2018-10-06

추천 알고리즘

3기 김현세

Growth Hackers

CONTENTS

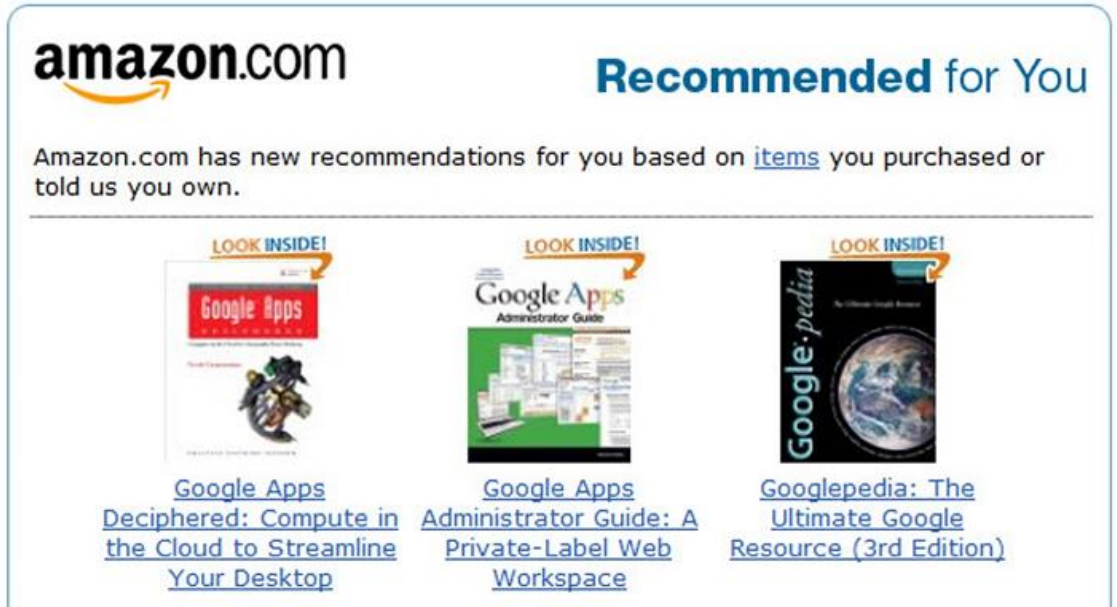
1. 추천 알고리즘
2. 협업 필터링
3. 차원 축소
4. 교차 타당성 검증
5. Surprise 패키지
6. QUEST

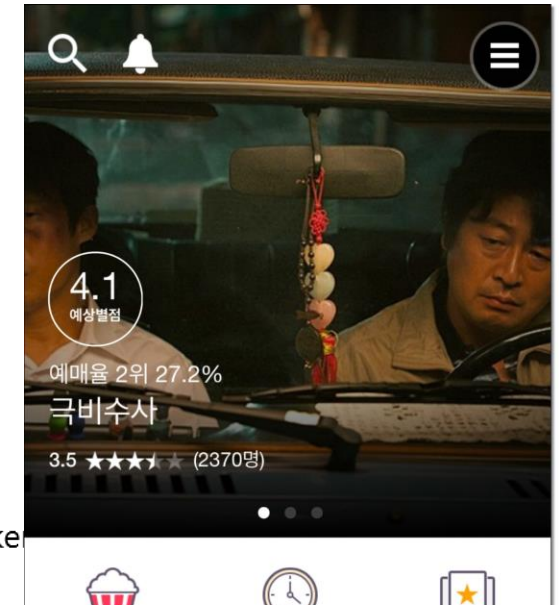
Growth Hackers

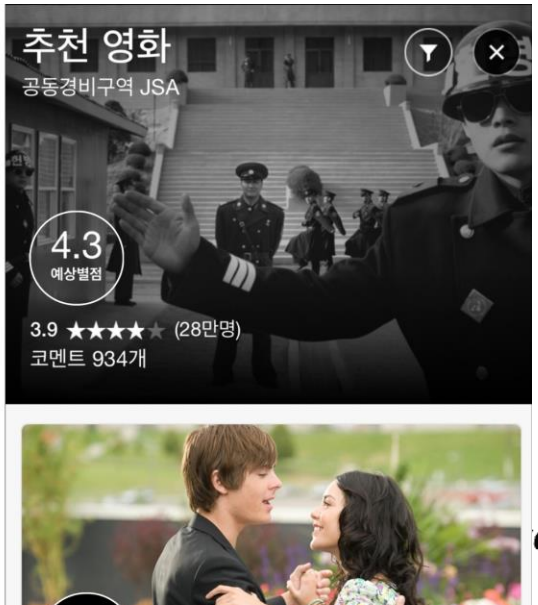
추천 알고리즘

사용자들의 이용 데이터를 기반으로 사용자에게 제품을 추천해주는 시스템

- Netflix : 대여되는 영화의 2/3가 추천으로부터 발생
- Google News : 38% 이상의 조회가 추천에 의해 발생
- Amazon : 판매의 35% 가 추천으로부터 발생







추천 알고리즘

알고리즘의 종류

기존의 추천 시스템

- **수작업을 이용한 추천**
- **인기도를 활용한 추천** (이것은 데이터가 없는 초기 사용자에 대해서는 유효)

- **협업 필터링**
- **내용 기반 필터링**
- **지식 기반 필터링**

Growth Hackers

추천 알고리즘

◆ 내용 기반 필터링

텍스트 중에서 형태소를 추려내고,
이 중에서도 핵심 키워드가 어떤 것인지를 분석하는 기술을 활용
이렇게 추려낸 키워드를 활용해서 소비자의 관심사항에 대한 분석이 가능.

◆ 지식 기반 필터링

특정 분야에 대한 전문가의 도움을 받아서
그 분야에 대한 전체적인 지식 구조를 만들어 활용
일반적으로 해당 분야에 대한 체계도를 만들어 활용한다.

협업 필터링

협업 필터링(Collaborative Filtering)

기본 아이디어

類類相從(유유상종)

1. 취향이 비슷한 사람의 집단이 존재한다는 가정
2. 추천의 대상이 되는 사람과 비슷한 집단을 찾는다
3. 이 집단의 사람들이 공통적으로 좋아하는 제품/서비스를 추천

협업 필터링

Utility Matrix

\ Movie User \	M1	M2	M3	M4	M5	Correlation with U1
U1	2	5	3			-
U2	4	4	3	5	1	0.19
U3	1	5	4		5	0.89
U4	3	5	3	2	5	0.94
U5	4	5	3	4		0.65
U3 & U4 평균	2	5	3.5	2	5	

KNN + α

1. 유사성을 계산해 Neighbor 분류 ➡ 2. Neighbor가 좋게 평가한 영화 찾기 ➡ 3. U1이 안 본 M5를 추천
   ▶ 3, 4를 Neighbor로 분류함

Growth Hackers

협업 필터링

사실 협업 필터링에는 두 가지 종류가 있습니다.

User Based CF → 본 세션에서는 이것을 기준으로 진행하겠습니다.

나와 가장 유사한 성향을 지닌 사람들을 기반으로 그 사람들이 구매한 아이템을 추천해주는 방식

사용자 간의 유사성을 기준으로 함 → 계산 多 정확도 高

Item Based CF → 현업에서 주로 사용되는 approach (ex. Amazon)

내가 선호하는 아이템을 기반으로 가장 유사한 성향의 아이템을 추천해주는 것

아이템 간의 유사성을 기준으로 함 → 계산 少 정확도 低

협업 필터링

Utility Matrix

Item-based Approach

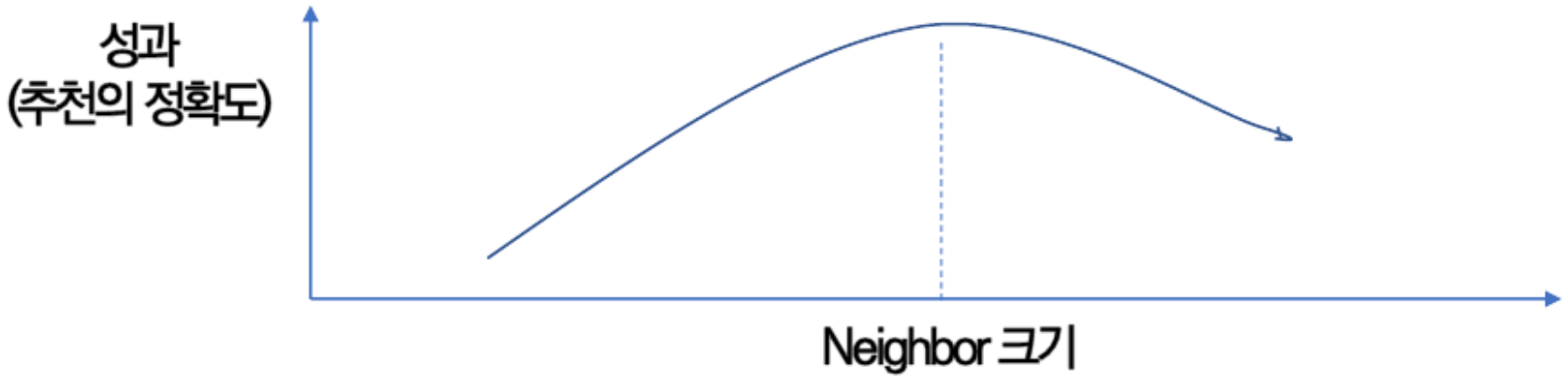
User-based Approach

\ Movie User \	M1	M2	M3	M4	M5	Correlation with U1
U1	2	5	3			-
U2	4	4	3	5	1	0.19
U3	1	5	4		5	0.89
U4	3	5	3	2	5	0.94
U5	4	5	3	4		0.65
U3 & U4 평균	2	5	3.5	2	5	

협업 필터링

알고리즘 구현 시, 고려 사항들

1. Neighbor를 몇 명으로 할 것인가? – KNN에서와 동일한 problem



2. 사용자 취향의 유사성(Similarity)를 어떤 기준으로 계산할 것인가?

- 상관 계수, **코사인 유사도, 유클리디언 거리**, 타니모토 계수 등...

Growth Hackers

협업 필터링

알고리즘 구현 시, 고려 사항들

3. Neighbor에 속한 사용자들의 평점의 단순 평균?

이에 수반하는 두 가지 문제점

1. 유사도가 더 높은 이웃과 낮은 이웃이 동등하게 평가되는 점
2. 평가 성향의 차이가 미반영

(ex. 평가 성향이 후한 사람의 이웃들이 우연히도 평가가 박한 사람들만 있다면, 그 사람들의 평가를 평균 시 사용자의 성향에 비해 낮은 예상 점수가 도출될 수 있음.)

협업 필터링

Sol to 1

그래서 가중 평균한 것을 추천 대상 사용자의 예상 평점으로 사용할 수 있다.

$$p_{ai} = \frac{\sum_{u=1}^{n} w_{au} \times r_{ui}}{\sum_{u=1}^{n} w_{au}}$$

a : 사용자 u : 이웃 사용자 n : 이웃 사용자 수

p_{ai} : 아이템 i에 대한 사용자 a 의 예상 평점

w_{au} : 사용자 a와 u의 유사도

r_{ui} : 아이템 i에 대한 사용자 u의 평점

협업 필터링

Sol to 1 & 2

이제 이 값을 가지고 높은 순서 대로 대상 사용자에게 추천을 해주면 된다

$$p_{ai} = \overline{r_a} + \frac{\sum_{u=1}^{n} w_{au} \times (r_{ui} - \overline{r_U})}{\sum_{u=1}^{n} w_{au}}$$

a : 사용자 u : 이웃 사용자 n : 이웃 사용자 수

p_{ai} : 아이템 i에 대한 사용자 a 의 예상 평점

w_{au} : 사용자 a와 u의 유사도

r_{ui} : 아이템 i에 대한 사용자 u의 평점

$\overline{r_a}$: 사용자 a의 전체 평점 평균

$\overline{r_U}$: 사용자 u의 전체 평점 평균

하지만 하드코딩에서는 이렇게 하지는 않을게요...

차원 축소

UBCF, IBCF의 큰 문제점: 유저와 아이템 수의 증가에 따라 복잡도가 커짐.

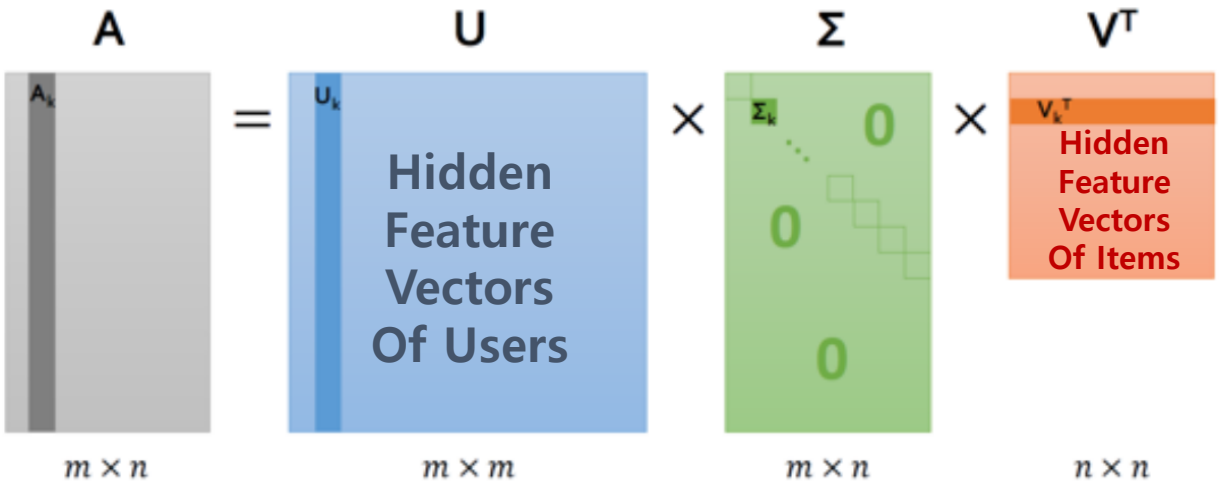
→ Latent Factor Model의 등장

사용자 OR 상품의 특성 벡터를,
몇 개의 잠재적 요인(Latent Factor)으로 이루어진 벡터로 간략화/추상화할 수 있다는 아이디어

차원 축소

SVD (Singular Value Decomposition)

$$A = U\Sigma V^T$$



임의의 m x n 행렬은
(m x m 행렬) x (m x n 행렬) x (n x n 행렬)로 분해될 수 있다.

소인수 분해처럼..
Ex. $x^2 + 3x + 2 = (x+1)(x+2)$

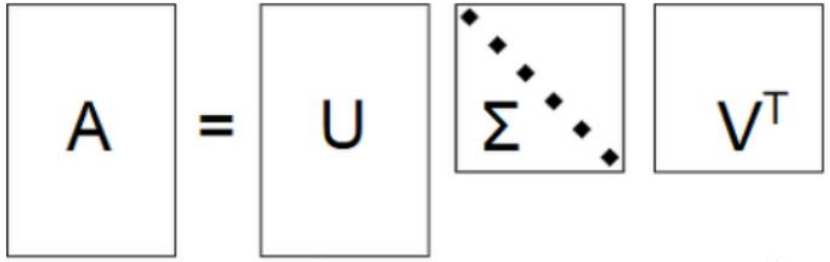
$$U = [\vec{u_1} \quad \vec{u_2} \quad \dots \quad \vec{u_m}]$$
$$V = [\vec{v_1} \quad \vec{v_2} \quad \dots \quad \vec{v_n}]$$
$$\vec{u_k} = \begin{bmatrix} u_{k1} \\ u_{k2} \\ \dots \\ u_{km} \end{bmatrix} \quad \vec{v_k} = \begin{bmatrix} v_{k1} \\ v_{k2} \\ \dots \\ v_{kn} \end{bmatrix}$$
$$\vec{u_k}^T \vec{u_k} = 1, \quad U^T U = I$$
$$\vec{v_k}^T \vec{v_k} = 1, \quad V^T V = I$$

SVD 분해의 존재성 증명: https://en.wikipedia.org/wiki/Singular-value_decomposition

Growth Hackers

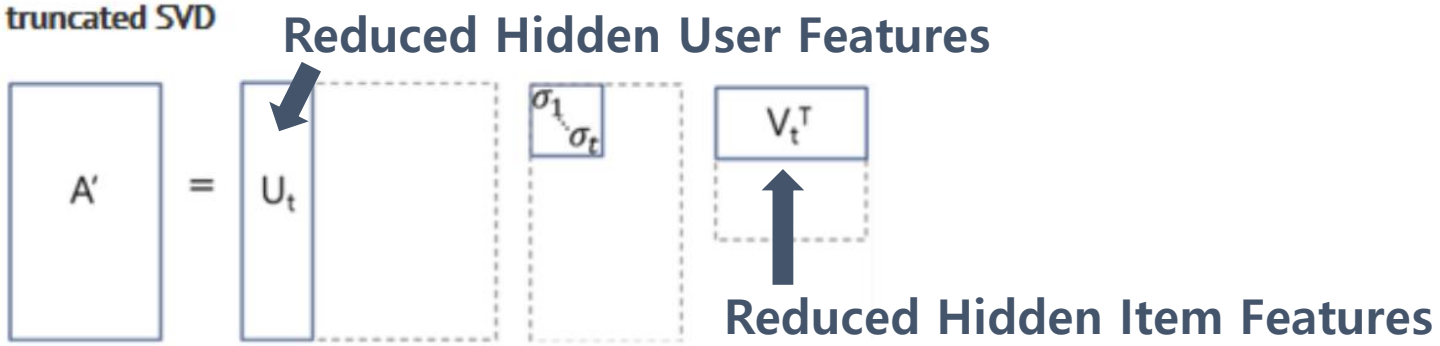
차원 축소

SVD (Singular Value Decomposition)



$$\Sigma_k = \sqrt{\lambda_k} \quad > 0$$

특이값(Σk)과 그에 대응되는 벡터들을 내림차순으로 sorting한 뒤에...



상위 t개만 남기고 모두 truncate

truncated SVD는 Σ 행렬의 대각원소(특이값) 가운데 상위 t개만 골라낸 형태입니다. 이렇게 하면 행렬 A를 원복할 수 없게 되지만, 데이터 정보를 상당히 압축했음에도 행렬 A를 근사할 수 있게 됩니다.

Growth Hackers

차원 축소

SVD (Singular Value Decomposition)

$$R = U\Sigma V^T$$

이 식에서

- U 는 $m \times m$ 크기의 행렬로 역행렬이 대칭 행렬
- Σ 는 $m \times n$ 크기의 행렬로 비대각 성분이 0
- V 는 $n \times n$ 크기의 행렬로 역행렬이 대칭 행렬

Σ의 대각 성분은 특이치라고 하며 전체 특이치 중에서 가장 값이 큰 k개의 특이치만을 사용하여 (Truncated SVD), 다음과 같은 행렬을 만들수 있다.

- $\hat{U}$ 는 U에서 가장 값이 큰 k개의 특이치에 대응하는 k개의 성분만을 남긴 $m \times k$ 크기의 행렬
- $\hat{\Sigma}$ 는 가장 값이 큰 k개의 특이치에 대응하는 k개의 성분만을 남긴 $k \times k$ 크기의 대각 행렬
- $\hat{V}$ 는 V에서 가장 값이 큰 k개의 특이치에 대응하는 k개의 성분만을 남긴 $k \times n$ 크기의 행렬

이 행렬을 다시 조합하면 원래의 행렬과 같은 크기를 가지고 유사한 원소를 가지는 행렬을 만들 수 있다.

$$\hat{U}\hat{\Sigma}\hat{V}^T = \hat{R} \approx R$$

$$R = \sum_{i=1}^{m} \lambda_i u_i v_i'$$

$$\wr \quad \wr$$

$$\hat{R} = \sum_{i=1}^{k} \lambda_i u_i v_i'$$

큰 특이값에 대응되는 것들 위주로 더했으므로 유사할 수밖에 없음.

Growth Hackers

차원 축소

SVD 예시

$$A = U\Sigma V^T$$

$$\begin{bmatrix} 2 & 3 \\ 1 & 4 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.82 & -0.58 & 0 & 0 \\ 0.58 & 0.82 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5.47 & 0 & 0 & 0 \\ 0 & 0.37 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0.40 & 0.91 \\ -0.91 & 0.40 \end{bmatrix}$$

truncated SVD 예시

$$A^{'} = U_1\Sigma_1 V_1^T$$

$$\begin{bmatrix} 1.79 & 4.08 \\ 1.27 & 2.89 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.82 \\ 0.58 \\ 0 \\ 0 \end{bmatrix} [5.47] [0.40 \quad 0.91]$$

Growth Hackers

차원 축소

그러나, 앞의 factorization은 utility matrix가 well-balanced라고 가정된 것
→ 예측을 할 필요가 없음

예측이 요구되는 실제의 데이터는 매우 sparse함

movieId	1	2	3	4	5	6	7	8	9	10	...	6031	6032	6033	6034	6035	6036	6037	6038	6039	6040
userId																					
1	5.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	5.0	5.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	3.0	0.0	0.0	0.0	0.0

그래서 통계적인 방식으로 factorization을 해야 함

https://surprise.readthedocs.io/en/stable/matrix_factorization.html?highlight=svd

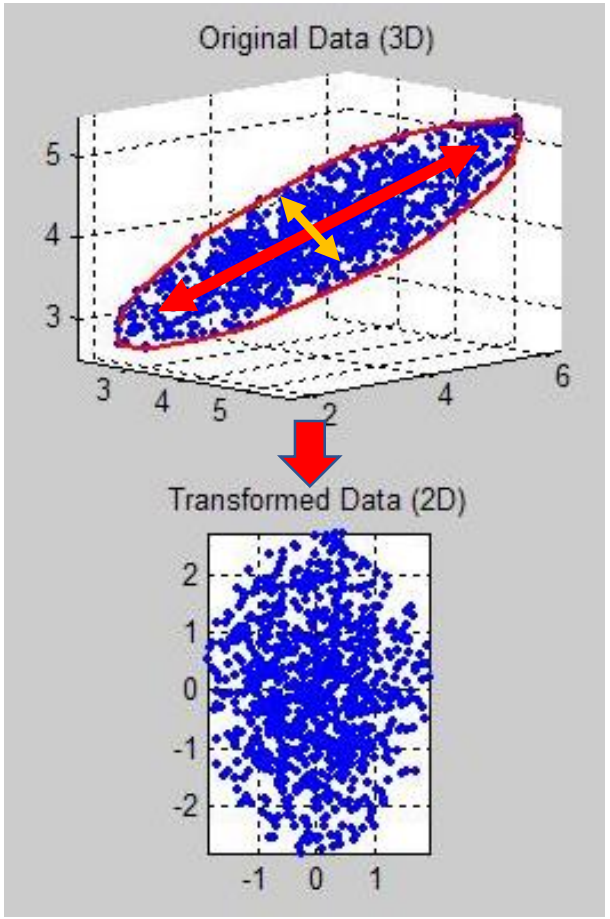
차원 축소

유저의 특성을 요약된 feature들로 나타낼 방법은 없을까?

	M1	M2	M3	M4	M5
U1	2	5	3		
U2	4	4	3	5	1
U3	1	5	4		5
U4	3	5	3	2	5
U5	4	5	3	4	

	코믹	멜로	액션
U1	5	3	1
U2	3	5	2
U3	4	2	1
U4	1	2	1
U5	5	5	2

차원 축소



PCA의 기본 아이디어

← 여기서의 점들을 표현하기 위해서는 (x, y, z) – 즉 3차원 정보가 필요!

← 근데 잘만 옮기면,
(x, y) 2차원으로도 이 정보들을 잘 표현할 수 있을 것 같은데...?!

Growth Hackers

차원 축소

Say, ***X* : *k x n* matrix** – *k*: # of features, *n*: # of observations

Want to convert (n x k) -> ()

$$\begin{pmatrix} - x1' - \\ - x2' - \\ .. \\ - xk' - \end{pmatrix}$$ <- make each row has zero mean!

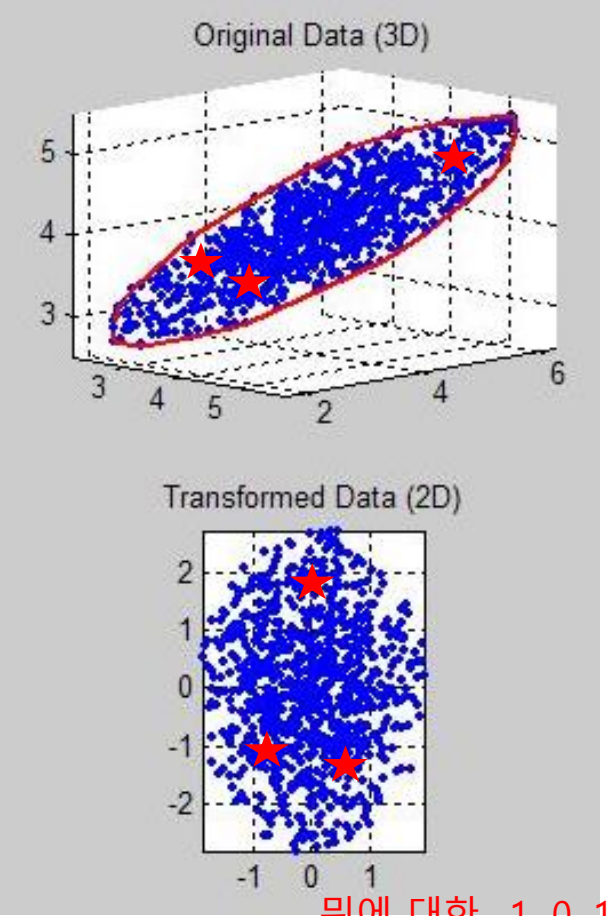
Want to find a linear map P s.t. each row of $Y=PX$ has max. variance & zero cov. with other rows P: k x k

$$\Sigma = YY^T = (PX)(PX)^T = P(XX^T)P^T$$ It's like diagonalization!

Sort eigenvalues & eigenvectors in descending order, which makes important vec. placed on the upper side
In this way, p_is which make high variance can remain…

$$P = \begin{pmatrix} - p1' - \\ - p2' - \\ .. \\ - pk' - \end{pmatrix} \Rightarrow PX = \begin{pmatrix} p1'x.1 & p1'x.2 & \dots & p1'xn \\ p2'x.1 & p2'x.2 & \dots & p2'xn \\ & & \dots & \\ pk'x.1 & pk'x.2 & \dots & pk'xn \end{pmatrix}$$ Use this as new X

차원 축소



뭐에 대한 -1, 0, 1?

중요한 건,
새로 mapping된 값들이 가지는 직관적 의미가 사라진다는 점!

Element 간의 관계에 주목할 때에만 사용!

친절한 설명: http://t-robotics.blogspot.com/2014/10/pca-principal-component-analysis.html#.W6y-fmgzZPa

조금 더 자세한 설명:
https://ko.wikipedia.org/wiki/%EC%A3%BC%EC%84%B1%EB%B6%84_%EB%B6%84%EC%84%9D

Growth Hackers

교차 타당성 검증

통상적인 정확도 검정 방식

Train set과 Test set을 나누어서, Train set에 기반해 모델을 만들고, Test set에 대한 **예측력 및 정확도**를 도출

이용되는 척도들

- RMSE

$$\mathrm{RMSE} = \sqrt{\frac{1}{|\hat{R}|}\sum_{\hat{r}_{ui}\in\hat{R}}(r_{ui}-\hat{r}_{ui})^2}$$

단순 평균이 좋지 않은 이유
→ (-100 + 100)/2 = 0

- Precision, Recall

Actual \ Predicted	Negative	Positive
Negative	A	B
Positive	C	D

$$\text{Precision} = \frac{a+d}{a+b+c+d} \qquad \text{True positive recommendation} = \frac{b}{b+d}$$

$$\text{Recall} = \frac{d}{c+d} \qquad \text{False positive recommendation} = \frac{d}{b+d}$$

Growth Hackers

교차 타당성 검증

통상적으로 **학습용으로 확보한 데이터를 학습용과 검증용으로 분리하여 학습용 데이터로 학습한 후 검증용 데이터로 검증한다**.

이 때 데이터를 어떻게 분리하느냐에 따라 검증 성능이 조금씩 달라질 수 있으므로 여러가지 방식으로 데이터를 분리하여 검증을 실시하고 **평균 성능(mean performance)과 성능 분산(performance variance)**를 모두 구한다.

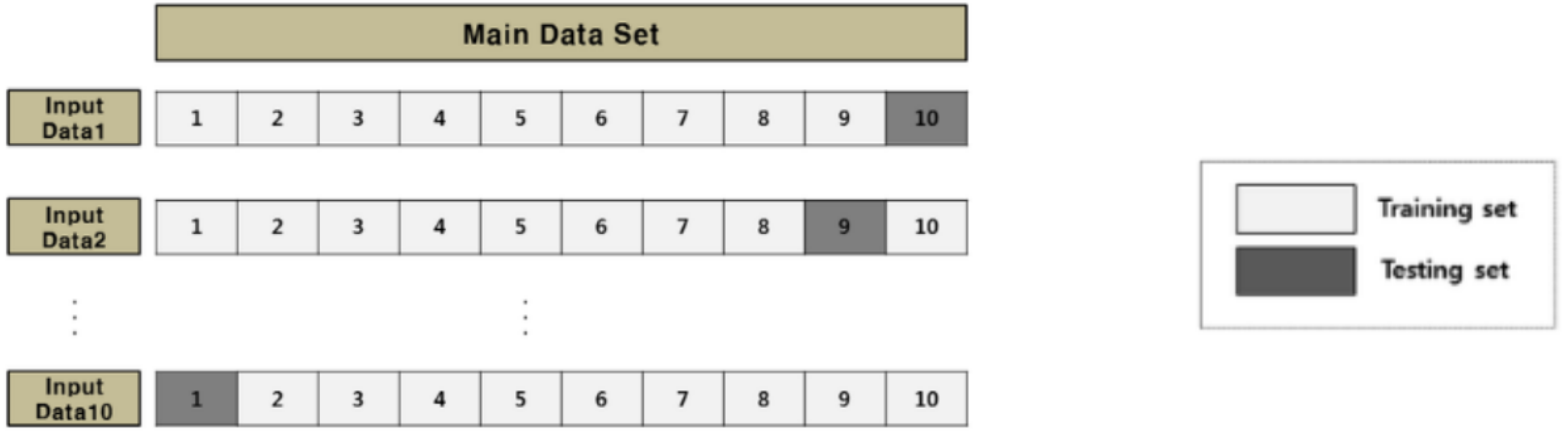
이러한 검증 방법을 **교차 검증(cross validation)**이라고 한다.

$$Var(\frac{X}{n}) = \frac{\sigma^2}{n}$$

교차 타당성 검증

K-Fold Cross Validation

K-fold CV(cross-validation) 방법은 데이터 셋을 K개의 sub-set로 분리하는 방법이다.
분리된 K개의 sub-set 중 하나만 제외한 K-1개의 sub-sets를 training set으로 이용하여 K개의 모형을 추정한다.



Growth Hackers

교차 타당성 검증

K-Fold Cross Validation

→ 사이킷런의 Kfold 모듈을 import하여 k-fold CV를 시행할 수 있다.

```
from sklearn.model_selection import Kfold

cv = KFold(n_splits=3, shuffle=True, random_state=0)

for train_index, test_index in cv.split(X):
        print("test index :", test_index)
        print("train index:", train_index)
```

cv.split(X)는 X를 k-fold로 나눈 뒤,
각 fold에서의 (train_index, test_index)를
반환하는 iterable

Surprise 패키지

Surprise는 Nicolas Hug가 만든 오픈 소스 추천 시스템 패키지
~~(여태까지 배운 것들 다 이걸로 할 수 있음.)~~

- https://github.com/NicolasHug/Surpirse
- http://surprise.readthedocs.io/en/latest/index.html

Surprise 패키지는 콘솔에서 pip 명령으로 설치하거나 github에서 repository를 clone하여 직접 설치

With pip (you'll need numpy, and a C compiler. Windows users might prefer using conda):

```
$ pip install numpy
$ pip install scikit-surprise
```

With conda:

```
$ conda install -c conda-forge scikit-surprise
```

Surprise 패키지

SVD 모형은 최신 버전의 master branch에만 있으므로 git clone해야 사용할 수 있다.

or the latest version, you can also clone the repo and build the source (you'll first need Cython and numpy):

```
$ pip install numpy cython
$ git clone https://github.com/NicolasHug/surprise.git
$ cd surprise
$ python setup.py install
```

이제 Jupyter Notebook으로 갑시다 ᕙ(˙ᗜ˙)ᕗ

참고 자료

- 서승현(GH 2기), 추천 시스템 세션 자료
- http://blog.kthdaisy.com:8080/recommendation_system_kthdaisy/
- https://en.wikipedia.org/wiki/Singular-value_decomposition
- https://stats.stackexchange.com/questions/31096/how-do-i-use-the-svd-in-collaborative-filtering
- http://t-robotics.blogspot.com/2014/10/pca-principal-component-analysis.html#.W6y-fmgzZPa
- https://ko.wikipedia.org/wiki/%EC%A3%BC%EC%84%B1%EB%B6%84_%EB%B6%84%EC%84%9D
- https://surprise.readthedocs.io/en/v1.0.0/_modules/surprise/prediction_algorithms/knns.html
- https://surprise.readthedocs.io/en/stable/matrix_factorization.html?highlight=svd
- https://datascienceschool.net/view-notebook/fcd3550f11ac4537acec8d18136f2066/
- https://datascienceschool.net/view-notebook/266d699d748847b3a3aa7b9805b846ae/

QUEST

Surprise 내장 영화 데이터를 로드하여 "UBCF hardcoding.ipynb" 내의 코드를 IBCF 코드로 바꾸고,
코사인 유사도(cos_sim 함수 이용)를 기준으로 KNN 알고리즘을 작성해주세요.
단, 5-fold Cross Validation을 시행하여, 정확도의 평균을 최종적으로 출력해주세요.