 'The Night Listener': 3.0,
    'You, Me and Dupree': 3.5},
    'Michael Phillips': {'Lady in the Water': 2.5, 'Snakes on a Plane': 3.0,
    'Superman Returns': 3.5, 'The Night Listener': 4.0},
    'Claudia Puig': {'Snakes on a Plane': 3.5, 'Just My Luck': 3.0,
    'The Night Listener': 4.5, 'Superman Returns': 4.0,
    'You, Me and Dupree': 2.5},
    'Mick LaSalle': {'Lady in the Water': 3.0, 'Snakes on a Plane': 4.0,
    'Just My Luck': 2.0, 'Superman Returns': 3.0, 'The Night Listener': 3.0,
    'You, Me and Dupree': 2.0},
    'Jack Matthews': {'Lady in the Water': 3.0, 'Snakes on a Plane': 4.0,
    'The Night Listener': 3.0, 'Superman Returns': 5.0, 'You, Me and Dupree': 3.5},
    'Toby': {'Snakes on a Plane':4.5,'You, Me and Dupree':1.0,'Superman Returns':4.0}}
```



Similarity Measures

- Pearson Correlation Coefficient

- 코드의 모습을 보면..

두 지표가 얼마나 비슷한 경향으로 움직이는가!

```
# Add up all the preferences
sum1=sum([prefs[p1][it] for it in si])
sum2=sum([prefs[p2][it] for it in si])

# Sum up the squares
sum1Sq=sum([pow(prefs[p1][it],2) for it in si])
sum2Sq=sum([pow(prefs[p2][it],2) for it in si])

# Sum up the products
pSum=sum([prefs[p1][it]*prefs[p2][it] for it in si])

# Calculate Pearson score
num=pSum-(sum1*sum2/n)
den=sqrt((sum1Sq-pow(sum1,2)/n)*(sum2Sq-pow(sum2,2)/n))
if den==0: return 0

r=num/den

return r
```

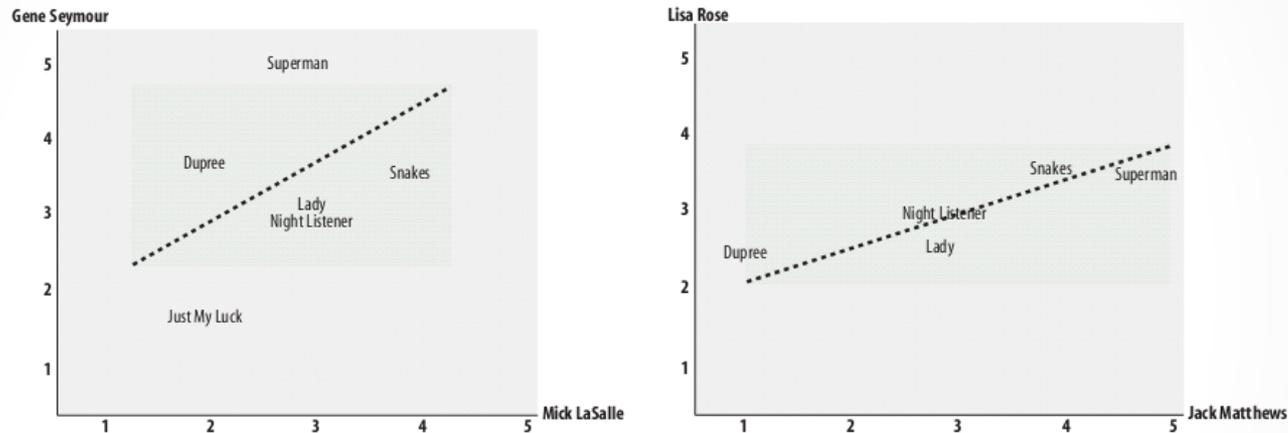
$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X\sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X\sigma_Y}$$

$$r = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}}$$

$$r_{xy} = \frac{\sum x_i y_i - n \bar{x} \bar{y}}{(n-1) s_x s_y} = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n \sum x_i^2 - (\sum x_i)^2} \sqrt{n \sum y_i^2 - (\sum y_i)^2}}$$

Similarity Measures

- Pearson Correlation Coefficient
 - 두 평론가 사이의 평가 경향을 살펴보면..



- Euclidean Distance에서는, 두 평론가의 경향은 비슷하지만 한쪽이 전체적으로 적은 점수를 주는 경우 찾아내기 어렵다
- 이 밖에도 Jaccard Index, Manhattan Distance 등 다양한 Similarity Metric 존재

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$$d_1(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\|_1 = \sum_{i=1}^n |p_i - q_i|$$

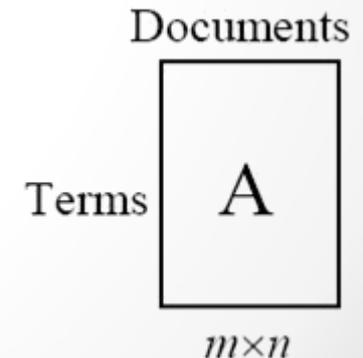
Matrix Factorization

- 다시 한번 Hooray! Collaborative Filtering 의 원리를 알았다!
 - 사용자들의 평가 내역을 이용하여, 유사한 사용자를 찾아낸다.
같은 상품에 대해 평가 내역이 비슷한 두 사람은, 비슷한 취향을 가질 것이라는 가정 아래, 어느 한쪽이 아직 보지 못했지만 평이 좋은 항목을 추천해 준다.
- 하지만 새로운 접근 방법이 있었으니..
수학적으로 보다 심오한 방법, 행렬 분해 (Matrix Factorization)
- Singular Value Decomposition

$$\begin{pmatrix}
 & \begin{matrix} X \\ x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{matrix} \\
 & m \times n
 \end{pmatrix}
 =
 \begin{pmatrix}
 & \begin{matrix} U \\ u_{11} & \dots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{matrix} \\
 & m \times r
 \end{pmatrix}
 \begin{pmatrix}
 & \begin{matrix} S \\ s_{11} & 0 & \dots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{matrix} \\
 & r \times r
 \end{pmatrix}
 \begin{pmatrix}
 & \begin{matrix} V^T \\ v_{11} & \dots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{matrix} \\
 & r \times n
 \end{pmatrix}$$

Matrix Factorization

- 계산 방법
 - $X = USV^T$
 - XX^T 의 고유벡터들을 구해,
고유값이 큰 순서에 따라 왼쪽에서부터 차례로 쌓는다 (Column Vector)
이렇게 구해진 행렬을 Orthonormal 행렬로 바꾼 것이 U
 - 유사하게 $X^T X$ 에 대하여 위의 과정의 수행한 것이 V
 - U와 V의 음이 아닌 고유값은 항상 같게 되는데,
이 고유값에 제곱근을 취하여 큰 순서대로 대각에 위치시킨 것이 S
- 하지만 이것만으로는 쉽게 와 닿질 않는다.
단어와 문서에 대한 예를 생각해 보자.
 - 각각의 행이 특정 단어, 열이 문서를 표현하는 행렬
 - 이때 문서를 나타내는 하나의 벡터는 앞서 살펴본 TF-IDF,
단어 빈도 등 유사한 방법을 통해 표현

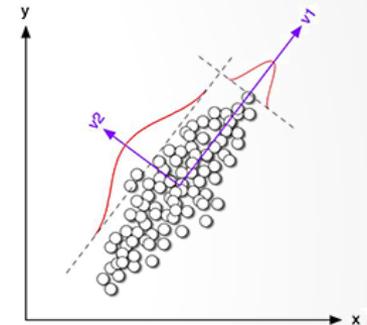


Matrix Factorization

- Latent Semantic Analysis

- XX^T = gives the correlation between the **terms** over the **documents**
 - $X^T X$ = gives the correlation between the **documents** over the **terms**

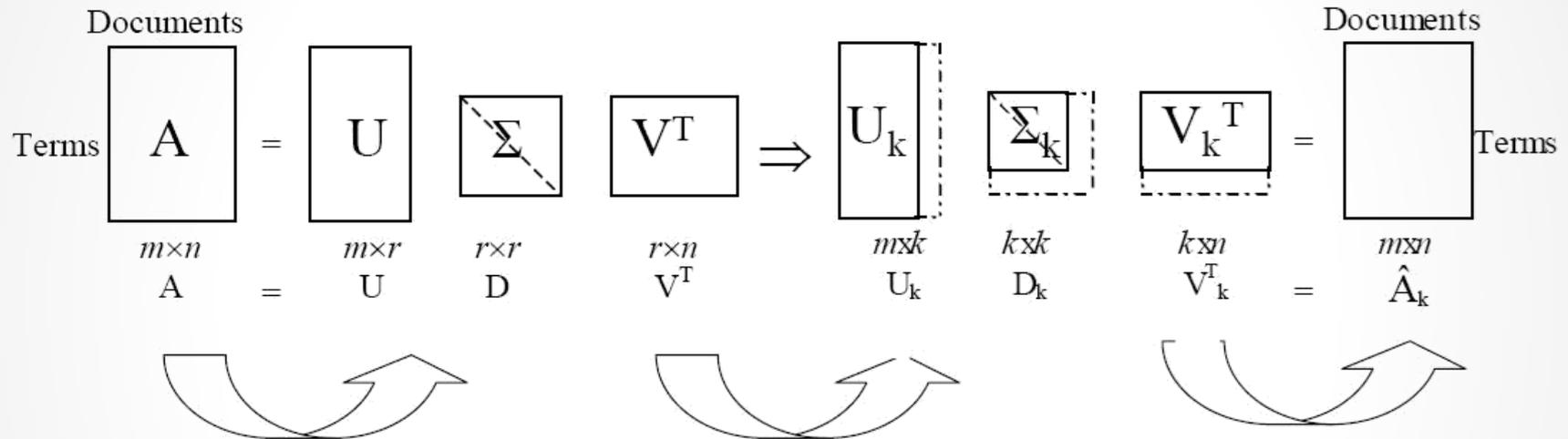
$$\begin{array}{c} X \\ (d_j) \\ \downarrow \\ \begin{bmatrix} x_{1,1} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,n} \end{bmatrix} \\ (t_i^T) \rightarrow \end{array} = (t_i^T) \rightarrow \begin{array}{c} U \\ \left[\begin{bmatrix} u_1 \\ \vdots \\ u_l \end{bmatrix} \right] \end{array} \cdot \begin{array}{c} \Sigma \\ \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} \end{array} \cdot \begin{array}{c} V^T \\ (d_j) \\ \downarrow \\ \begin{bmatrix} v_1 \\ \vdots \\ v_l \end{bmatrix} \end{array}$$



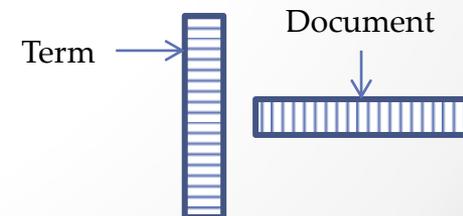
- 고유값이 큰 순서대로 고유벡터를 나열한다는 것은 주어진 데이터를 가장 잘 분별해내는 순서대로 나열한다는 것
 - 보통 전체 고유벡터를 취하지 않고 그보다 적은 k개의 고유벡터를 취함
 - 데이터 분별하는데 별로 도움이 되지 않는 차원은 무시하겠다는 이야기
 - 차원을 줄임으로써 보다 컴팩트 한 표현, 노이즈 제거, 비슷한 건 더 비슷하게 다른 건 더 다르게

Matrix Factorization

- Latent Semantic Analysis



- 분해된 U, V 행렬은 단어, 문서를 보다 분명하게 구별 유사 단어를 찾거나, 유사 문서를 찾을 수 있다
- 차원을 줄인 U, V의 곱은 원래 행렬을 근사
- 자료를 구분 짓는 잠재 요소를 발견



Matrix Factorization

- Matrix Factorization in Collaborative Filtering

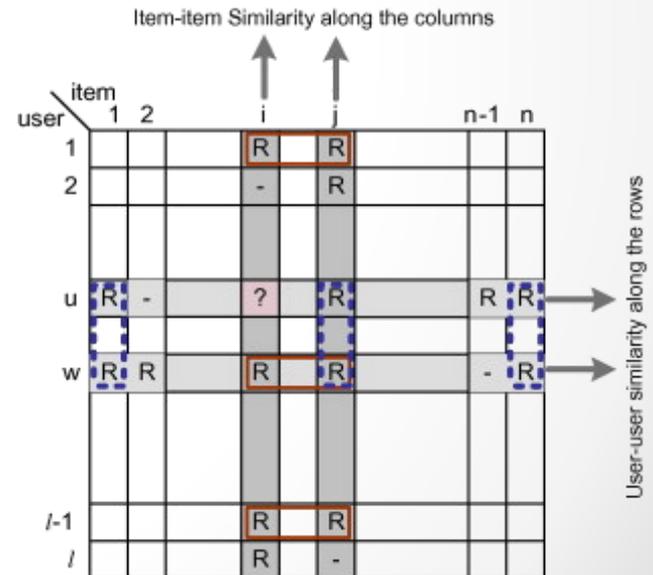
- Utility Matrix

상품
↓

		<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	<i>e</i>	<i>f</i>	<i>g</i>	<i>h</i>
사용자 →	<i>A</i>	4	5		5	1		3	2
	<i>B</i>		3	4	3	1	2	1	
	<i>C</i>	2		1	③		4	5	3

↓
상품에 대한 평가

- Utility Matrix를 이용하여 사용자-사용자, 상품-상품 간의 유사성을 비교해 볼 수 있다
- Utility Matrix에 행렬 분해 기법을 적용한다면..

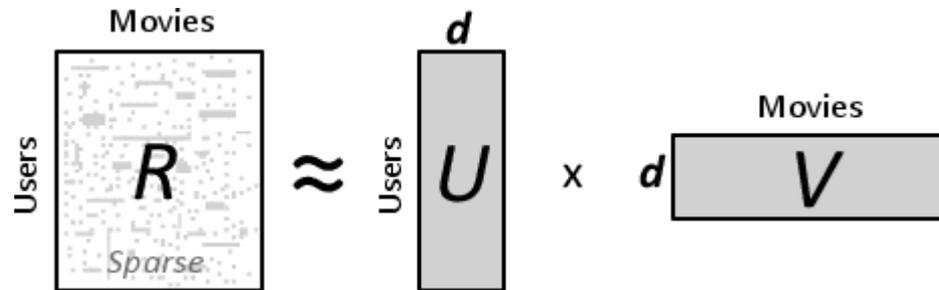


Matrix Factorization

- Matrix Factorization in Collaborative Filtering

- Toy Example

정말 이런 기법이 유효하게 동작하는가?



- 다음과 같은, 영화에 대한 가상의 평가를 이용하여 수행

	슈퍼맨	배트맨	색계	아멜리에
병곤	1	2	8	10
용섭	10	7	8	3
우상	8	9	9	2
성수	4	5	9	7

Matrix Factorization

- Matrix Factorization in Collaborative Filtering

- Toy Example

```
matR = {{1., 2., 8., 10.},
        {10., 7., 8., 3.},
        {8., 9., 9., 2.},
        {4., 5., 9., 7.}}
```

- 두 개의 고유값을
이용하도록 SVD를 수행,

```
{matU, matD, matV} = SingularValueDecomposition[matR, 2]
```

- 다시 U,S,V 곱으로
원래 행렬을 근사한 결과

```
{matU // MatrixForm, matD // MatrixForm, matV // MatrixForm}
```

$$\left\{ \begin{pmatrix} -0.399611 & 0.76066 \\ -0.544757 & -0.397447 \\ -0.554105 & -0.418556 \\ -0.486332 & 0.297057 \end{pmatrix}, \begin{pmatrix} 26.1407 & 0. \\ 0. & 10.1655 \end{pmatrix}, \begin{pmatrix} -0.467676 & -0.528657 \\ -0.460246 & -0.348487 \\ -0.647224 & 0.178271 \\ -0.388013 & 0.753192 \end{pmatrix} \right\}$$

```
matU . matD . Transpose[matV]
```

```
{{0.797568, 2.1131, 8.13944, 9.87724},
 {8.79574, 7.96201, 8.49642, 2.48235},
 {9.02346, 8.14925, 8.61632, 2.41554},
 {4.34918, 4.79879, 8.76652, 7.20725}}
```

Matrix Factorization

- Matrix Factorization in Collaborative Filtering
 - Toy Example
 - 두 개의 차원으로 데이터를 표현
 - 원래의 표현보다 유사성을 잘 나타냄

```
In[56]:= (matUser = matU.matD) // MatrixForm
Out[56]/MatrixForm=
  ( -10.4461  7.73246
    -14.2403 -4.04023
    -14.4847 -4.25482
    -12.713   3.01972 )
```

```
In[57]:= CosineDistance[Normalize[matR[[2]]], Normalize[matR[[3]]]]
```

```
Out[57]= 0.0219707
```

```
In[58]:= CosineDistance[Normalize[matUser[[2]]], Normalize[matUser[[3]]]]
```

```
Out[58]= 0.0000428412
```

Cosine distance between two vectors:

```
In[1]:= CosineDistance[{a, b, c}, {x, y, z}]
```

$$\text{Out[1]} = 1 - \frac{a x + b y + c z}{\sqrt{\text{Abs}[a]^2 + \text{Abs}[b]^2 + \text{Abs}[c]^2} \sqrt{\text{Abs}[x]^2 + \text{Abs}[y]^2 + \text{Abs}[z]^2}}$$

Matrix Factorization

- Matrix Factorization in Collaborative Filtering
 - Toy Example - 사용자2는 배트맨을 좋아할까? 아멜리에를 좋아할까?

```
In[177]:= matR = {{1., 2., 8., 10.},
                 {10., 0., 8., 0.},
                 {8., 9., 9., 2.},
                 {4., 5., 9., 7.}};
```

빈 공간에 평균을 채워넣음

```
In[178]:= matR[[2, 2]] = matR[[2, 4]] = (10 + 8) / 2;
```

```
In[179]:= matR // MatrixForm
```

```
Out[179]//MatrixForm=
  ( 1.  2.  8. 10.
    10. 9  8.  9
     8.  9.  9.  2.
     4.  5.  9.  7. )
```

평균과 표준편차를 구함

```
In[180]:= matMean = {{1, 1, 1, 1} * Mean[matR[[1]]],
                    {1, 1, 1, 1} * Mean[matR[[2]]],
                    {1, 1, 1, 1} * Mean[matR[[3]]],
                    {1, 1, 1, 1} * Mean[matR[[4]]]};
```

```
In[181]:= lstSD = { StandardDeviation[matR[[1]]],
                   StandardDeviation[matR[[2]]],
                   StandardDeviation[matR[[3]]],
                   StandardDeviation[matR[[4]]]};
```

Z - Score로 Normalization

```
In[182]:= matStd = { (matR[[1]] - matMean[[1]]) / lstSD[[1]],
                    (matR[[2]] - matMean[[2]]) / lstSD[[2]],
                    (matR[[3]] - matMean[[3]]) / lstSD[[3]],
                    (matR[[4]] - matMean[[4]]) / lstSD[[4]]};
```

```
In[183]:= {matU, matD, matV} = SingularValueDecomposition[matStd, 2];
```

```
In[184]:= {matU // MatrixForm, matD // MatrixForm, matV // MatrixForm}
```

```
Out[184]= { ( 0.576409  0.285215
             -0.52096  0.439333
             -0.229438 -0.82483
             0.586269 -0.212827 ) , ( 2.84451  0.
             0.  1.93989 ) , ( -0.652018  0.11
                               -0.312929 -0.2
                               0.557928 -0.5
                               0.407019  0.7 ) }
```

```
In[185]:= matTemp = matU . matD . Transpose[matV];
```

```
In[186]:= (matApprox = matMean + (matTemp * lstSD)) // MatrixForm
```

```
Out[186]//MatrixForm=
  ( 0.815867  2.24803  7.89111  10.045
    9.87324  9.17075  7.92504  9.03097
    7.77973  9.29672  8.86974  2.05381
    3.72804  5.36634  8.83918  7.06644 )
```

배트맨을 더 좋아할 것 같다!

Comparison and Conclusion

- Content-based

- Pros

- 다른 사용자의 정보나 평가 내역이 필요하지 않다
 - 새로 추가 된 항목 (아직 평가되지 않은) 또한 추천 가능

- Cons

- 명시적으로 표현된 특징만을 다룰 수 있고, 질적인(Qualitative) 부분을 포착해내지 못한다 ↔ 협업 필터링의 경우, 항목의 평가는 사람의 주관에 따라 이뤄지므로 평가에 대한 질적인 부분을 포착해낼 수 있다
 - 추천하는 항목이 비슷한 장르에 머무르는 한계

- Collaborative Filtering

- Pros

- 잠재적인 특징들을 고려, 보다 다양한 범위의 추천 가능 (Cross-genre)
 - SVD 등의 Dimensionality Reduction 기법은 확장성 제공

- Cons

- 아직 평가되지 않은 항목은 추천 대상으로 발견되기 어렵다
 - 평가 내역이 준비되어야 취향을 비교할 수 있으므로 초기 사용자에게 대해선 믿음만한 추천을 하기 어렵다 (Data Sparsity, Cold Start Problem)
 - Gray Sheep. 평가가 일관적이지 않은 사용자는 도움이 되지 않는다

Comparison and Conclusion

- Conclusion

- 추천 시스템의 중요성이 날로 더해가고 있다
 - 효율적인 정보 소비 및 시간 활용을 위해
 - 비즈니스 수익 창출의 관점에서
- 크게 두 가지 접근 방법이 존재
 - Content-based
 - 상품 설명에 의존
 - 적은 자료로 추천 가능하나, 좁은 추천 범위
 - Collaborative Filtering
 - 여러 사람의 평가 정보를 활용
 - 다양한 범위의 추천 가능, 많은 자료가 필요
- 최근엔 두 가지 방법을 결합한 Hybrid 방식도 연구
- 유사성, 잠재 요소 등을 고려하는 점은 다른 분야에서도 참고 가능

References

- Recommender Systems Handbook, Ricci, F.; Rokach, L.; Shapira, B.; Kantor, P.B. (Eds.), Springer, 2011
- Mining of Massive Datasets, Anand Rajaraman; Jeffrey David Ullman, Cambridge, 2011
- http://en.wikipedia.org/wiki/Recommendation_system
- http://en.wikipedia.org/wiki/Singular_value_decomposition
- http://en.wikipedia.org/wiki/Latent_semantic_analysis
- http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient
- <http://reference.wolfram.com/mathematica/ref/SingularValueDecomposition.html>
- <http://reference.wolfram.com/mathematica/ref/CosineDistance.html>

Thank You!

