행렬 완성 개론

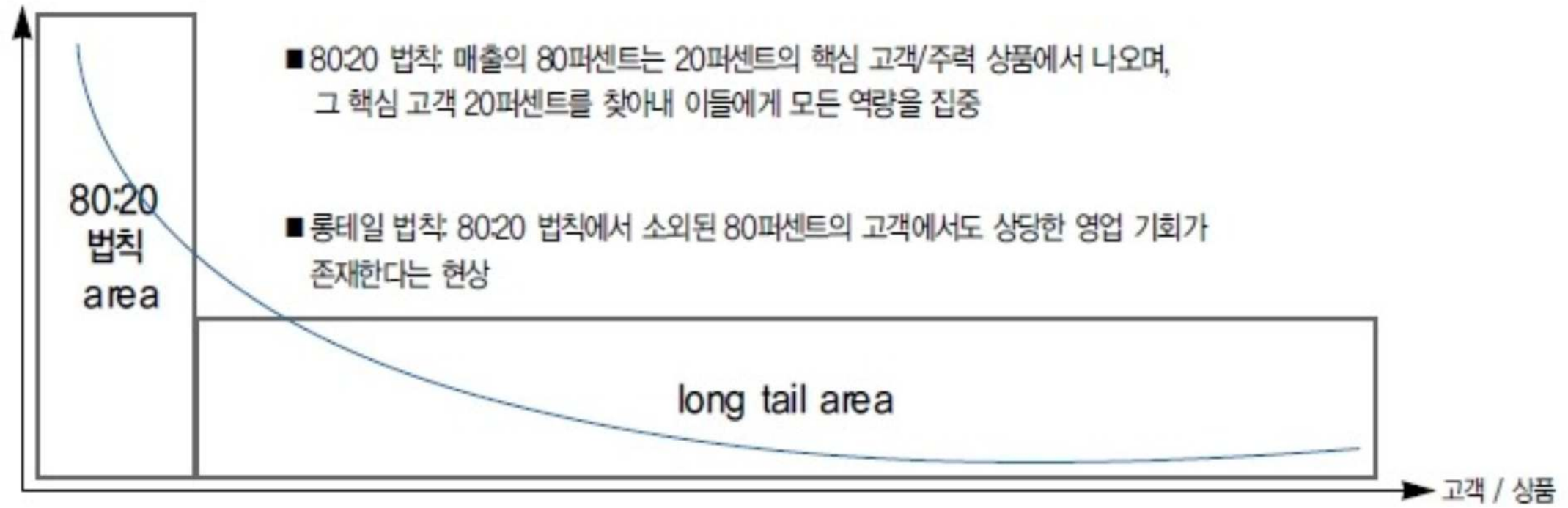
Introduction to Matrix Completion

KAIST 전산학부 기계학습 및 지능연구실(MLILAB)
석사과정 이준영
2018-07-18

“설명하기에 앞서, 이 슬라이드는 간결한 설명을 위해
엄밀하지 않게 제작되었음을 알려드립니다.”

추천의 중요성

파레토 법칙과 롱테일 법칙



출처: http://mbanote2.tistory.com/197

- IT 서비스는 오프라인 점포와 달리 롱테일에 있는 상품들을 진열 하는 데 드는 비용이 매우 적음.
- **심지어 매입 가격도 훨씬 저렴함.**
  (음악 스트리밍 서비스의 경우, 신곡보다 오래된 곡이 비용이 더 저렴함)
- **고객들에게 고정된 금액을 받는데, 고객 만족도를 유지하면서 저렴한 제품을 공급할 수 있다면?**

추천의 중요성

롱테일 법칙 사례

amazon

추천을 통한 판매가 전체 판매의 **35%**

https://m.blog.naver.com/PostView.nhn?blogId=sanny0314&logNo=220630765408&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F

NETFLIX

추천을 통한 재생이 전체 재생의 **67%**

http://www.hani.co.kr/arti/PRINT/694128.html

Spotify®

Discover Weekly 추천을 통한 재생이 전체 재생의 **4%**

https://jacesky1.wordpress.com/2016/10/09/%EB%8D%B0%EC%9D%B4%ED%84%B0%EB%A1%9C-%EC%9D%8C%EC%95%85%EC%9D%84-%EB%8D%94%EC%9A%B1-%EB%B9%9B%EB%82%98%EA%B2%8C-spotify/ https://brunch.co.kr/@hmin0606/7

kakao

간단한 알고리즘의 도입만으로 클릭률 **150%** 달성!

KAIST 리크루팅에서 이야기 했음.

추천의 중요성

롱테일 법칙 사례

amazon

추천을 통한 판매가 전체 판매의 35%

https://m.blog.naver.com/PostView.nhn?blogId=sanny0314&logNo=220630765408&proxyReferer=https%3A%2F%2Fwww.google.co.kr%2F

NETFLIX

추천을 통한 재생이 전체 재생의 **67%**

http://www.hani.co.kr/arti/PRINT/694128.html

Spotify®

Discover Weekly 추천을 통한 재생이 전체 재생의 **4%**

kakao

간단한 알고리즘의 도입만으로 클릭률 **150%** 달성!

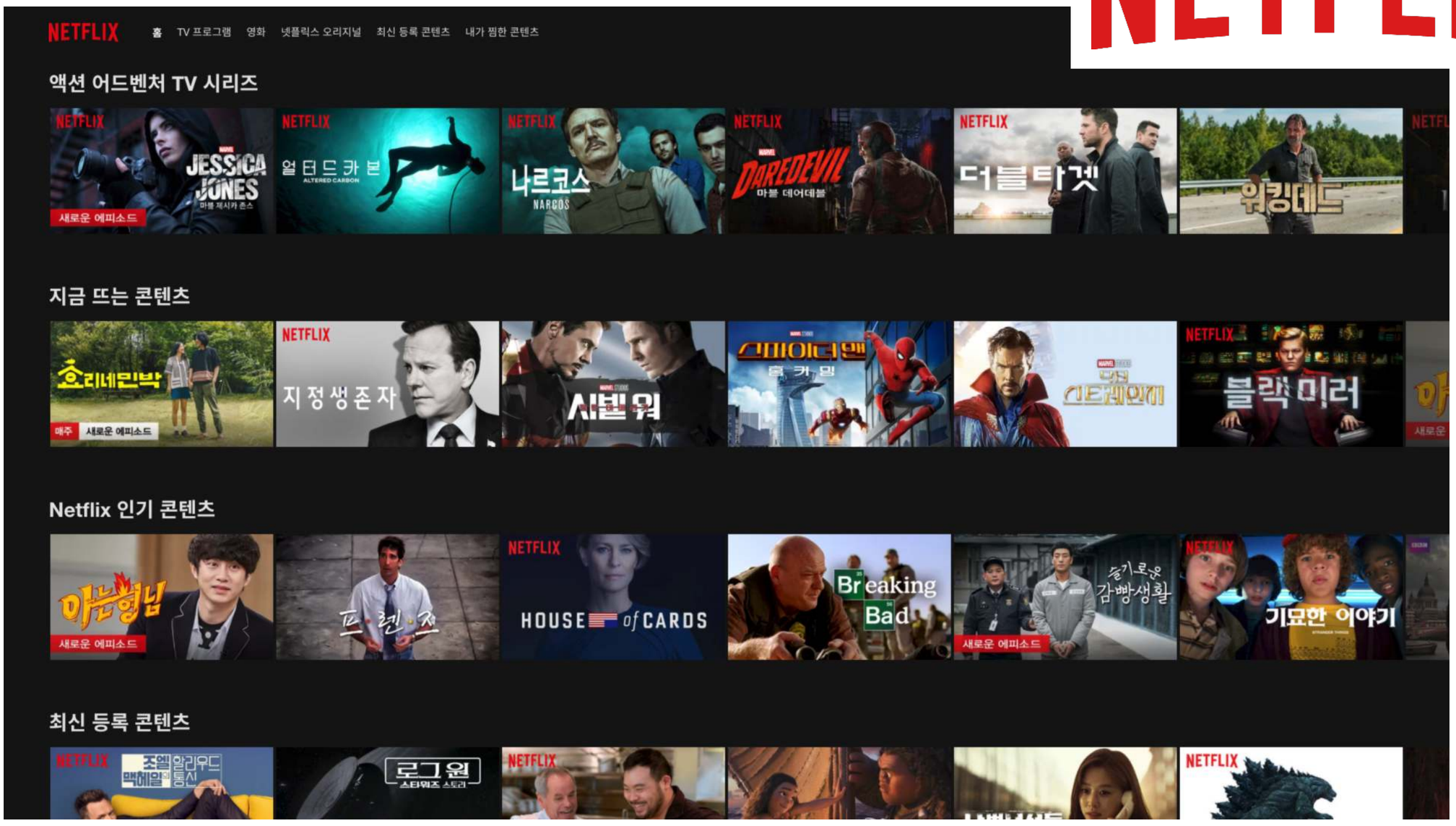
KAIST 리크루팅에서 이야기 했음.

오늘 세미나에서는
NETFLIX의 추천에 대해서
집중적으로 다뤄보고자 합니다!

Netflix Prize

넷플릭스 : 세계 최대 VOD 스트리밍 사이트

NETFLIX



WATCHA PLAY 는 한국형 넷플릭스!

Netflix Prize

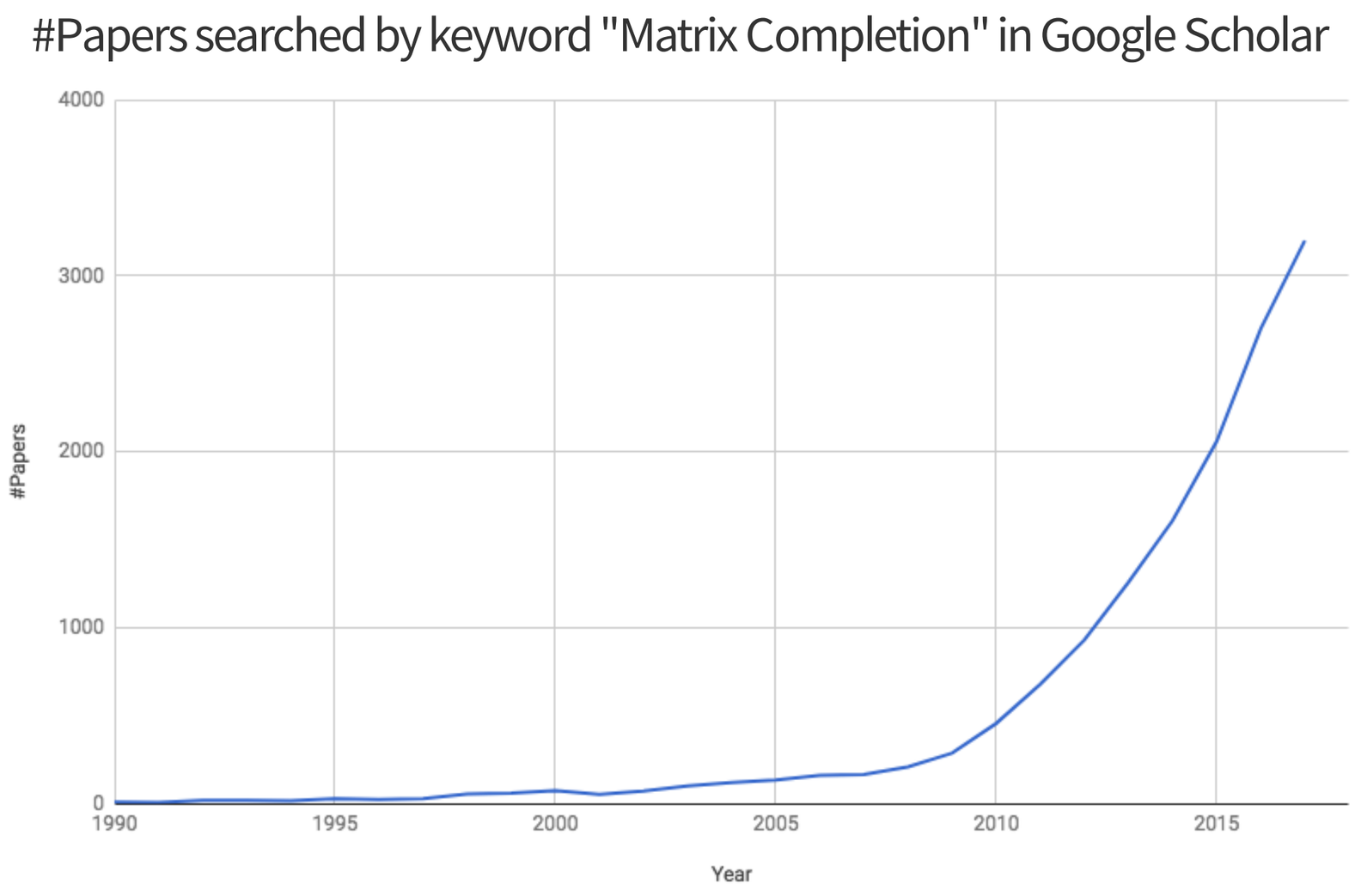
2006년, 넷플릭스는 Netflix Prize를 개최

백만달러(약 12억원)의 상금이 걸린 대회를 열었음!

현재 넷플릭스 추천 엔진 성능의 10%를 개선한 추천 엔진을 만들면 백만달러를 주는 대회.

Netflix Prize

2006년 이후, Netflix Prize 관련 논문 수 폭발적 증가



Netflix Prize를 기점으로
많은 연구자들이 Netflix Problem을 풀기 위해 노력하기 시작!

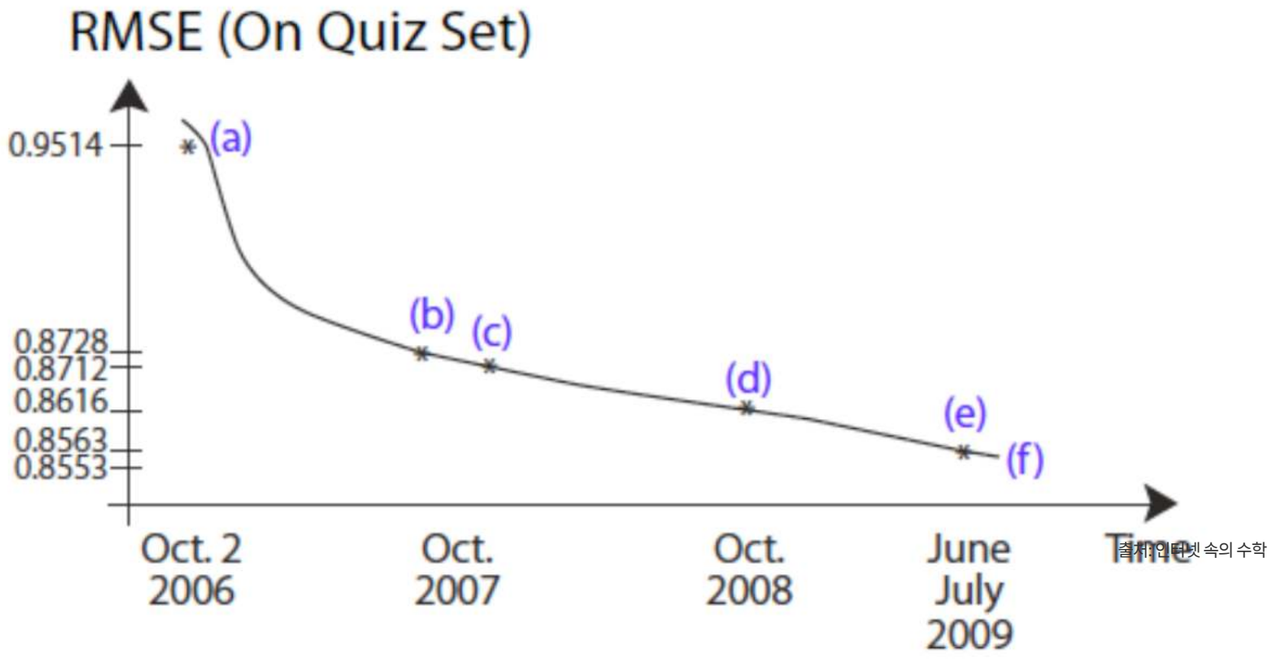
Netflix Prize

당시 넷플릭스가 보유하고 있던 추천 엔진

- 지금은 Netflix에 별점을 주는 기능이 사라졌지만, 이 때만해도 별점을 주는 기능이 있었음.
- 그리고, 그 별점 이력을 바탕으로 추천을 수행했음.
- 기존 넷플릭스의 추천엔진: Cinematch
  - "straightforward statistical linear models with a lot of data conditioning”

출처: https://web.archive.org/web/20070821194257/http://www.netflixprize.com/faq

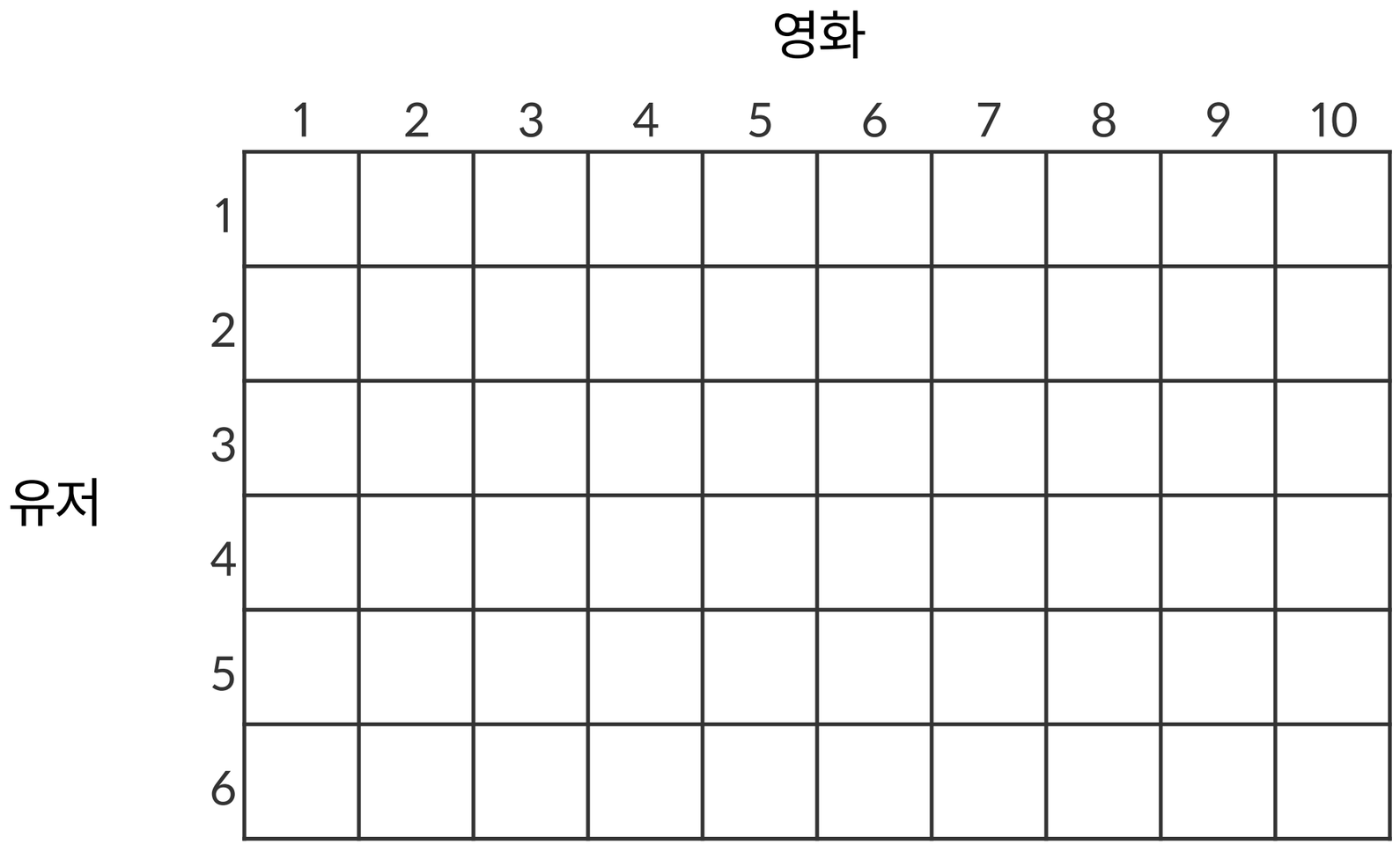
- Cinematch를 10% 개선하는 것이 얼마나 어려울까?
  - 10% 개선에 3년이 걸림!



Netflix Prize

The Problem of Netflix Prize

다음과 같은 Rating 행렬이 주어진다고 가정(열이 유저, 행이 영화).



Netflix Prize

The Problem of Netflix Prize

Rating 행렬에서 일부 Rating 에 대한 정보를 알고 있을 때,

영화

유저	1	2	3	4	5	6	7	8	9	10
1		4		2	4					
2	3		3	1			3		3	
3		3	2	3	1				4	
4			2			4		1	2	5
5	3			3			1			
6			2					3	2	

Netflix Prize

The Problem of Netflix Prize

모르는 행렬 entry에 대한 rating을 맞추는 문제!

행렬을 완성한다고 해서 **행렬완성(Matrix Completion)** 문제라고 합니다.

* 실제 Netflix Problem에서는 rating이 된 timestamp 정보까지 주어졌긴 했습니다만, 간결함을 위해 이 슬라이드에서는 제외하였습니다.

Netflix Prize

Netflix Prize Winner

In 2009, the BellKor's Programatic Chaos 팀이 Netflix Prize에서 우승!

어떻게 기존 알고리즘을 10% 개선했을까?

Netflix Prize

Winner’s Main Idea

10% 향상 중에 8% 향상에 기여한 아이디어에 대해서만 소개! (나머지 향상은 대부분 앙상블!)

영화

유저	1	2	3	4	5	6	7	8	9	10
1		4		2	4					
2	3		3	1			3		3	
3		3	2	3					4	
4			2			4		1	2	5
5	3			3			1			
6			2					3	2	

아무런 가정이 없다면, 행렬을 완성할 수 없음!
(가정이 없으면 Gain도 없다)
-> Low-rank 가정을 추가해 행렬을 완성!

Low-rank Matrix Completion(LRMC)

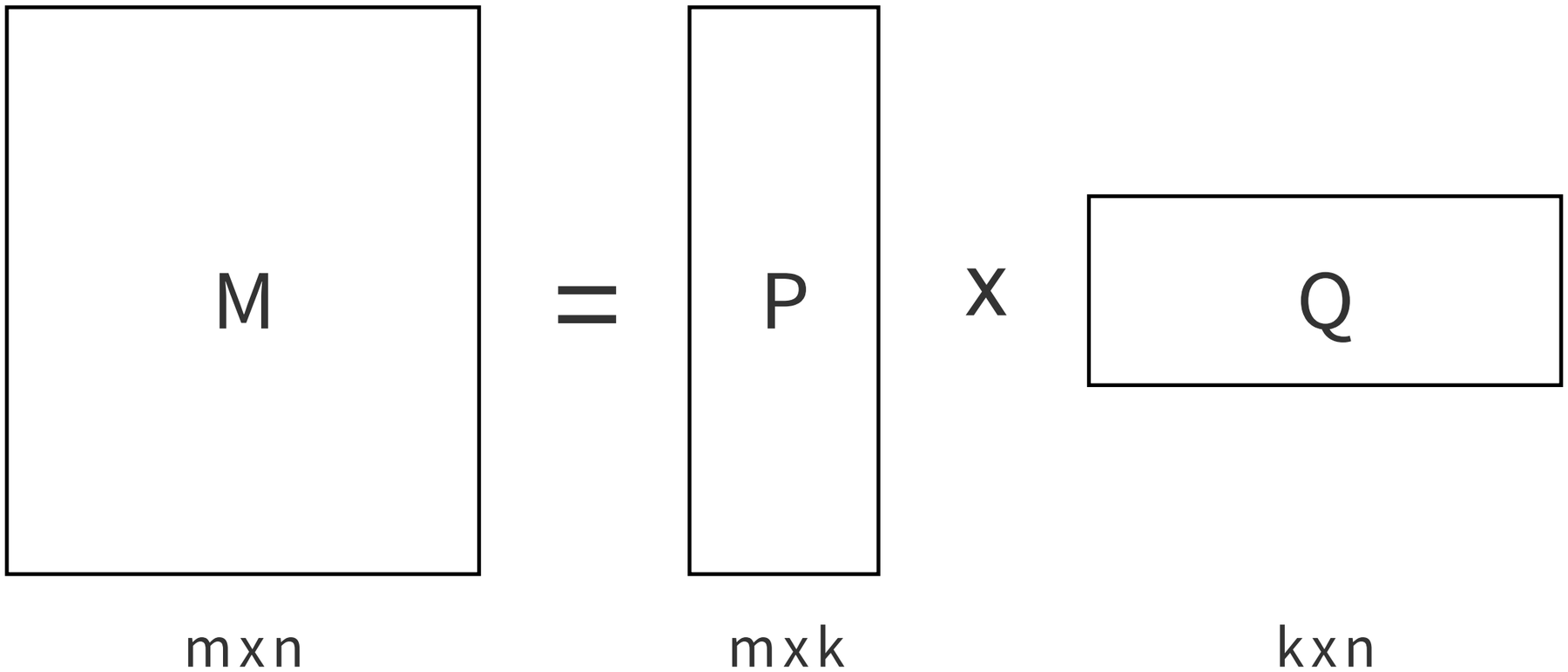
Recall: 행렬의 rank란 무엇인가?

- 정의: 행렬 A의 row space의 dimension과 column space의 dimension 중 작은 값.

Low-rank Matrix Completion(LRMC)

Recall: 행렬의 rank란 무엇인가?

- 정의: 행렬 A의 row space의 dimension과 column space의 dimension 중 작은 값.



어떤 행렬 M의 rank가 k이면,
위 등식을 만족하는 행렬 P, Q 가 존재합니다.*

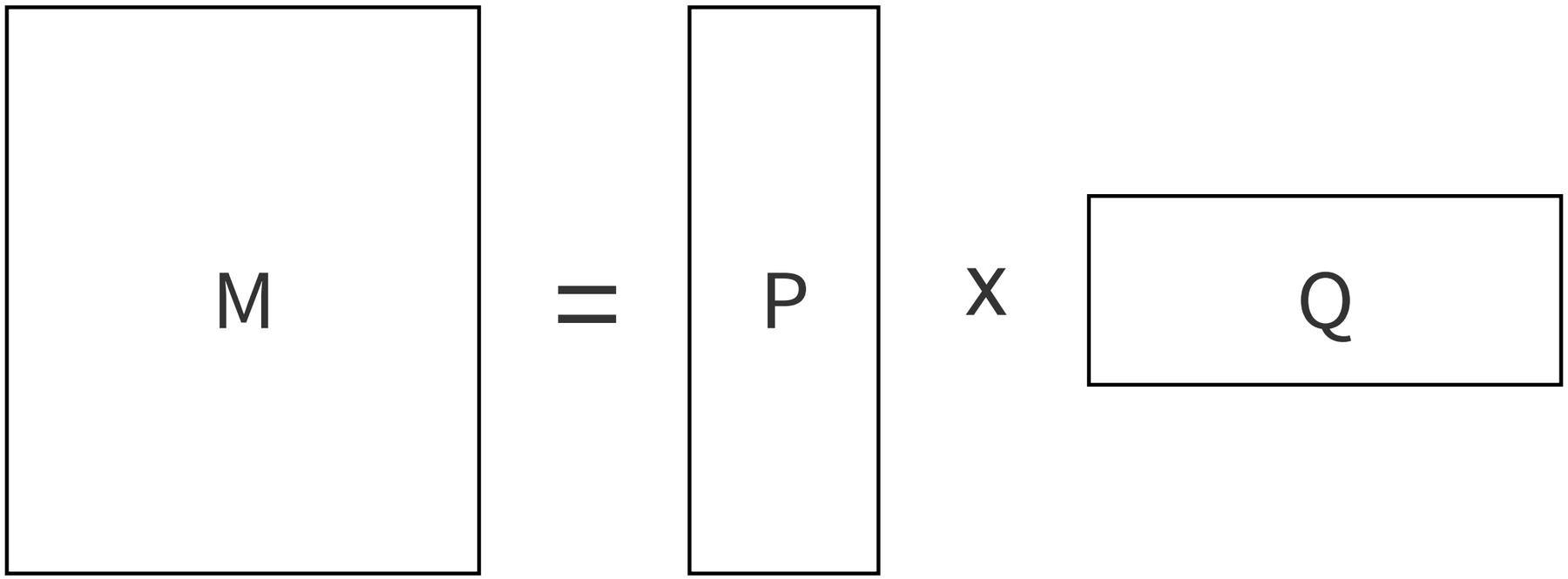
* Howard Anton, Robert C. Busby. Theorem 8.6.1. Contemporary Linear Algebra. ISBN 978-0-471-16362-6 Wiley.

Low-rank Matrix Completion(LRMC)

low-rank란 무엇인가?

- rank가 행렬의 크기에 비해 충분히 작다는 것!

$$k \ll \min(n, m)$$

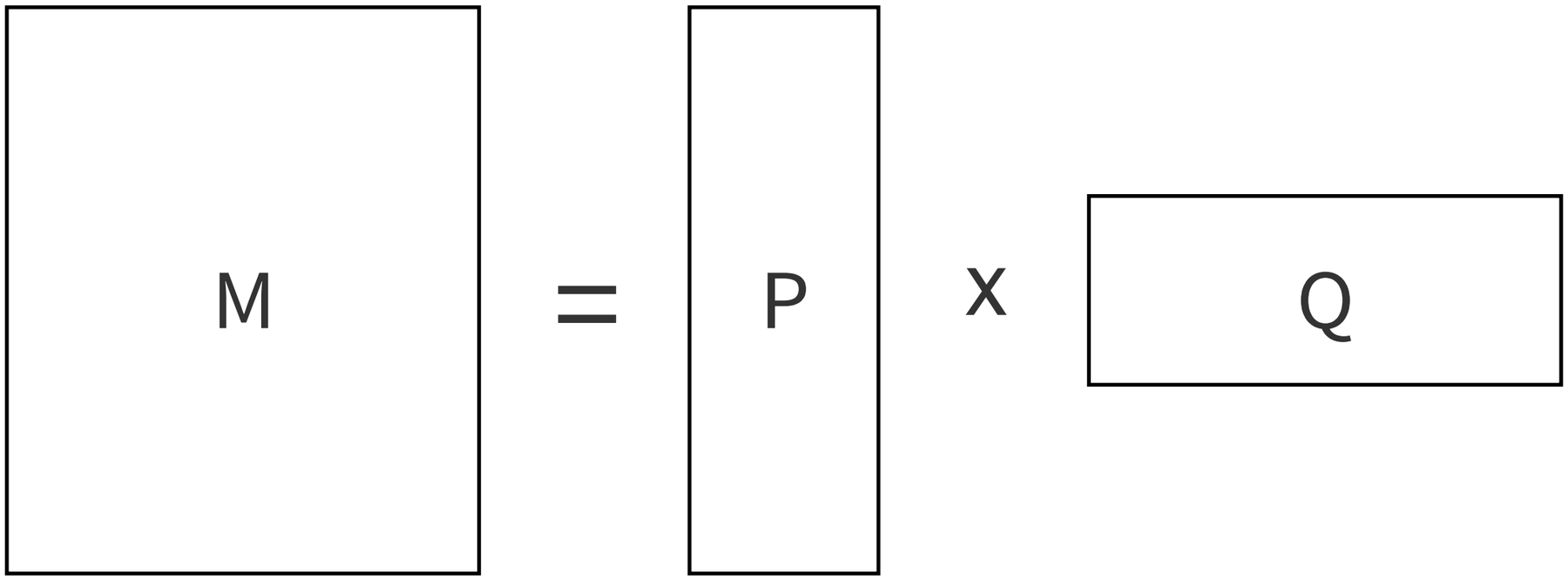


Low-rank Matrix Completion(LRMC)

low-rank란 무엇인가?

- rank가 행렬의 크기에 비해 충분히 작다는 것!

$$k \ll \min(n, m)$$

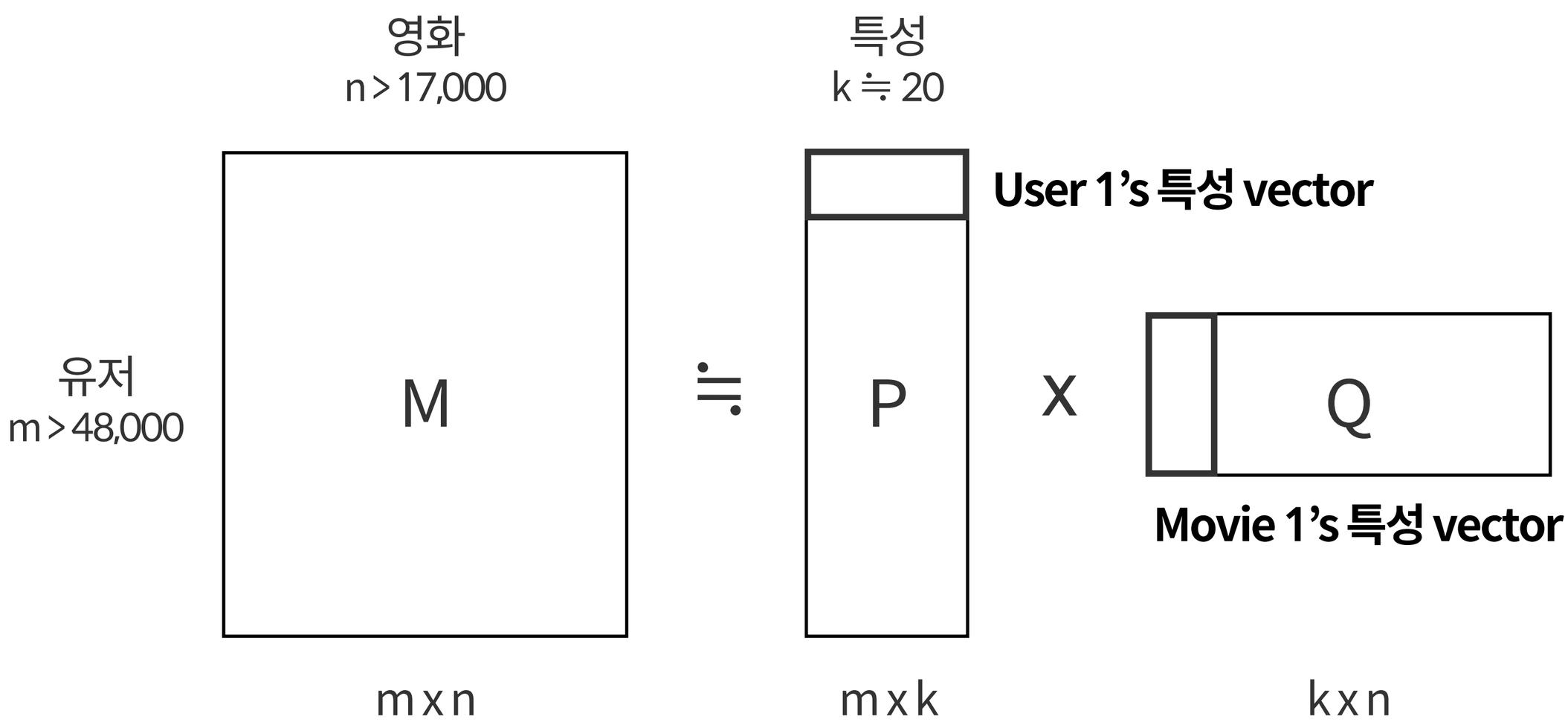


넷플릭스 Prize 우승팀의 경우에는 k를 약 20정도로 설정함!

$m > 48000, n > 17000$

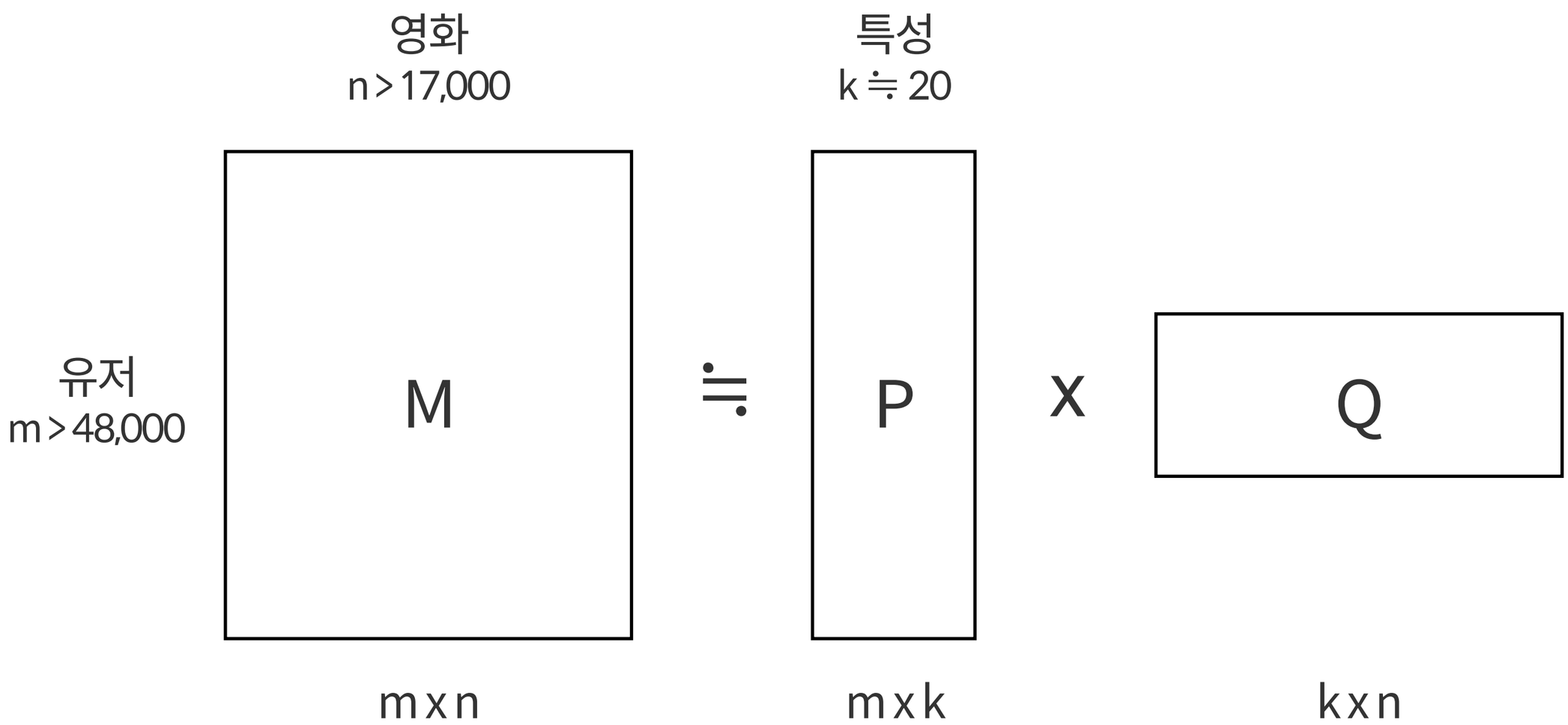
Low-rank Matrix Completion(LRMC)

Low-rank Matrix Completion for Netflix Prize



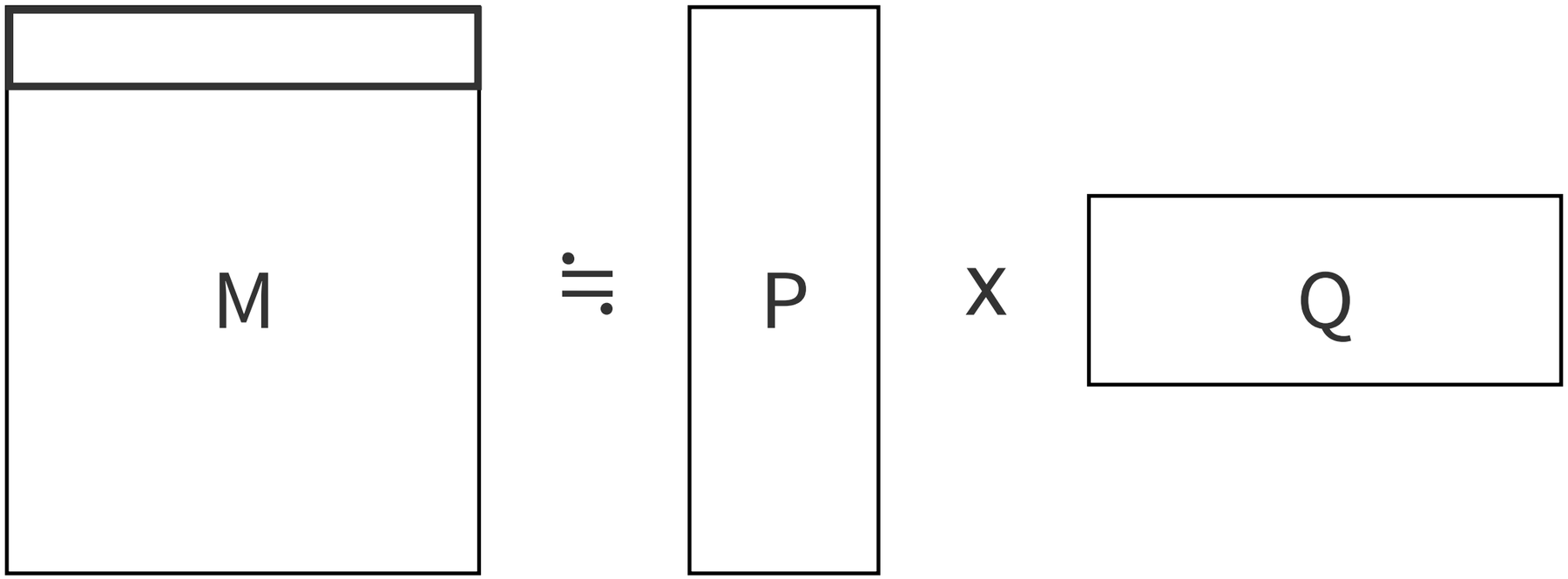
Low-rank Matrix Completion(LRMC)

Low-rank Matrix Completion for Netflix Prize



Low-rank Matrix Completion(LRMC)

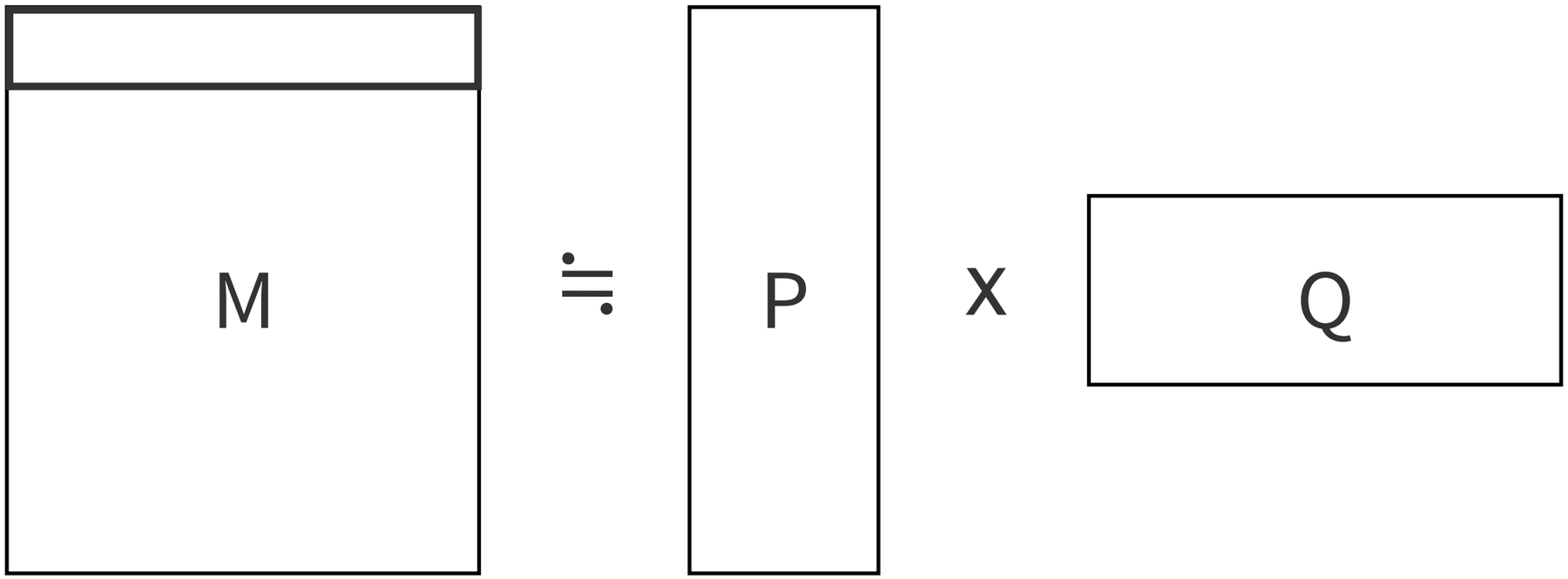
왜 Low-rank 가정이 합리적인가?(1/2)



행렬 M의 rank가 k 라는 말은, 행렬 M의 각 row vector들이
행렬 M의 k개의 row vector들의 linear combination으로 표현 가능하다는 말!

Low-rank Matrix Completion(LRMC)

왜 Low-rank 가정이 합리적인가?(1/2)

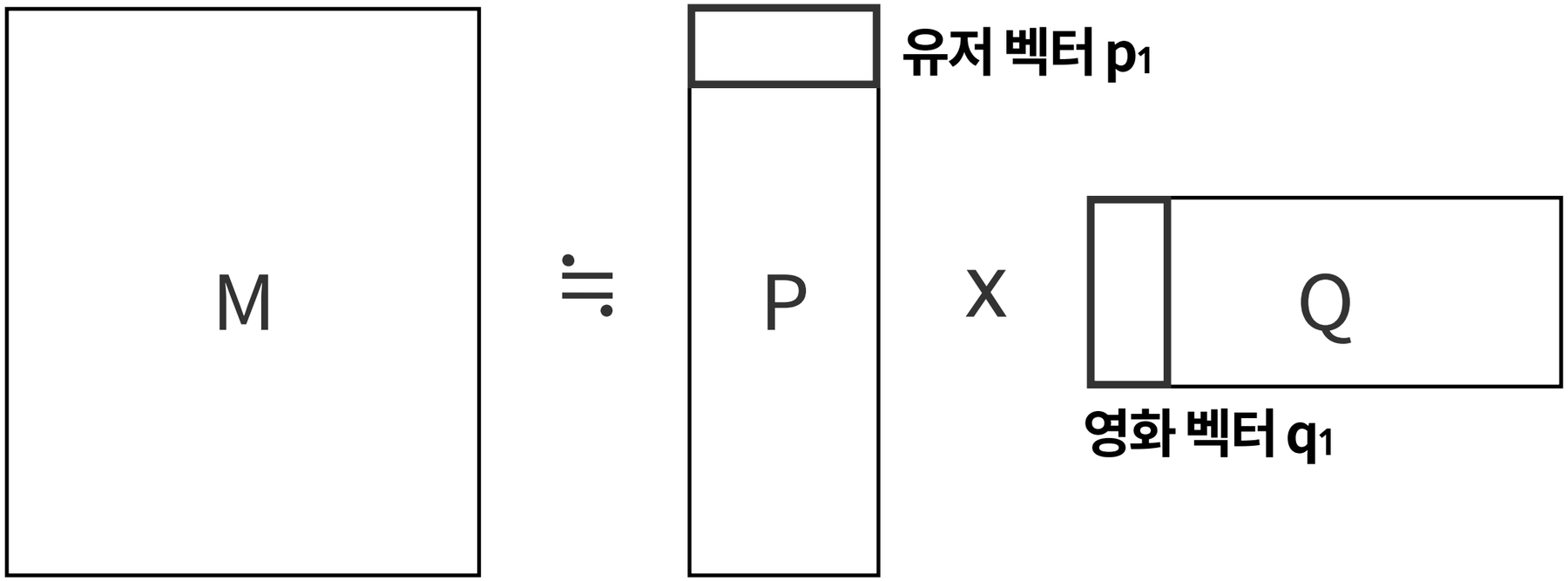


행렬 M의 rank가 k 라는 말은, 행렬 M의 각 row vector들이
행렬 M의 k개의 row vector들의 linear combination으로 표현 가능하다는 말!

다시말해서, 어떤 유저의 취향은
소수(k 명의) 유저들의 취향의 linear combination으로 표현 가능!
즉, 서로 서로 취향이 비슷하다!

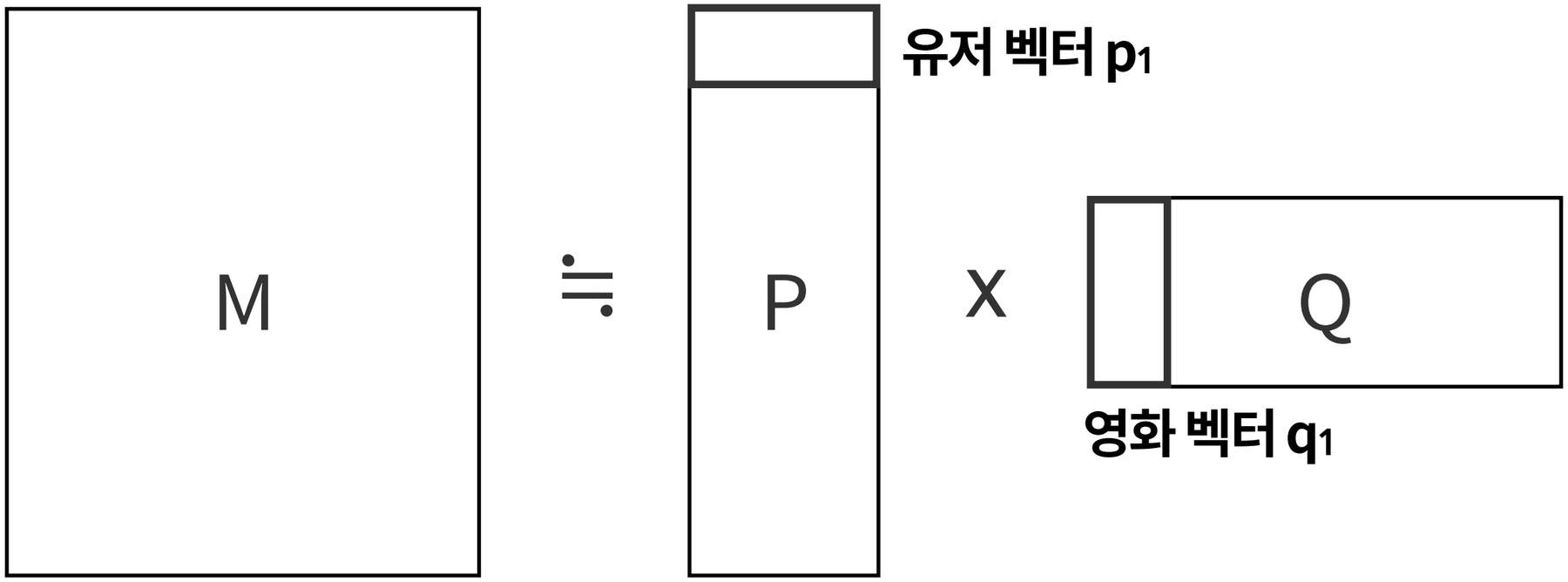
Low-rank Matrix Completion(LRMC)

왜 Low-rank 가정이 합리적인가?(2/2)



Low-rank Matrix Completion(LRMC)

왜 Low-rank 가정이 합리적인가?(2/2)



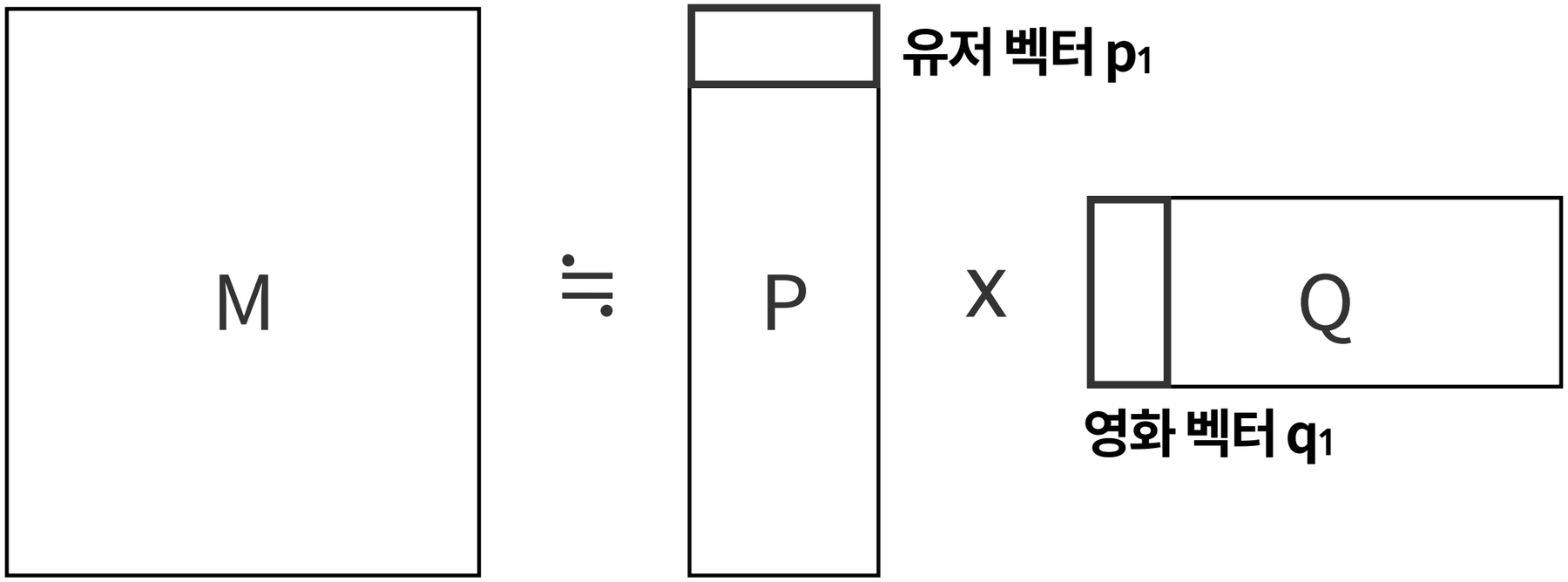
- 고전 추천 시스템에서는 데이터 사이언티스트들이 다음과 같이 추천을 진행!

	액션	스릴러	…	드라마
유저 벡터 p_1	3	4		5
영화 벡터 q_1	0.1	0.2		0.05

데이터 사이언티스트들이 유저 마다, 영화마다 이 값을 정해서 입력해줬어야 했다.

Low-rank Matrix Completion(LRMC)

왜 Low-rank 가정이 합리적인가?(2/2)



- 우리는 각각의 장르를 latent variable로 볼 수 있다.

	장르$_1$	장르$_2$	…	장르$_k$
유저 벡터 p_1	3	4		5
영화 벡터 q_1	0.1	0.2		0.05

Low-rank Matrix Completion(LRMC)

Low-rank Matrix Completion의 수학적 정의

- 먼저 $P_\Omega(S)$를 정의하자.

$$\mathcal{P}_\Omega(S) = \begin{cases} S_{ij} & \text{if } (i,j) \in \Omega, \\ 0 & \text{otherwise.} \end{cases}$$

← Set of known entries

Low-rank Matrix Completion(LRMC)

Low-rank Matrix Completion의 수학적 정의

- 먼저 $P_\Omega(S)$를 정의하자.

$$\mathcal{P}_\Omega(S) = \begin{cases} S_{ij} & \text{if } (i,j) \in \Omega, \\ 0 & \text{otherwise.} \end{cases}$$

← Set of known entries

$$\min_{P,Q} \|\mathcal{P}_\Omega(PQ) - \mathcal{P}_\Omega(M)\|_F^2$$

그러니까 기존 데이터를 가장 잘 설명하는 Low-rank Matrix를 찾는 것!

Low-rank Matrix Completion(LRMC)

Low-rank Matrix Completion의 수학적 정의

- 먼저 $P_\Omega(S)$를 정의하자.

$$\mathcal{P}_\Omega(S) = \begin{cases} S_{ij} & \text{if } (i,j) \in \Omega, \\ 0 & \text{otherwise.} \end{cases}$$

← Set of known entries

$$\min_{P,Q} \|\mathcal{P}_\Omega(PQ) - \mathcal{P}_\Omega(M)\|_F^2$$

그러니까 기존 데이터를 가장 잘 설명하는 Low-rank Matrix를 찾는 것!

아쉽게도 이 문제는 NP-Hard!

Low-rank Matrix Completion(LRMC)

Low-rank Matrix Completion의 수학적 정의

- 먼저 $P_\Omega(S)$를 정의하자.

$$\mathcal{P}_\Omega(S) = \begin{cases} S_{ij} & \text{if } (i,j) \in \Omega, \\ 0 & \text{otherwise.} \end{cases}$$

← Set of known entries

$$\min_{P,Q} \|\mathcal{P}_\Omega(PQ) - \mathcal{P}_\Omega(M)\|_F^2$$

그러니까 기존 데이터를 가장 잘 설명하는 Low-rank Matrix를 찾는 것!

아쉽게도 이 문제는 NP-Hard!

아직은 polynomial time 안에 해를 찾을 수 없다!

Low-rank Matrix Completion(LRMC)

Low-rank Matrix Completion의 수학적 정의

- 먼저 $P_\Omega(S)$를 정의하자.

$$\mathcal{P}_\Omega(S) = \begin{cases} S_{ij} & \text{if } (i,j) \in \Omega, \\ 0 & \text{otherwise.} \end{cases}$$

← Set of known entries

$$\min_{P,Q} \|\mathcal{P}_\Omega(PQ) - \mathcal{P}_\Omega(M)\|_F^2$$

그러니까 기존 데이터를 가장 잘 설명하는 Low-rank Matrix를 찾는 것!

아쉽게도 이 문제는 NP-Hard!

그러면, 어떻게 풀까?

Low-rank Matrix Completion(LRMC)

Low-rank Matrix Completion의 수학적 정의

- 먼저 $P_\Omega(S)$를 정의하자.

$$\mathcal{P}_\Omega(S) = \begin{cases} S_{ij} & \text{if } (i,j) \in \Omega, \\ 0 & \text{otherwise.} \end{cases}$$

← Set of known entries

$$\min_{P,Q} \|\mathcal{P}_\Omega(PQ) - \mathcal{P}_\Omega(M)\|_F^2$$

그러니까 기존 데이터를 가장 잘 설명하는 Low-rank Matrix를 찾는 것!

아쉽게도 이 문제는 NP-Hard!

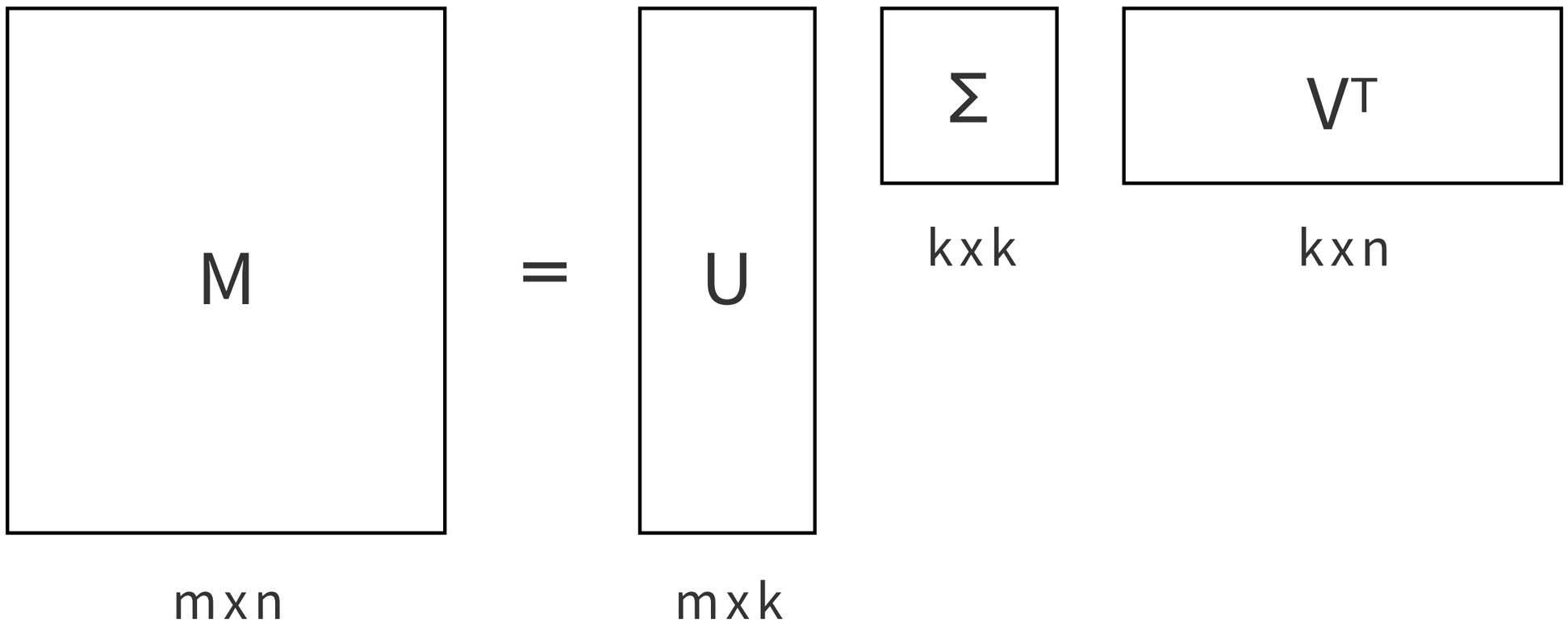
그러면, 어떻게 풀까? **SVD Method.**

SVD Method

Recall: Singular Vector Decomposition(SVD) 이란?

- SVD: 어떤 행렬 M을 $M = U\Sigma V^T$로 분해하는 방법! ⟵ MAS109 선대개 시간에 배움!

만약 행렬 M의 rank가 k라면, 아래와 같이 분해 된다!



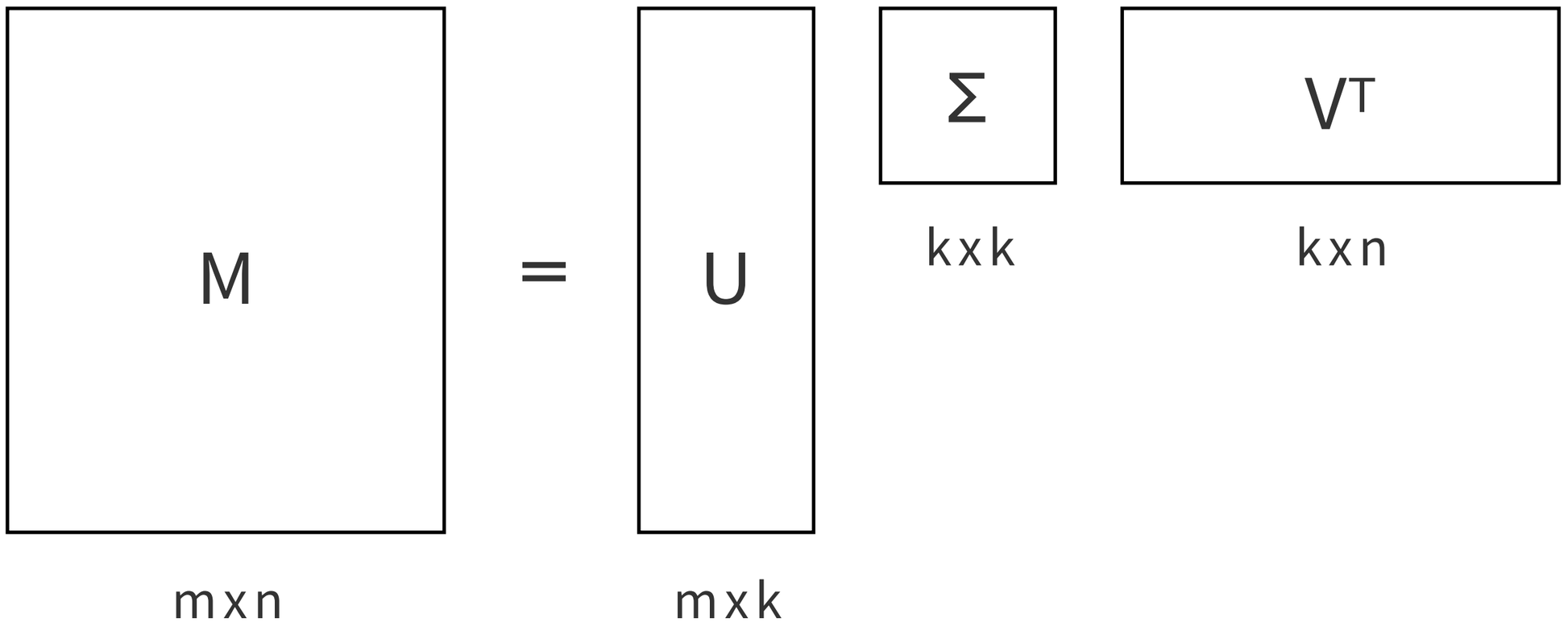
* 여기서 U, V의 column vectors는 orthonormal basis여야 하지만, 논지에 중요치 않아 넘어가도 무방합니다.

SVD Method

Recall: Singular Vector Decomposition(SVD) 이란?

- SVD: 어떤 행렬 M을 $M = U\Sigma V^T$로 분해하는 방법! ← MAS109 선대개 시간에 배움!

만약 행렬 M의 rank가 k라면, 아래와 같이 분해 된다!



여기서 핵심은 행렬 Σ는 **대각 행렬**이라는 것!

$\Sigma_{ij} = 0 \text{ if } i != j$

그리고, $\Sigma_{ii} >= \Sigma_{jj} \text{ if } i >= j$ 인 관계도 만족한다.

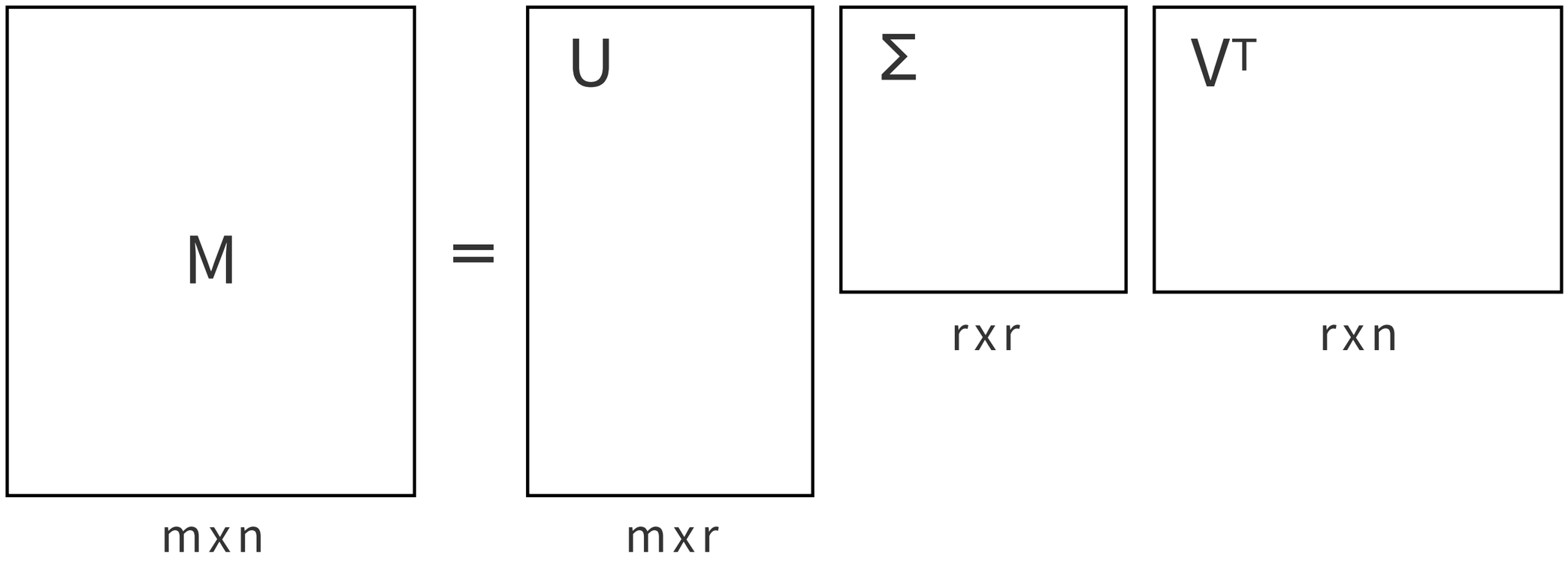
* 여기서 U, V의 column vectors는 orthonormal basis여야 하지만, 논지에 중요치 않아 넘어가도 무방합니다.

SVD Method

Recall: Principal Component Analysis(PCA)

- 만약에 우리가 행렬 M의 모든 entry를 알고 있을 때,
  rank가 r인 행렬 M을 가장 잘 설명하는 rank k 행렬을 찾고 싶다면?

$$\min_{P,Q} \|\mathcal{P}_\Omega(PQ) - \mathcal{P}_\Omega(M)\|_F^2 = \min_{P,Q} \|PQ - M\|_F^2$$

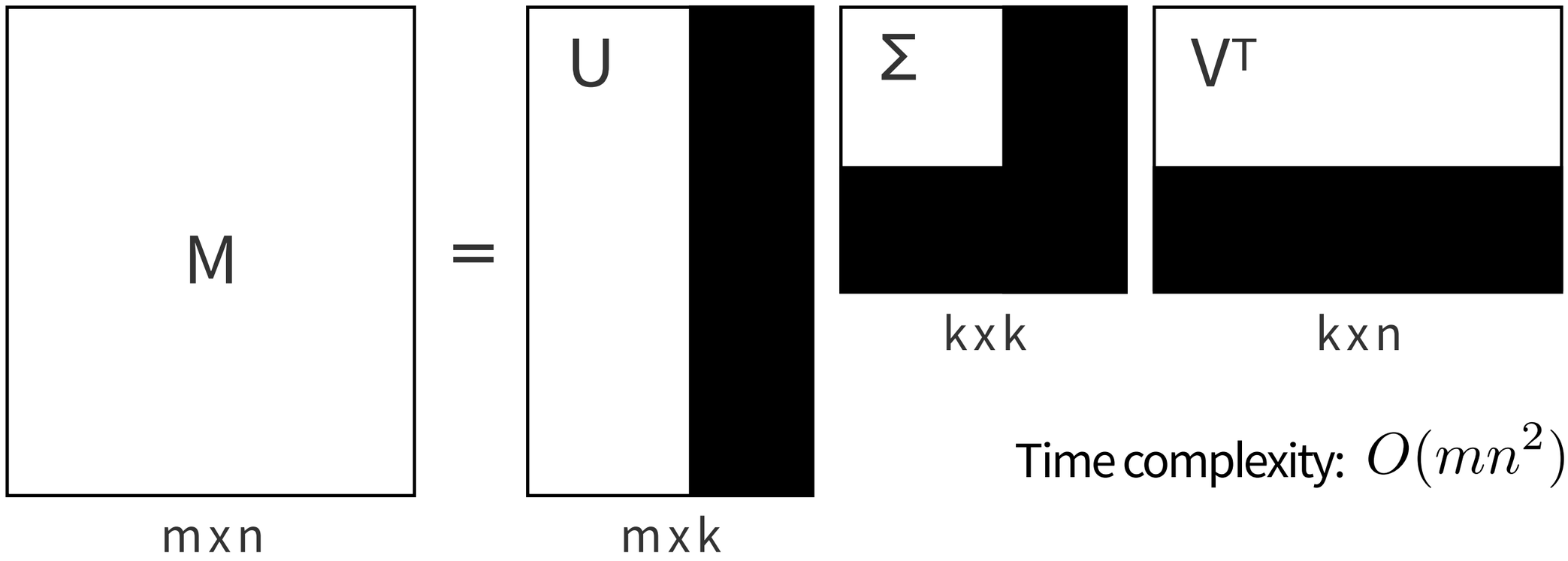


SVD Method

Recall: Principal Component Analysis(PCA)

- 만약에 우리가 행렬 M의 모든 entry를 알고 있을 때,
  rank가 r인 행렬 M을 가장 잘 설명하는 rank k 행렬을 찾고 싶다면?

$$\min_{P,Q} \|\mathcal{P}_\Omega(PQ) - \mathcal{P}_\Omega(M)\|_F^2 = \min_{P,Q} \|PQ - M\|_F^2$$



Time complexity: $O(mn^2)$

행렬 M을 SVD한 이후에 검은 부분을 날리면 된다!

SVD Method

Principal Component Analysis(PCA)는 사용할 수 없다.

- 행렬 M의 모든 entry를 알 때는 PCA를 사용할 수 있지만,
  행렬완성 문제에서는 행렬 M의 일부 정보만 알 수 있으므로 PCA는 사용할 수 없다.

- 행렬완성 문제에서 Global minimum을 구하는 문제는 NP-hard이다.
  그럼 **Local minimum**이라도 찾아야 하지 않을까?

SVD Method

Principal Component Analysis(PCA)는 사용할 수 없다.

- 행렬 M의 모든 entry를 알 때는 PCA를 사용할 수 있지만,
  행렬완성 문제에서는 행렬 M의 일부 정보만 알 수 있으므로 PCA는 사용할 수 없다.

- 행렬완성 문제에서 Global minimum을 구하는 문제는 NP-hard이다.
  그럼 **Local minimum**이라도 찾아야 하지 않을까?

  ⟶ 어떻게? Alternating Minimization

SVD Method

Principal Component Analysis(PCA)는 사용할 수 없다.

- 행렬 M의 모든 entry를 알 때는 PCA를 사용할 수 있지만,
  행렬완성 문제에서는 행렬 M의 일부 정보만 알 수 있으므로 PCA는 사용할 수 없다.

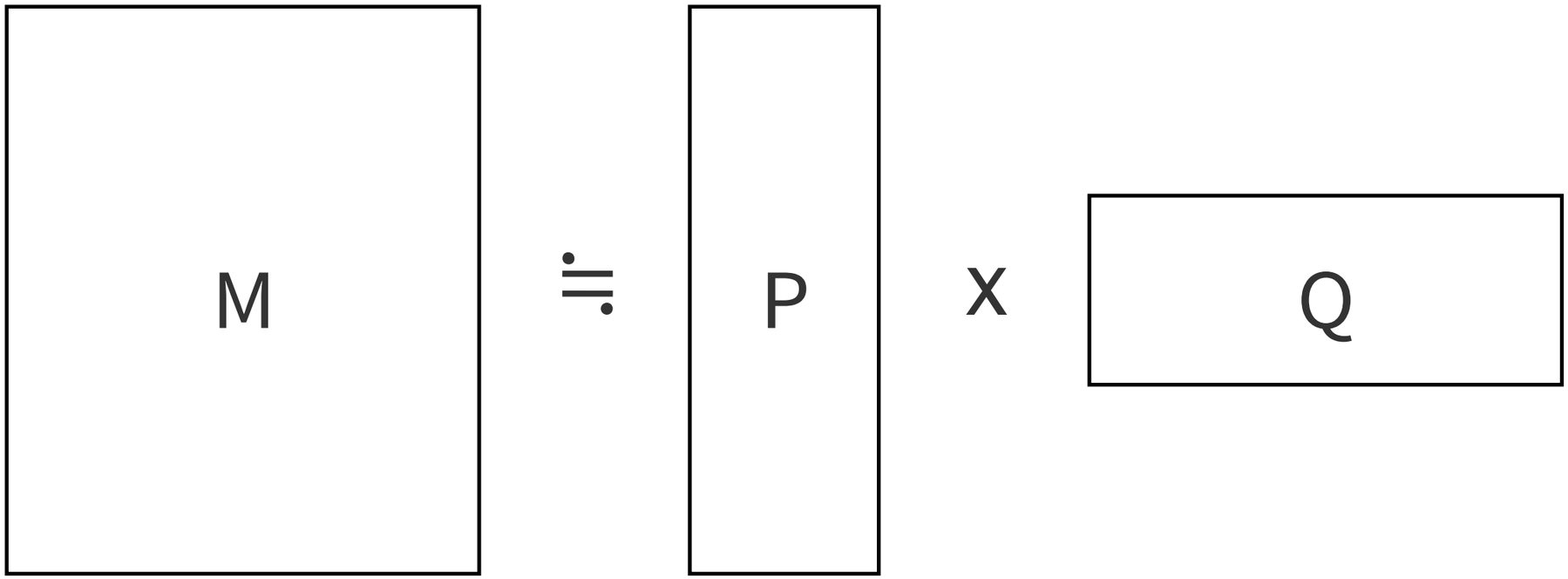
- 행렬완성 문제에서 Global minimum을 구하는 문제는 NP-hard이다.
  그럼 **Local minimum**이라도 찾아야 하지 않을까?

  → 어떻게? Alternating Minimization

  NP-hard 문제를 풀 때 자주 사용되는 method 중 하나! EM도 AM 중 하나!

SVD Method

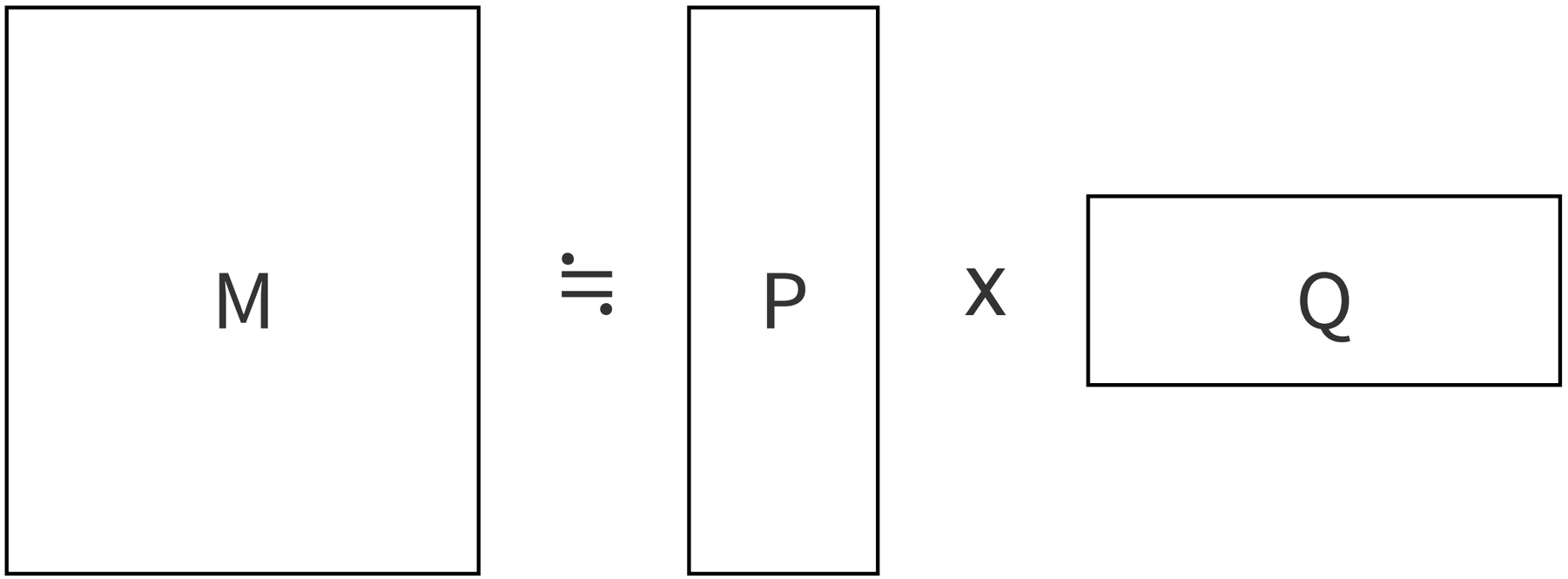
SVD using Alternating Minimization



1. P를 초기화 한다.
2. P가 고정인 상태에서 M을 가장 잘 설명하는 Q를 구한다.
3. Q를 고정시켜 놓고, M을 가장 잘 설명하는 P를 구한다.
4. 수렴할 때 까지 반복한다.

SVD Method

SVD using Alternating Minimization

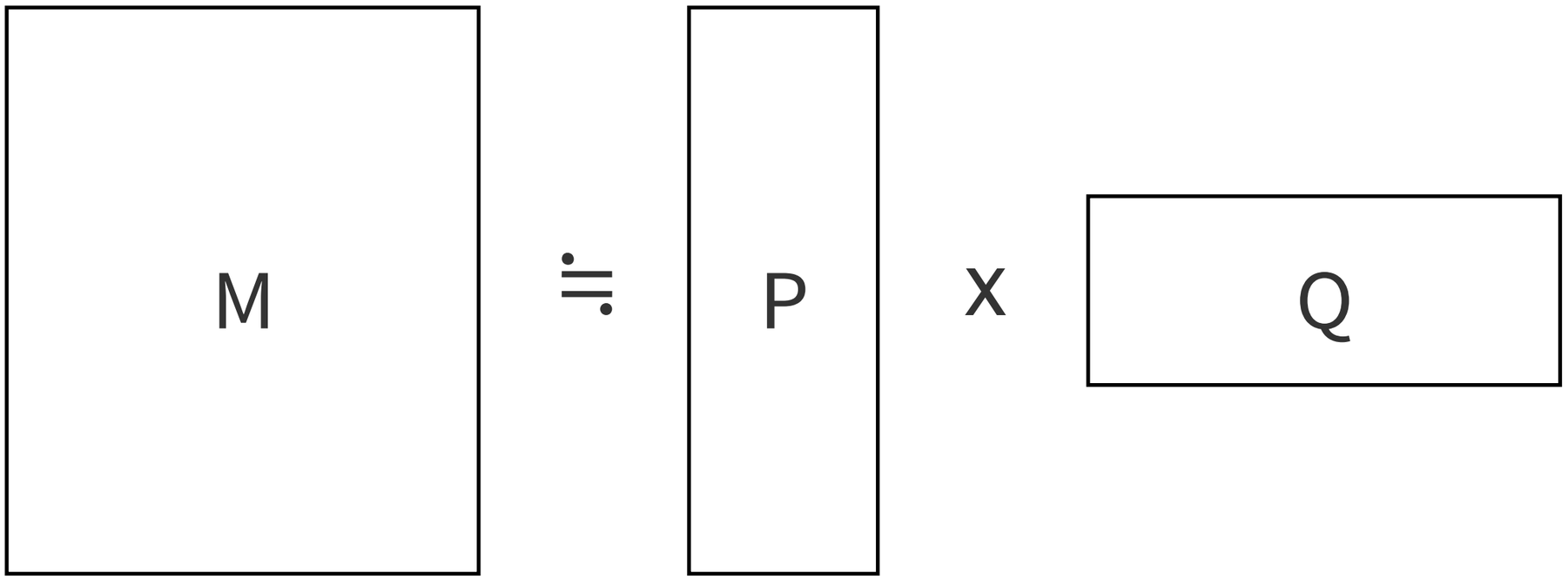


1. P를 초기화 한다. -> 예를 들면, random init.
2. P가 고정인 상태에서 M을 가장 잘 설명하는 Q를 구한다. -> SVD로 풀 수 있음(의사역행렬).
3. Q를 고정시켜 놓고, M을 가장 잘 설명하는 P를 구한다. -> SVD로 풀 수 있음(의사역행렬).
4. 수렴할 때 까지 반복한다.

질문! 그냥 P, Q를 SGD로 구하면 왜 안될까?

SVD Method

SVD using Alternating Minimization

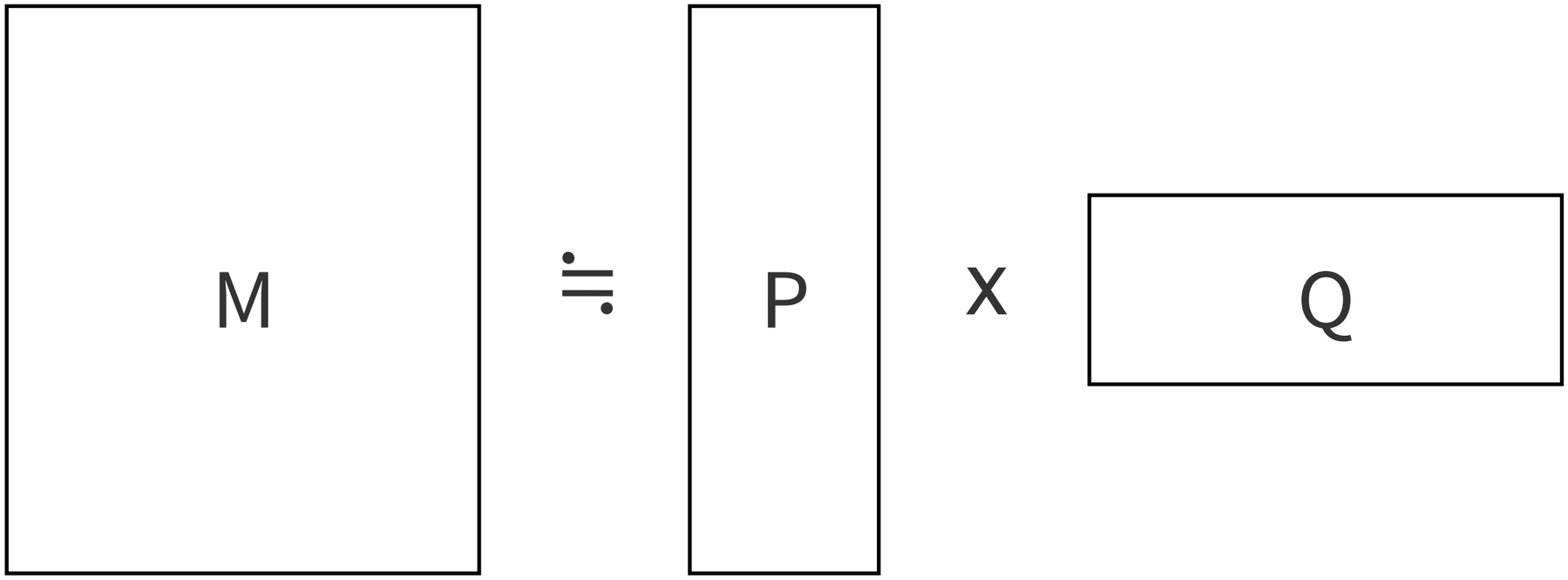


1. P를 초기화 한다. -> 예를 들면, random init.
2. P가 고정인 상태에서 M을 가장 잘 설명하는 Q를 구한다. -> SVD로 풀 수 있음(의사역행렬).
3. Q를 고정시켜 놓고, M을 가장 잘 설명하는 P를 구한다. -> SVD로 풀 수 있음(의사역행렬).
4. 수렴할 때 까지 반복한다.

질문! 그냥 P, Q를 SGD로 구하면 왜 안될까? J. Besag(1986)에서 Overfitting Issue가 있음이 보고!

SVD Method

Algorithm



1: Input: observed set Ω, values $P_\Omega(M)$

2: Initialize P^0 randomly.

3: `for` $t = 1, \cdots, T$:

4: $\quad Q^t \leftarrow \arg\min_{Q \in \mathbb{R}^{k \times n}} \|P_\Omega(M - P^{t-1}Q)\|_F^2$

5: $\quad P^t \leftarrow \arg\min_{P \in \mathbb{R}^{m \times k}} \|P_\Omega(M - PQ^t)\|_F^2$

6: Return $\hat{M} = P^T Q^T$

SVD Method

SVD Method는 Local Optimum에 빠질 수 밖에 없다?

2013년에 Jain 등은 [1] 논문에서 몇 가지 가정 아래에서
SVD Method는 Local Optimum에 빠지지 않는다는 것을 증명.
(물론 알고리즘도 살짝 바꿨긴 했다.)

Algorithm 2 AltMinComplete: Alternating minimization for matrix completion

1: Input: observed set Ω, values $P_\Omega(M)$
2: Partition Ω into $2T+1$ subsets $\Omega_0, \cdots, \Omega_{2T}$ with each element of Ω belonging to one of the Ω_t with equal probability (sampling with replacement)
3: $\widehat{U}^0 = SVD(\frac{1}{p}P_{\Omega_0}(M), k)$ i.e., top-k left singular vectors of $\frac{1}{p}P_{\Omega_0}(M)$
4: Clipping step : Set all elements of $\widehat{U}^0$ that have magnitude greater than $\frac{2\mu\sqrt{k}}{\sqrt{n}}$ to zero and orthonormalize the columns of $\widehat{U}^0$
5: **for** $t = 0, \cdots, T-1$ **do**
6: $\quad \widehat{V}^{t+1} \leftarrow \text{argmin}_{V \in \mathbb{R}^{n \times k}} \ \|P_{\Omega_{t+1}}(\widehat{U}^t V^\dagger - M)\|_F^2$
7: $\quad \widehat{U}^{t+1} \leftarrow \text{argmin}_{U \in \mathbb{R}^{m \times k}} \ \|P_{\Omega_{T+t+1}}(U\left(\widehat{V}^{t+1}\right)^\dagger - M)\|_F^2$
8: **end for**
9: Return $X = \widehat{U}^T(\widehat{V}^T)^\dagger$

SVD Method

SVD Method는 Local Optimum에 빠질 수 밖에 없다?

2013년에 Jain 등은 [1] 논문에서 몇 가지 가정 아래에서
SVD Method는 Local Optimum에 빠지지 않는다는 것을 증명.
(물론 알고리즘도 살짝 바꿨긴 했다.)

Algorithm 2 AltMinComplete: Alternating minimization for matrix completion

1: Input: observed set Ω, values $P_\Omega(M)$
2: Partition Ω into $2T+1$ subsets $\Omega_0, \cdots, \Omega_{2T}$ with each element of Ω belonging to one of the Ω_t with equal probability (sampling with replacement)
3: $\widehat{U}^0 = SVD(\frac{1}{p}P_{\Omega_0}(M), k)$ i.e., top-k left singular vectors of $\frac{1}{p}P_{\Omega_0}(M)$
4: Clipping step : Set all elements of $\widehat{U}^0$ that have magnitude greater than $\frac{2\mu\sqrt{k}}{\sqrt{n}}$ to zero and orthonormalize the columns of $\widehat{U}^0$
5: **for** $t = 0, \cdots, T-1$ **do**
6: $\quad \widehat{V}^{t+1} \leftarrow \text{argmin}_{V \in \mathbb{R}^{n\times k}} \ \|P_{\Omega_{t+1}}(\widehat{U}^t V^\dagger - M)\|_F^2$
7: $\quad \widehat{U}^{t+1} \leftarrow \text{argmin}_{U \in \mathbb{R}^{m\times k}} \ \|P_{\Omega_{T+t+1}}(U\left(\widehat{V}^{t+1}\right)^\dagger - M)\|_F^2$
8: **end for**
9: Return $X = \widehat{U}^T(\widehat{V}^T)^\dagger$

$\hat{U} == P, V^\dagger == Q, V == Q^T$

P^t : t 번째 iteration에서의 행렬 P의 값.

SVD Method

SVD Method는 Local Optimum에 빠질 수 밖에 없다?

2013년에 Jain 등은 [1] 논문에서 몇 가지 가정 아래에서
SVD Method는 Local Optimum에 빠지지 않는다는 것을 증명.
(물론 알고리즘도 살짝 바꿨긴 했다.)

2: Partition Ω into $2T+1$ subsets $\Omega_0, \cdots, \Omega_{2T}$ with each element of Ω belonging to one of the Ω_t with equal probability (sampling with replacement)

어렵게 써있는 것 같지만, mini-batch를 하라는 뜻.

SVD Method

SVD Method는 Local Optimum에 빠질 수 밖에 없다?

2013년에 Jain 등은 [1] 논문에서 몇 가지 가정 아래에서
SVD Method는 Local Optimum에 빠지지 않는다는 것을 증명.
(물론 알고리즘도 살짝 바꿨긴 했다.)

Algorithm 2 AltMinComplete: Alternating minimization for matrix completion

1: Input: observed set Ω, values $P_\Omega(M)$
2: Partition Ω into $2T+1$ subsets $\Omega_0, \cdots, \Omega_{2T}$ with each element of Ω belonging to one of the Ω_t with equal probability (sampling with replacement)
3: $\widehat{U}^0 = SVD(\frac{1}{p}P_{\Omega_0}(M), k)$ i.e., top-k left singular vectors of $\frac{1}{p}P_{\Omega_0}(M)$
4: Clipping step : Set all elements of $\widehat{U}^0$ that have magnitude greater than $\frac{2\mu\sqrt{k}}{\sqrt{n}}$ to zero and orthonormalize the columns of $\widehat{U}^0$

5: **for** $t = 0, \cdots, T-1$ **do**
6: $\quad \widehat{V}^{t+1} \leftarrow \text{argmin}_{V \in \mathbb{R}^{n \times k}} \ \|P_{\Omega_{t+1}}(\widehat{U}^t V^\dagger - M)\|_F^2$
7: $\quad \widehat{U}^{t+1} \leftarrow \text{argmin}_{U \in \mathbb{R}^{m \times k}} \ \|P_{\Omega_{T+t+1}}(U\left(\widehat{V}^{t+1}\right)^\dagger - M)\|_F^2$
8: **end for**
9: Return $X = \widehat{U}^T(\widehat{V}^T)^\dagger$

앞서 설명한 Alternating Minimization!

SVD Method

SVD Method는 Local Optimum에 빠질 수 밖에 없다?

2013년에 Jain 등은 [1] 논문에서 몇 가지 가정 아래에서
SVD Method는 Local Optimum에 빠지지 않는다는 것을 증명.
(물론 알고리즘도 살짝 바꿨긴 했다.)

Algorithm 2 AltMinComplete: Alternating minimization for matrix completion

1: Input: observed set Ω, values $P_\Omega(M)$
2: Partition Ω into $2T+1$ subsets $\Omega_0, \cdots, \Omega_{2T}$ with each element of Ω belonging to one of the Ω_t with equal probability (sampling with replacement)
3: $\widehat{U}^0 = SVD(\frac{1}{p}P_{\Omega_0}(M), k)$ i.e., top-k left singular vectors of $\frac{1}{p}P_{\Omega_0}(M)$
4: Clipping step : Set all elements of $\widehat{U}^0$ that have magnitude greater than $\frac{2\mu\sqrt{k}}{\sqrt{n}}$ to zero and orthonormalize the columns of $\widehat{U}^0$
5: **for** $t = 0, \cdots, T-1$ **do**
6: $\widehat{V}^{t+1} \leftarrow \text{argmin}_{V \in \mathbb{R}^{n\times}}$
7: $\widehat{U}^{t+1} \leftarrow \text{argmin}_{U \in \mathbb{R}^{m\times}}$
8: **end for**
9: Return $X = \widehat{U}^T(\widehat{V}^T)^\dagger$

초기화를 SVD + Clipping을 이용해서 할 때,
SVD method가 찾는 Local Optimum은
Global Optimum에 충분히 가깝다!

SVD Method

SVD Method는 Local Optimum에 빠질 수 밖에 없다?

2013년에 Jain 등은 [1] 논문에서 몇 가지 가정 아래에서
SVD Method는 Local Optimum에 빠지지 않는다는 것을 증명.
(물론 알고리즘도 살짝 바꿨긴 했다.)

3: $\widehat{U}^0 = SVD(\frac{1}{p}P_{\Omega_0}(M), k)$ i.e., top-k left singular vectors of $\frac{1}{p}P_{\Omega_0}(M)$

4: Clipping step : Set all elements of $\widehat{U}^0$ that have magnitude greater than $\frac{2\mu\sqrt{k}}{\sqrt{n}}$ to zero and orthonormalize the columns of $\widehat{U}^0$

Srinadh Bhojanapalli(2016 NIPS) 논문을 통해
이 초기화 과정이 필요 없음이 알려짐!
-> Practical 하게는 Random Initial Point를 설정하면 됨.

SVD Method

SVD Method는 Local Optimum에 빠질 수 밖에 없다?

2013년에 Jain 등은 [1] 논문에서 몇 가지 가정 아래에서
SVD Method는 Local Optimum에 빠지지 않는다는 것을 증명.
(물론 알고리즘도 살짝 바꿨긴 했다.)

Algorithm 2 AltMinComplete: Alternating minimization for matrix completion

1: Input: observed set Ω, values $P_\Omega(M)$
2: Partition Ω into $2T+1$ subsets $\Omega_0, \cdots, \Omega_{2T}$ with each element of Ω belonging to one of the Ω_t with equal probability (sampling with replacement)
3: $\widehat{U}^0 = SVD(\frac{1}{p}P_{\Omega_0}(M), k)$ i.e., top-k left singular vectors of $\frac{1}{p}P_{\Omega_0}(M)$
4: Clipping step : Set all elements of $\widehat{U}^0$ that have magnitude greater than $\frac{2\mu\sqrt{k}}{\sqrt{n}}$ to zero and orthonormalize the columns of $\widehat{U}^0$
5: **for** $t = 0, \cdots, T-1$ **do**
6: $\widehat{V}^{t+1} \leftarrow$ a
7: $\widehat{U}^{t+1} \leftarrow$ a
8: **end for**
9: Return $X =$

Srinadh Bhojanapalli(2016 NIPS) 논문을 통해
이 초기화 과정이 필요 없음이 알려짐!
-> Practical 하게는 Random Initial Point를 설정하면 됨.

* 구현체: https://github.com/JoonyoungYi/MCAM-numpy

Probabilistic Matrix Factorization(PMF)

Low-rank Matrix Completion의 확률적 접근.

- Probabilistic Matrix Factorization(PMF)
  - 2008년 NIPS에 발표된 논문[2].
  - 참고: NIPS에서는 Matrix Factorization으로, ICML에서는 Matrix Completion으로 부르는 경향이 있음.

Probabilistic Matrix Factorization(PMF)

Low-rank Matrix Completion의 확률적 접근.

- Probabilistic Matrix Factorization(PMF)
  - 2008년 NIPS에 발표된 논문[2].
  - 참고: NIPS에서는 Matrix Factorization으로, ICML에서는 Matrix Completion으로 부르는 경향이 있음.

- MAP(Maximum A Posteriori) 관점에서 Low-rank Matrix Completion을 분석.
  - 우리가 관측한 Rating이 Low-rank Structure로 이루어져 있고
    관측과 Latent Variable인 P와 Q에 Gaussian Noise가 존재할 때,
    가장 그럴싸한(=가능성이 높은, =확률이 높은) P와 Q를 찾는 것!

Probabilistic Matrix Factorization(PMF)

Low-rank Matrix Completion의 확률적 접근.

- Probabilistic Matrix Factorization(PMF)
  - 2008년 NIPS에 발표된 논문[2].
  - 참고: NIPS에서는 Matrix Factorization으로, ICML에서는 Matrix Completion으로 부르는 경향이 있음.

- MAP(Maximum A Posteriori) 관점에서 Low-rank Matrix Completion을 분석.
  - 우리가 관측한 Rating이 Low-rank Structure로 이루어져 있고
    관측과 Latent Variable인 P와 Q에 Gaussian Noise가 존재할 때,
    가장 그럴싸한(=가능성이 높은, =확률이 높은) P와 Q를 찾는 것!

- Optimization Form으로 표현하면 다음과 같음.

$$\max_{P,Q} \Pr[P, Q | M, \Omega, \sigma^2, \sigma_P^2, \sigma_Q^2]$$

  - Given: M, Ω, m, n
  - hyperparameters: $\sigma, \sigma_P, \sigma_Q, k$

Probabilistic Matrix Factorization(PMF)

Low-rank Matrix Completion의 확률적 접근.

- Probabilistic Matrix Factorization(PMF)
  - 2008년 NIPS에 발표된 논문[2].
  - 참고: NIPS에서는 Matrix Factorization으로, ICML에서는 Matrix Completion으로 부르는 경향이 있음.

- MAP(Maximum A Posteriori) 관점에서 Low-rank Matrix Completion을 분석.
  - 우리가 관측한 Rating이 Low-rank Structure로 이루어져 있고
    관측과 Latent Variable인 P와 Q에 Gaussian Noise가 존재할 때,
    가장 그럴싸한(=가능성이 높은, =확률이 높은) P와 Q를 찾는 것!

- Optimization Form으로 표현하면 다음과 같음.

$$\max_{P,Q} \Pr[P, Q | M, \Omega, \sigma^2, \sigma_P^2, \sigma_Q^2]$$

사후확률, Objective Function

  - Given: M, Ω, m, n
  - hyperparameters: $\sigma, \sigma_P, \sigma_Q, k$

Probabilistic Matrix Factorization(PMF)

- 우리는 이미 다음과 같은 Low-rank 관계를 알고 있음.

$$\Pr[M|P,Q,\sigma^2] = \prod_{i=1}^{n}\prod_{j=1}^{m}[\mathcal{N}[M_{ij}|P_iQ_j,\sigma^2]]^{I_{ij}^{\Omega}},$$

$$\Pr[P|\sigma_P^2] = \prod_{i=1}^{n}\mathcal{N}[P_i|0,\sigma_P^2\mathbb{I}], \Pr[Q|\sigma_Q^2] = \prod_{j=1}^{m}\mathcal{N}[Q_i|0,\sigma_Q^2\mathbb{I}],$$

$$I_{ij}^{\Omega} = \begin{cases} 1 & (i,j)\in\Omega \\ 0 & \text{otherwise.} \end{cases}$$

Probabilistic Matrix Factorization(PMF)

- 우리는 이미 다음과 같은 Low-rank 관계를 알고 있음.

$$\Pr[M|P,Q,\sigma^2] = \prod_{i=1}^{n}\prod_{j=1}^{m}[\mathcal{N}[M_{ij}|P_iQ_j,\sigma^2]]^{I_{ij}^{\Omega}},$$

$$\Pr[P|\sigma_P^2] = \prod_{i=1}^{n}\mathcal{N}[P_i|0,\sigma_P^2\mathbb{I}], \Pr[Q|\sigma_Q^2] = \prod_{j=1}^{m}\mathcal{N}[Q_i|0,\sigma_Q^2\mathbb{I}],$$

$$I_{ij}^{\Omega} = \begin{cases} 1 & (i,j) \in \Omega \\ 0 & \text{otherwise.} \end{cases}$$

- 앞에서 제시한 objective function(사후 확률)에 위 식들을 넣고 로그 취해 전개하면,

$$\ln \Pr[P,Q|M,\Omega,\sigma^2,\sigma_P^2,\sigma_Q^2]$$

$$= -\frac{1}{2\sigma^2}\sum_{i=1}^{m}\sum_{j=1}^{n}I_{ij}^{\Omega}(M_{ij}-P_iQ_j)^2 - \frac{1}{2\sigma_P^2}\sum_{i=1}^{n}P_i^TP_i - \frac{1}{2\sigma_Q^2}\sum_{j=1}^{m}Q_jQ_j^T$$

$$-\frac{1}{2}((\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbb{I}_{ij}^{\Omega})\ln\sigma^2 + nk\ln\sigma_P^2 + mk\ln\sigma_Q^2) + C$$

Probabilistic Matrix Factorization(PMF)

- 우리는 이미 다음과 같은 Low-rank 관계를 알고 있음.

$$\Pr[M|P,Q,\sigma^2] = \prod_{i=1}^{n}\prod_{j=1}^{m}[\mathcal{N}[M_{ij}|P_iQ_j,\sigma^2]]^{I_{ij}^{\Omega}},$$

$$\Pr[P|\sigma_P^2] = \prod_{i=1}^{n}\mathcal{N}[P_i|0,\sigma_P^2\mathbb{I}], \Pr[Q|\sigma_Q^2] = \prod_{j=1}^{m}\mathcal{N}[Q_i|0,\sigma_Q^2\mathbb{I}],$$

$$I_{ij}^{\Omega} = \begin{cases} 1 & (i,j)\in\Omega \\ 0 & \text{otherwise.} \end{cases}$$

정규 분포 식이 exp form이라 log 취하면 식이 편해짐.

- 앞에서 제시한 objective function(사후 확률)에 위 식들을 넣고 로그 취해 전개하면,

$$\ln\Pr[P,Q|M,\Omega,\sigma^2,\sigma_P^2,\sigma_Q^2]$$

$$= -\frac{1}{2\sigma^2}\sum_{i=1}^{m}\sum_{j=1}^{n}I_{ij}^{\Omega}(M_{ij}-P_iQ_j)^2 - \frac{1}{2\sigma_P^2}\sum_{i=1}^{n}P_i^TP_i - \frac{1}{2\sigma_Q^2}\sum_{j=1}^{m}Q_jQ_j^T$$

$$-\frac{1}{2}((\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbb{I}_{ij}^{\Omega})\ln\sigma^2 + nk\ln\sigma_P^2 + mk\ln\sigma_Q^2) + C$$

Probabilistic Matrix Factorization(PMF)

- 우리는 이미 다음과 같은 Low-rank 관계를 알고 있음.

$$\Pr[M|P,Q,\sigma^2] = \prod_{i=1}^{n}\prod_{j=1}^{m}[\mathcal{N}[M_{ij}|P_iQ_j,\sigma^2]]^{I_{ij}^{\Omega}},$$

$$\Pr[P|\sigma_P^2] = \prod_{i=1}^{n}\mathcal{N}[P_i|0,\sigma_P^2\mathbb{I}], \Pr[Q|\sigma_Q^2] = \prod_{j=1}^{m}\mathcal{N}[Q_i|0,\sigma_Q^2\mathbb{I}],$$

$$I_{ij}^{\Omega} = \begin{cases} 1 & (i,j)\in\Omega \\ 0 & \text{otherwise.} \end{cases}$$

- 앞에서 제시한 objective function(사후 확률)에 위 식들을 넣고 로그 취해 전개하면,

$$\ln\Pr[P,Q|M,\Omega,\sigma^2,\sigma_P^2,\sigma_Q^2]$$

$$= -\frac{1}{2\sigma^2}\sum_{i=1}^{m}\sum_{j=1}^{n}I_{ij}^{\Omega}(M_{ij}-P_iQ_j)^2 - \frac{1}{2\sigma_P^2}\sum_{i=1}^{n}P_i^TP_i - \frac{1}{2\sigma_Q^2}\sum_{j=1}^{m}Q_jQ_j^T$$

$$-\frac{1}{2}((\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbb{I}_{ij}^{\Omega})\ln\sigma^2 + nk\ln\sigma_P^2 + mk\ln\sigma_Q^2) + C$$

P와 Q의 관점에서 상수!

Probabilistic Matrix Factorization(PMF)

- 우리는 이미 다음과 같은 Low-rank 관계를 알고 있음.

$$\Pr[M|P,Q,\sigma^2] = \prod_{i=1}^{n}\prod_{j=1}^{m}[\mathcal{N}[M_{ij}|P_iQ_j,\sigma^2]]^{I_{ij}^{\Omega}},$$

$$\Pr[P|\sigma_P^2] = \prod_{i=1}^{n}\mathcal{N}[P_i|0,\sigma_P^2\mathbb{I}], \Pr[Q|\sigma_Q^2] = \prod_{j=1}^{m}\mathcal{N}[Q_i|0,\sigma_Q^2\mathbb{I}],$$

$$I_{ij}^{\Omega} = \begin{cases} 1 & (i,j)\in\Omega \\ 0 & \text{otherwise.} \end{cases}$$

- 앞에서 제시한 objective function(사후 확률)에 위 식들을 넣고 로그 취해 전개하면,

$$\ln \Pr[P,Q|M,\Omega,\sigma^2,\sigma_P^2,\sigma_Q^2]$$

결국 이걸 maximize 하면 됨!

$$= -\frac{1}{2\sigma^2}\sum_{i=1}^{m}\sum_{j=1}^{n}I_{ij}^{\Omega}(M_{ij}-P_iQ_j)^2 - \frac{1}{2\sigma_P^2}\sum_{i=1}^{n}P_i^TP_i - \frac{1}{2\sigma_Q^2}\sum_{j=1}^{m}Q_jQ_j^T$$

$$-\frac{1}{2}((\sum_{i=1}^{n}\sum_{j=1}^{m}\mathbb{I}_{ij}^{\Omega})\ln\sigma^2 + nk\ln\sigma_P^2 + mk\ln\sigma_Q^2) + C$$

Probabilistic Matrix Factorization(PMF)

$$\arg\max_{P,Q} -\frac{1}{2\sigma^2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{\Omega}(M_{ij}-P_iQ_j)^2 - \frac{1}{2\sigma_P^2}\sum_{i=1}^{n} P_i^T P_i - \frac{1}{2\sigma_Q^2}\sum_{j=1}^{m} Q_j Q_j^T$$

$$= \arg\max_{P,Q} -\frac{1}{2\sigma^2}\|\mathcal{P}_{\Omega}(M-PQ)\|_F^2 - \frac{1}{2\sigma_P^2}\|P\|_F^2 - \frac{1}{2\sigma_Q^2}\|Q\|_F^2$$

$$= \arg\min_{P,Q} \|\mathcal{P}_{\Omega}(M-PQ)\|_F^2 + \frac{\sigma^2}{\sigma_P^2}\|P\|_F^2 + \frac{\sigma^2}{\sigma_Q^2}\|Q\|_F^2$$

$$= \arg\min_{P,Q} \|\mathcal{P}_{\Omega}(M-PQ)\|_F^2 + \lambda_P\|P\|_F^2 + \lambda_Q\|Q\|_F^2$$

$$= \arg\min_{P,Q} \|\mathcal{P}_{\Omega}(M)-\mathcal{P}_{\Omega}(PQ)\|_F^2 + \lambda_P\|P\|_F^2 + \lambda_Q\|Q\|_F^2$$

Probabilistic Matrix Factorization(PMF)

$$\arg\max_{P,Q} -\frac{1}{2\sigma^2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{\Omega}(M_{ij}-P_iQ_j)^2 - \frac{1}{2\sigma_P^2}\sum_{i=1}^{n} P_i^T P_i - \frac{1}{2\sigma_Q^2}\sum_{j=1}^{m} Q_j Q_j^T$$

$$= \arg\max_{P,Q} -\frac{1}{2\sigma^2}\|\mathcal{P}_{\Omega}(M-PQ)\|_F^2 - \frac{1}{2\sigma_P^2}\|P\|_F^2 - \frac{1}{2\sigma_Q^2}\|Q\|_F^2$$

$$= \arg\min_{P,Q} \|\mathcal{P}_{\Omega}(M-PQ)\|_F^2 + \frac{\sigma^2}{\sigma_P^2}\|P\|_F^2 + \frac{\sigma^2}{\sigma_Q^2}\|Q\|_F^2$$

$$= \arg\min_{P,Q} \|\mathcal{P}_{\Omega}(M-PQ)\|_F^2 + \lambda_P\|P\|_F^2 + \lambda_Q\|Q\|_F^2$$

$$= \arg\min_{P,Q} \boxed{\|\mathcal{P}_{\Omega}(M)-\mathcal{P}_{\Omega}(PQ)\|_F^2} + \lambda_P\|P\|_F^2 + \lambda_Q\|Q\|_F^2$$

SVD using Alternating Minimization에서 풀고자하는 문제 와 동일!

Probabilistic Matrix Factorization(PMF)

$$\arg\max_{P,Q} -\frac{1}{2\sigma^2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{\Omega}(M_{ij}-P_iQ_j)^2 - \frac{1}{2\sigma_P^2}\sum_{i=1}^{n} P_i^T P_i - \frac{1}{2\sigma_Q^2}\sum_{j=1}^{m} Q_j Q_j^T$$

$$= \arg\max_{P,Q} -\frac{1}{2\sigma^2}\|\mathcal{P}_{\Omega}(M-PQ)\|_F^2 - \frac{1}{2\sigma_P^2}\|P\|_F^2 - \frac{1}{2\sigma_Q^2}\|Q\|_F^2$$

$$= \arg\min_{P,Q} \|\mathcal{P}_{\Omega}(M-PQ)\|_F^2 + \frac{\sigma^2}{\sigma_P^2}\|P\|_F^2 + \frac{\sigma^2}{\sigma_Q^2}\|Q\|_F^2$$

$$= \arg\min_{P,Q} \|\mathcal{P}_{\Omega}(M-PQ)\|_F^2 + \lambda_P\|P\|_F^2 + \lambda_Q\|Q\|_F^2$$

$$= \arg\min_{P,Q} \|\mathcal{P}_{\Omega}(M) - \mathcal{P}_{\Omega}(PQ)\|_F^2 + \boxed{\lambda_P\|P\|_F^2 + \lambda_Q\|Q\|_F^2}$$

이 부분만 추가하면, noise가 Gaussian Model일 때를 처리할 수 있게 됨!

Probabilistic Matrix Factorization(PMF)

$$\arg\max_{P,Q} -\frac{1}{2\sigma^2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{\Omega}(M_{ij}-P_iQ_j)^2 - \frac{1}{2\sigma_P^2}\sum_{i=1}^{n} P_i^T P_i - \frac{1}{2\sigma_Q^2}\sum_{j=1}^{m} Q_j Q_j^T$$

$$= \arg\max_{P,Q} -\frac{1}{2\sigma^2}\|\mathcal{P}_{\Omega}(M-PQ)\|_F^2 - \frac{1}{2\sigma_P^2}\|P\|_F^2 - \frac{1}{2\sigma_Q^2}\|Q\|_F^2$$

$$= \arg\min_{P,Q} \|\mathcal{P}_{\Omega}(M-PQ)\|_F^2 + \frac{\sigma^2}{\sigma_P^2}\|P\|_F^2 + \frac{\sigma^2}{\sigma_Q^2}\|Q\|_F^2$$

$$= \arg\min_{P,Q} \|\mathcal{P}_{\Omega}(M-PQ)\|_F^2 + \lambda_P\|P\|_F^2 + \lambda_Q\|Q\|_F^2$$

$$= \arg\min_{P,Q} \|\mathcal{P}_{\Omega}(M) - \mathcal{P}_{\Omega}(PQ)\|_F^2 + \boxed{\lambda_P\|P\|_F^2 + \lambda_Q\|Q\|_F^2}$$

이 부분만 추가하면, noise가 Gaussian Model일 때를 처리할 수 있게 됨!

노이즈 처리를 위한 정규화 term으로 보면 됨!

(딥러닝에서도 종종 사용됩니다.)

Biased Matrix Factorization(Biased-MF)

PMF의 문제점: User와 Movie의 Rating 경향성이 반영되지 않음

- 어떤 유저는 rating을 높게 주는 경향이 있고, 어떤 유저는 rating을 낮게 주는 경향이 있을 수 있음
  - 영화도 마찬가지!
    어떤 영화는 rating을 높게 받는 경향이 있고, 어떤 영화는 rating을 낮게 받는 경향이 있음!

Biased Matrix Factorization

- Netflix Prize Winner의 아이디어[3]: 이런 영화와 유저의 Bias를 핸들링 해보자!
  - 딥러닝 네트워크 학습시킬 때 등장하는 $\sigma(Wx+b)$ 에서의 bias term과 동일한 역할!
  - 이 알고리즘이 전체 10% 향상 중 8% 향상을 가져왔음!

Biased Matrix Factorization(Biased-MF)

Biased Matrix Factorization

- Netflix Prize Winner의 아이디어[3]: 이런 영화와 유저의 Bias를 핸들링 해보자!
  - 딥러닝 네트워크 학습시킬 때 등장하는 σ(Wx + b) 에서의 bias term과 동일한 역할!
  - 이 알고리즘이 전체 10% 향상 중 8% 향상을 가져왔음!

- (Recall) PMF의 Optimization Form:

$$\arg\min_{P,Q} \|\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(PQ)\|_F^2 + \lambda_P \|P\|_F^2 + \lambda_Q \|Q\|_F^2$$

$$= \arg\min_{P,Q} \sum_{(i,j)\in\Omega} (M_{ij} - P_i \cdot Q_j)^2 + \lambda_P \|P_i\|_2^2 + \lambda_Q \|Q_j\|_2^2$$

Biased Matrix Factorization(Biased-MF)

Biased Matrix Factorization

- Netflix Prize Winner의 아이디어[3]: 이런 영화와 유저의 Bias를 핸들링 해보자!
  - 딥러닝 네트워크 학습시킬 때 등장하는 σ(Wx + b) 에서의 bias term과 동일한 역할!
  - 이 알고리즘이 전체 10% 향상 중 8% 향상을 가져왔음!

- (Recall) PMF의 Optimization Form:

$$\begin{aligned}
&\arg\min_{P,Q} \|\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(PQ)\|_F^2 + \lambda_P \|P\|_F^2 + \lambda_Q \|Q\|_F^2 \\
&= \arg\min_{P,Q} \sum_{(i,j)\in\Omega} (M_{ij} - P_i \cdot Q_j)^2 + \lambda_P \|P_i\|_2^2 + \lambda_Q \|Q_j\|_2^2
\end{aligned}$$

- Biased Matrix Factorization의 Optimization Form:

$$\begin{aligned}
&\arg\min_{P,Q,b^{user},b^{item}} \sum_{(i,j)\in\Omega} (M_{ij} - \mu - b_i^{user} - b_j^{item} - P_i \cdot Q_j)^2 \\
&\qquad + \lambda_P \|P_i\|_2^2 + \lambda_Q \|Q_j\|_2^2 + \lambda_{b^{user}} b_i^2 + \lambda_{b^{item}} b_j^2 \\
&= \arg\min_{P,Q,b^{user},b^{item}} \sum_{(i,j)\in\Omega} (M_{ij} - \mu - b_i^{user} - b_j^{item} - P_i \cdot Q_j)^2 \\
&\qquad + \lambda(\|P_i\|_2^2 + \|Q_j\|_2^2 + (b_i^{user})^2 + (b_j^{item})^2)
\end{aligned}$$

Biased Matrix Factorization(Biased-MF)

Implementation

```
u = tf.placeholder(tf.int32, [None], name='u')
i = tf.placeholder(tf.int32, [None], name='i')
r = tf.placeholder(tf.float32, [None], name='r')

p = tf.Variable(tf.random_normal([N, K]) / np.sqrt(N))  # p latent matrix
q = tf.Variable(tf.random_normal([M, K]) / np.sqrt(M))  # q latent matrix
p_lookup = tf.nn.embedding_lookup(p, u)
q_lookup = tf.nn.embedding_lookup(q, i)

mu = tf.reduce_mean(r)
b_u = tf.Variable(tf.zeros([N]))
b_i = tf.Variable(tf.zeros([M]))
b_u_lookup = tf.nn.embedding_lookup(b_u, u)
b_i_lookup = tf.nn.embedding_lookup(b_i, i)

b_ui = mu + tf.add(b_u_lookup, b_i_lookup)
r_ui_hat = tf.add(b_ui, tf.reduce_sum(tf.multiply(p_lookup, q_lookup), 1))
```

Biased Matrix Factorization(Biased-MF)

Implementation

```
reconstruction_loss = tf.reduce_sum(
    tf.square(tf.subtract(r, r_ui_hat)), reduction_indices=[0])
regularizer_loss = tf.add_n([
    tf.reduce_sum(tf.square(p)),
    tf.reduce_sum(tf.square(q)),
    tf.reduce_sum(tf.square(b_u)),
    tf.reduce_sum(tf.square(b_i)),
])
loss = tf.add(reconstruction_loss, lambda_value * regularizer_loss)

rmse = tf.sqrt(tf.reduce_mean(tf.square(tf.subtract(r, r_ui_hat))))

optimizer = tf.train.GradientDescentOptimizer(1e-4)
train_op = optimizer.minimize(loss, var_list=[b_u, b_i, p, q])
```

- 구현체: https://github.com/JoonyoungYi/BiasedMF-tensorflow

Neural Network Matrix Factorization(NNMF)

함수 관점에서 재서술한 Biased-MF

- Biased MF에서 예상별점은 다음과 같이 계산된다.

$$\hat{M}_{ij} = \mu + b_i^{user} + b_j^{item} + P_i \cdot Q_j$$

Neural Network Matrix Factorization(NNMF)

함수 관점에서 재서술한 Biased-MF

- Biased MF에서 예상별점은 다음과 같이 계산된다.

$$\hat{M}_{ij} = \mu + b_i^{user} + b_j^{item} + P_i \cdot Q_j$$

- Biased MF에서 예상별점을 다음과 같이 표현할 수도 있을 것이다.

$$\hat{M}_{ij} = f_{\text{Biased-MF}}(b_i^{user}, b_j^{item}, P_i \cdot Q_j),$$

$$f_{\text{Biased-MF}}(b_i^{user}, b_j^{item}, P_i \cdot Q_j) = \mu + b_i^{user} + b_j^{item} + P_i \cdot Q_j$$

Neural Network Matrix Factorization(NNMF)

함수 관점에서 재서술한 Biased-MF

- Biased MF에서 예상별점은 다음과 같이 계산된다.

$$\hat{M}_{ij} = \mu + b_i^{user} + b_j^{item} + P_i \cdot Q_j$$

- Biased MF에서 예상별점을 다음과 같이 표현할 수도 있을 것이다.

$$\hat{M}_{ij} = f_{\text{Biased-MF}}(b_i^{user}, b_j^{item}, P_i \cdot Q_j),$$

$$f_{\text{Biased-MF}}(b_i^{user}, b_j^{item}, P_i \cdot Q_j) = \mu + b_i^{user} + b_j^{item} + P_i \cdot Q_j$$

- 한편, Universial Approximation Theorem에 의해
  딥 네트워크가 어떤 함수던 잘 모사할 수 있음이 알려져 있다.

Neural Network Matrix Factorization(NNMF)

함수 관점에서 재서술한 Biased-MF

- Biased MF에서 예상별점은 다음과 같이 계산된다.

$$\hat{M}_{ij} = \mu + b_i^{user} + b_j^{item} + P_i \cdot Q_j$$

- Biased MF에서 예상별점을 다음과 같이 표현할 수도 있을 것이다.

$$\hat{M}_{ij} = f_{\text{Biased-MF}}(b_i^{user}, b_j^{item}, P_i \cdot Q_j),$$

$$f_{\text{Biased-MF}}(b_i^{user}, b_j^{item}, P_i \cdot Q_j) = \mu + b_i^{user} + b_j^{item} + P_i \cdot Q_j$$

- 한편, Universial Approximation Theorem에 의해
  딥 네트워크가 어떤 함수던 잘 모사할 수 있음이 알려져 있다.

- Neural Network Matrix Factorization의 메인 아이디어:

$f_{\text{Biased-MF}}$를 **딥 네트워크로 교체**해보면 어떨까?

Neural Network Matrix Factorization(NNMF)

Biased MF으로부터 시작

- Biased MF의 Latent Variable들을 확장하자!

$$\hat{M}_{ij} = f_{\text{Biased-MF}}(b_i^{user}, b_j^{item}, P_i \cdot Q_j)$$

$$\rightarrow \hat{M}_{ij} = f_{\theta}(b_i^{user}, b_j^{item}, P_i \cdot Q_j)$$

f 함수를 Deep Neural Network로 교체!

Neural Network Matrix Factorization(NNMF)

Biased MF으로부터 시작

- Biased MF의 Latent Variable들을 확장하자!

$$\hat{M}_{ij} = f_{\text{Biased-MF}}(b_i^{user}, b_j^{item}, P_i \cdot Q_j)$$

$$\rightarrow \hat{M}_{ij} = f_\theta(b_i^{user}, b_j^{item}, P_i \cdot Q_j)$$

$$\rightarrow \hat{M}_{ij} = f_\theta(B_i^{user}, B_j^{item}, P_i \cdot Q_j)$$

Bias 에 해당하는 Latent Variable을 scalar에서 vector로 확장!

Neural Network Matrix Factorization(NNMF)

Biased MF으로부터 시작

- Biased MF의 Latent Variable들을 확장하자!

$$\hat{M}_{ij} = f_{\text{Biased-MF}}(b_i^{user}, b_j^{item}, P_i \cdot Q_j)$$

$$\rightarrow \hat{M}_{ij} = f_\theta(b_i^{user}, b_j^{item}, P_i \cdot Q_j)$$

$$\rightarrow \hat{M}_{ij} = f_\theta(B_i^{user}, B_j^{item}, P_i \cdot Q_j)$$

$$\rightarrow \hat{M}_{ij} = f_\theta(B_i^{user}, B_j^{item}, P_{i,1} \cdot Q_{j,1}, \cdots, P_{i,D} \cdot Q_{j,D})$$

Low-rank Matrix 의 갯수를 D개로 증가!

Neural Network Matrix Factorization(NNMF)

Biased MF으로부터 시작

- Biased MF의 Latent Variable들을 확장하자!

$$\hat{M}_{ij} = f_{\text{Biased-MF}}(b_i^{user}, b_j^{item}, P_i \cdot Q_j)$$

$$\rightarrow \hat{M}_{ij} = f_\theta(b_i^{user}, b_j^{item}, P_i \cdot Q_j)$$

$$\rightarrow \hat{M}_{ij} = f_\theta(B_i^{user}, B_j^{item}, P_i \cdot Q_j)$$

$$\rightarrow \hat{M}_{ij} = f_\theta(B_i^{user}, B_j^{item}, P_{i,1} \cdot Q_{j,1}, \cdots, P_{i,D} \cdot Q_{j,D})$$

- 딥네트워크 f_θ를 학습시키기 위한 Loss Function 디자인:

$$\sum_{(i,j)\in\Omega} (\hat{M}_{ij} - M_{ij})^2 + \lambda[\sum_i \|B_i^{user}\|_F^2 + \sum_i \|P_i\|_F^2 + \sum_j \|B_j^{item}\|_F^2 + \sum_j \|Q_j\|_F^2]$$

Neural Network Matrix Factorization(NNMF)

Biased MF으로부터 시작

- Biased MF의 Latent Variable들을 확장하자!

$$\hat{M}_{ij} = f_{\text{Biased-MF}}(b_i^{user}, b_j^{item}, P_i \cdot Q_j)$$

$$\rightarrow \hat{M}_{ij} = f_\theta(b_i^{user}, b_j^{item}, P_i \cdot Q_j)$$

$$\rightarrow \hat{M}_{ij} = f_\theta(B_i^{user}, B_j^{item}, P_i \cdot Q_j)$$

$$\rightarrow \hat{M}_{ij} = f_\theta(B_i^{user}, B_j^{item}, P_{i,1} \cdot Q_{j,1}, \cdots, P_{i,D} \cdot Q_{j,D})$$

- 딥네트워크 f_θ를 학습시키기 위한 Loss Function 디자인:

$$\sum_{(i,j)\in\Omega} (\hat{M}_{ij} - M_{ij})^2 + \lambda[\sum_i \|B_i^{user}\|_F^2 + \sum_i \|P_i\|_F^2 + \sum_j \|B_j^{item}\|_F^2 + \sum_j \|Q_j\|_F^2]$$

- 구현체: https://github.com/JoonyoungYi/NNMF-tensorflow

Conclusion

정리

- SVD using Alternating Minimization

$$\min_{P,Q} \|\mathcal{P}_\Omega(PQ) - \mathcal{P}_\Omega(M)\|_F^2$$

- PMF(Probabilistic Matrix Factorization)

$$\arg\min_{P,Q} \sum_{(i,j)\in\Omega} (M_{ij} - P_i \cdot Q_j)^2 + \lambda_P \|P_i\|_2^2 + \lambda_Q \|Q_j\|_2^2$$

- Biased-MF(Biased Matrix Factorization)

$$\arg\min_{P,Q,b^{user},b^{item}} \sum_{(i,j)\in\Omega} (M_{ij} - \mu - b_i^{user} - b_j^{item} - P_i \cdot Q_j)^2 + \lambda(\|P_i\|_2^2 + \|Q_j\|_2^2 + (b_i^{user})^2 + (b_j^{item})^2)$$

- NNMF(Neural Network Matrix Factorization)

$$\sum_{(i,j)\in\Omega} (\hat{M}_{ij} - M_{ij})^2 + \lambda[\sum_i \|B_i^{user}\|_F^2 + \sum_i \|P_i\|_F^2 + \sum_j \|B_j^{item}\|_F^2 + \sum_j \|Q_j\|_F^2]$$

$$\hat{M}_{ij} = f_\theta(B_i^{user}, B_j^{item}, P_{i,1} \cdot Q_{j,1}, \cdots, P_{i,D} \cdot Q_{j,D})$$

NAME | 이준영(Joonyoung Yi)
EMAIL | joonyoung.yi@mli.kaist.ac.kr joonyoung.yi@kaist.ac.kr
PHONE | +82-10-9765-0885

Appendix

Reference

[1] Jain, Prateek, Praneeth Netrapalli, and Sujay Sanghavi. "Low-rank matrix completion using alternating minimization." Proceedings of the forty-fifth annual ACM symposium on Theory of computing. ACM, 2013.

[2] Mnih, Andriy, and Ruslan R. Salakhutdinov. "Probabilistic matrix factorization." Advances in neural information processing systems. 2008.

[3] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 8 (2009): 30-37.

[4] Dziugaite, Gintare Karolina, and Daniel M. Roy. "Neural network matrix factorization." arXiv preprint arXiv:1511.06443 (2015).