비즈니스 Case Study를 통한

추천 시스템 구현

3강

머신러닝을 활용한

모델 기반 협업 필터링

3강

머신러닝을 활용한 모델 기반 협업 필터링

Contents

1. 모델 기반 협업 필터링이란?

Collaborative Filtering의 분류

Neighborhood-based Collaborative Filtering

- User-based
- Item-based

Model-based Collaborative Filtering

- Singular Value Decomposition
- Matrix Factorization (SGD, ALS, BPR)
- Probabilistic Model
- Deep Learning

Hybrid Collaborative Filtering

- Content-based Recommendation과의 결합

Collaborative Filtering의 분류

Neighborhood-based Collaborative Filtering

- User-based
- Item-based

Model-based Collaborative Filtering

- Singular Value Decomposition
- Matrix Factorization (SGD, ALS, BPR)
- Probabilistic Model
- Deep Learning

Hybrid Collaborative Filtering

- Content-based Recommendation과의 결합

이웃 기반 Collaborative Filtering (2강 복습)

Memory-based Collaborative Filtering이라고도 불리며,

유저와 아이템을 직접 연관시켜 추천하기 때문에 설명력이 높고 적용이 용이함

특징

- 쉽고 구현이 간단함
- 유저/아이템 간의 유사도에 크게 의존하는 기법
- 데이터의 sparsity에 취약함
  - 그러나 대부분 real word 데이터는 sparse함
- 추천 결과를 생성할 때마다 많은 연산을 요구함
  - 유저, 아이템이 늘어날수록 확장성(scalability)이 떨어짐

2강 복습

유저 $u \in U$, 아이템 $i \in I$ 에 대해 평점 데이터 $r(u,i)$ 가 존재할 때, 유저 u의 아이템 i에 대한 평점을 예측해보자

1) user-based: 아이템 i 에 대한 평점이 있으면서 유저 u 와 유사한 유저들의 집합을 Ω_i라고 하면,

$$\hat{r}(u,i) = \frac{\sum_{u' \in \Omega_i} sim(u,u')r(u',i)}{\sum_{u' \in \Omega_i} sim(u,u')}$$

2) item-based: 유저 u 가 평가를 한 다른 아이템 중에서 아이템 i와 유사한 아이템들의 집합을 Φ_u 라고 하면,

$$\hat{r}(u,i) = \frac{\sum_{i' \in \Phi_u} sim(i,i')r(u,i')}{\sum_{i' \in \Phi_u} sim(i,i')}$$

Model-based Collaborative Filtering 특징

Parametric Machine Learning을 사용하는가?

주어진 데이터를 사용하여 모델을 학습

데이터 정보가 파라미터의 형태로 모델에 압축

모델의 파라미터는 데이터의 패턴을 나타내고, 최적화를 통해 업데이트

데이터의 패턴 = 유저/아이템의 잠재적 특성

이웃 기반 CF은 유저/아이템 벡터를 데이터를 통해 계산된 형태로 저장하고 있지만,

Model-based CF의 경우 유저, 아이템 벡터는 모두 학습을 통해 변하는 파라미터임

Real world에서 Matrix Factorization 기법이 가장 많이 사용됨

최근에 MF 원리를 Deep Learning에 응용한 모델이 더 높은 성능을 냄

Model-based Collaborative Filtering 장점

모델의 학습/서빙

유저-아이템 데이터는 학습에만 사용되고 학습된 모델은 압축된 형태로 저장됨

이미 학습된 모델을 통해 추천 결과를 서빙하기 때문에 속도가 빠름

Sparsity/Scalability 극복

이웃 기반 CF에 비해 sparse한 데이터에서도 좋은 성능을 보임

사용자, 아이템 개수가 늘어나도 좋은 추천 성능을 보임

Overfitting 방지

이웃 기반 CF와 비교했을 때 전체 데이터의 패턴을 학습하도록 모델이 작동함

Limited Coverage

이웃 기반 CF의 경우 공통의 유저 / 아이템을 많이 공유해야만 유사도 값이 정확해짐

유사도 값이 정확하지 않은 경우 이웃의 효과를 보기 어려움

Rating Matrix (Explicit Feedback)

	I_1	I_2	I_3	I_4
U_1	5		?	4
U_3		4		
U_3	3			1
U_4		?	2	

	I_1	I_2	I_3	I_4
U_1	5		4	4
U_3	1	4		
U_3	3			1
U_4		3	2	

User-Item Matrix (Implicit Feedback)

	I_1	I_2	I_3	I_4
U_1	O		?	O
U_3		O		
U_3	O			O
U_4		?	O	

	I_1	I_2	I_3	I_4
U_1	O		O	O
U_3		O		
U_3	O			O
U_4		X	O	

2. Latent Factor Model과 SVD

Latent Factor Model

유저와 아이템을 잠재적 요인을 사용해 표현할 수 있다고 보는 모델

다양하고 복잡한 유저와 아이템의 특성을 몇 개의 벡터로 compact하게 표현

유저와 아이템을 같은 차원의 벡터로 표현하여 나타냄

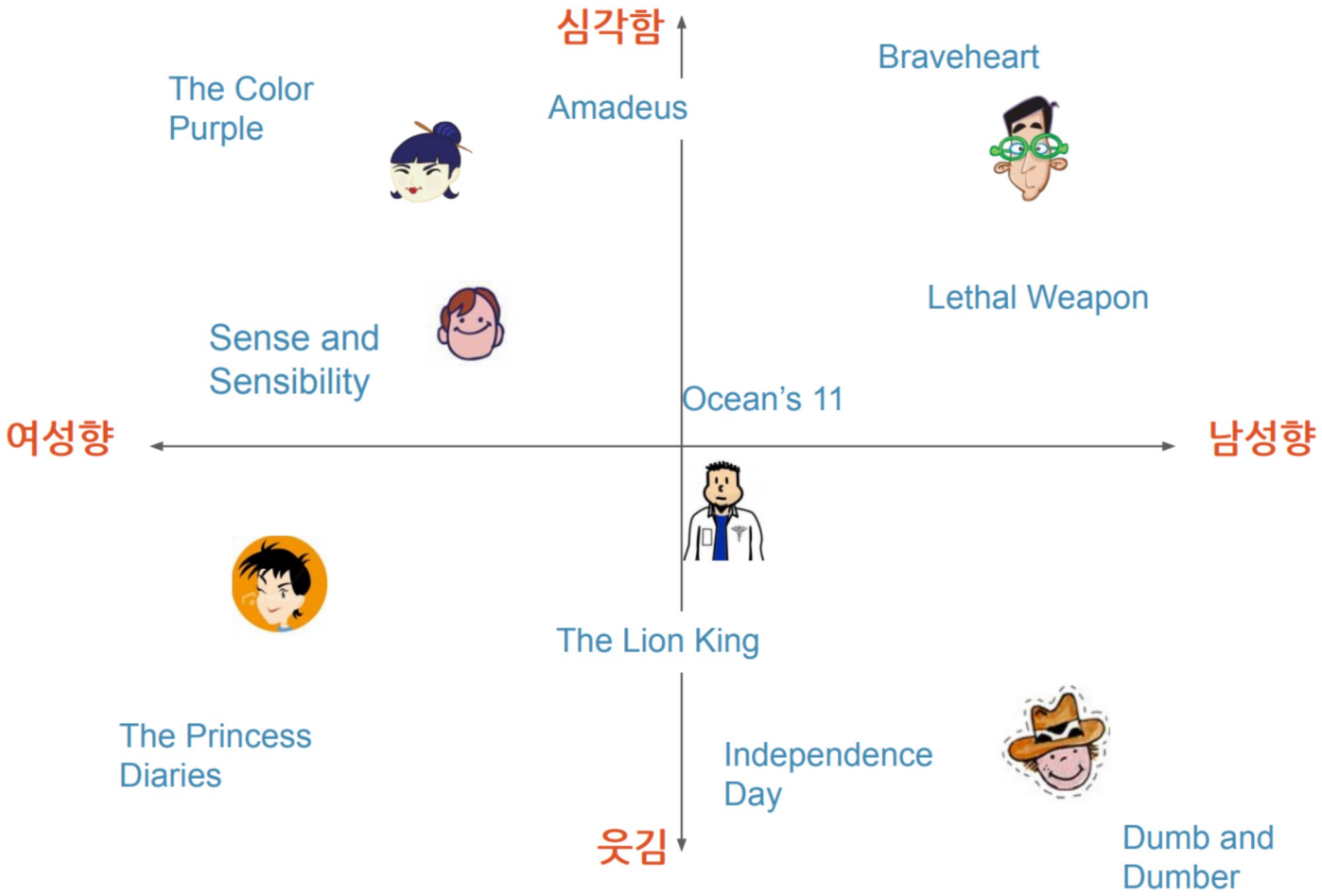
차원의 개수는 여러개

각 차원의 의미는 모델 학습을 통해 생성되며 표면적으로 알 수 없음

같은 벡터 공간에서 유저와 아이템 벡터가 놓일 경우 유저와 아이템 유사한 정도를 확인할 수 있음

유저 벡터와 아이템 벡터가 유사한 경우 추천이 될 가능성이 높음

Latent Factor Model



Singular Value Decomposition

Rating Matrix R 에 대해 유저와 아이템의 잠재 요인을 포함할 수 있는 행렬로 분해

1) 유저 잠재 요인 행렬, 2) 잠재 요인 대각행렬, 3) 아이템 잠재 요인 행렬

선형 대수학에서 차원 축소 기법 중 하나로 분류됨

주성분분석(PCA)도 차원 축소 기법 중 하나

Netflix Prize에서 추천 시스템에 적용된 단일 알고리즘으로 가장 좋은 성능을 보임

현재는 SVD의 원리를 차용하되 다양한 최적화 기법을 적용한 MF가 더 많이 사용됨 (SGD, ALS, BPR 등)

Singular Value Decomposition

Full SVD: $R = U\Sigma V^T$

U: 유저의 Latent Factor

U의 열벡터는 R 의 left singular vector

V: 아이템의 Latent Factor

V 의 열벡터는 R 의 right singular vector

Σ: Latent Factor의 중요도를 나타냄

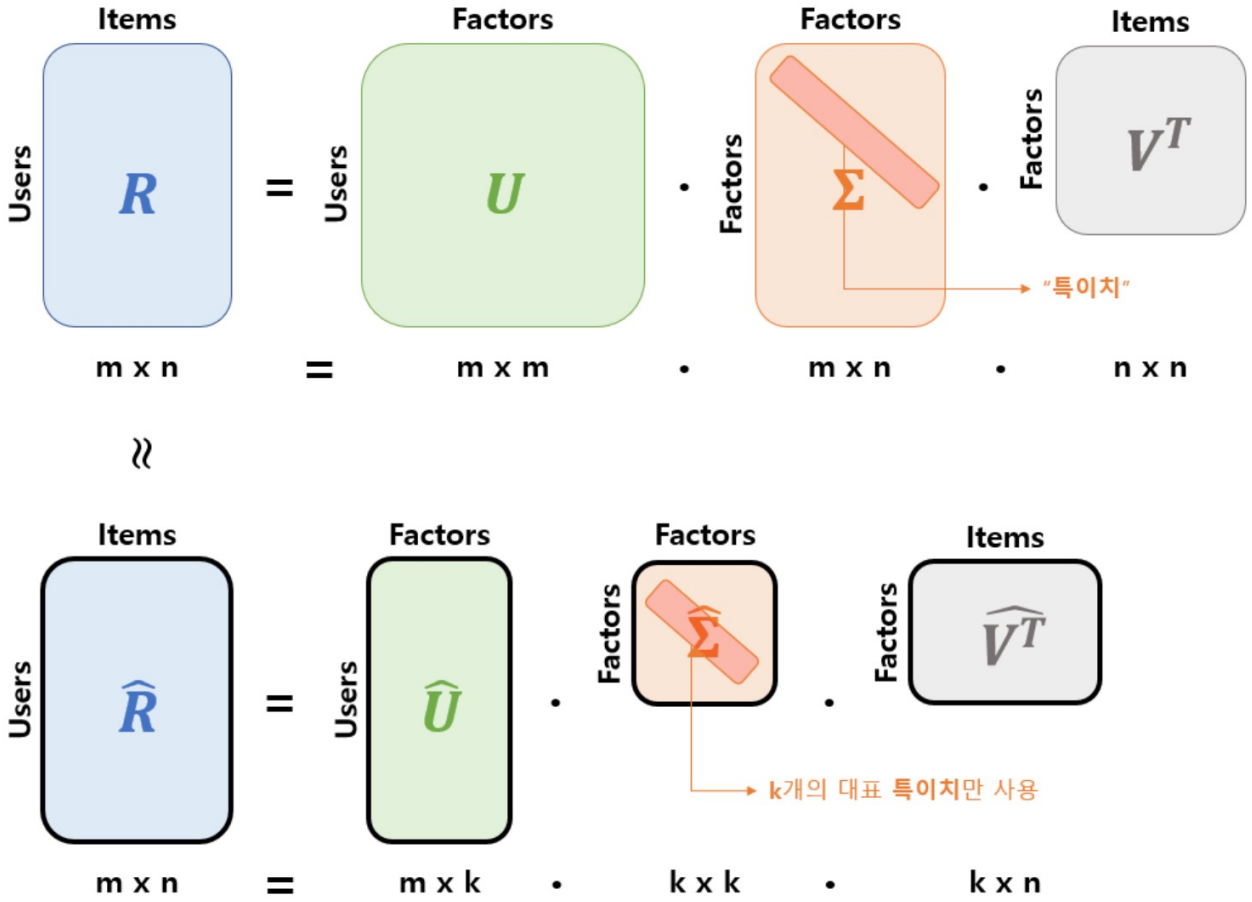
RR^T을 고유값 분해해서 얻은 직사각 대각 행렬로
대각 원소들은 R 의 singular value

참고

$UU^T = I, \quad V^TV = I \quad (U, V$는 직교행렬)

$RR^T = U(\Sigma\Sigma^T)U^T$

$R^TR = V(\Sigma^T\Sigma)V^T$

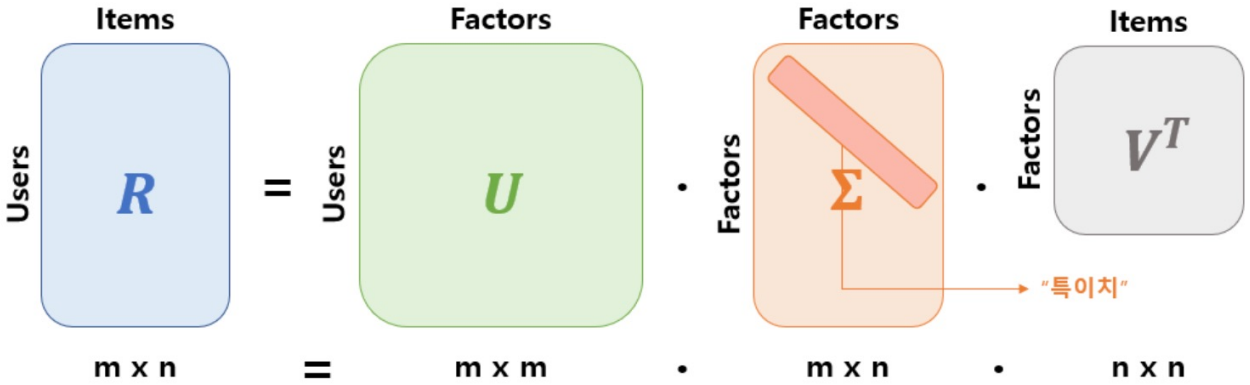


Singular Value Decomposition

Truncated SVD: $R = U\Sigma V^T \approx \hat{U}\Sigma_k\widehat{V^T} = \hat{R}$

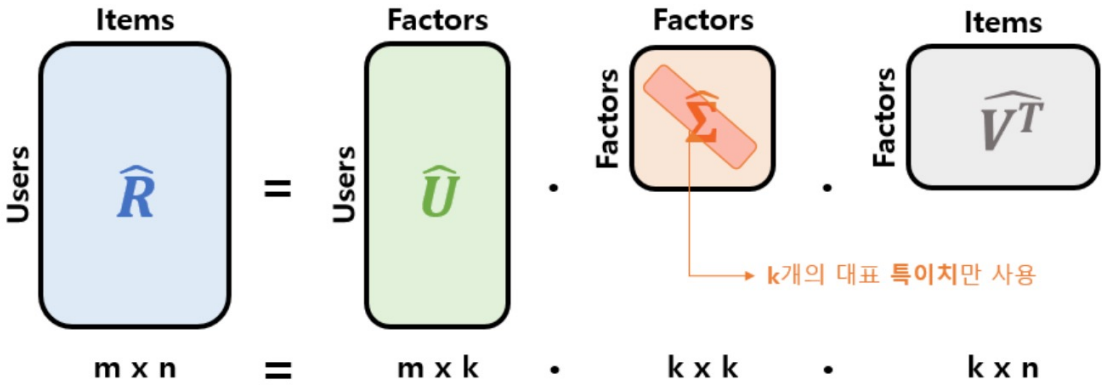
대표값으로 사용될 k개의 특이값만 사용함

k는 모델의 하이퍼 파라미터



평점 예측은 $\hat{R}$ 은 축소된 $\hat{U}, \widehat{V^T}, \Sigma_k$에 의해 계산됨

k개의 Latent Factor의 의미를 유추할 수 있지만
정확히 어떤 특성을 갖는 지 알 수 없음



전통적인 SVD의 문제점

분해(Decomposition)하려는 행렬의 Knowledge가 불완전할 때 정의되지 않음

Sparsity가 높은 데이터의 경우 결측치가 매우 많고, 실제 데이터는 대부분 Sparse Matrix임

따라서 결측된 entry를 모두 채우는 Imputation을 통해 Dense Matrix를 만들어 SVD를 수행함

Imputation은 데이터의 양을 상당히 증가시키므로, Computation 비용이 높아짐

ex) 결측된 entry를 0 or 유저/아이템의 평균 평점으로 채움

정확하지 않은 Imputation은 데이터를 왜곡시키고 예측 성능을 떨어뜨림

행렬의 entry가 매우 적을 때 SVD를 적용하면 과적합 되기 쉬움

➔ SVD의 원리를 차용하되, MF를 학습하기 위한 근사적인 방법이 필요함

SVD와 Matrix Factorization

실제 Matrix Factorization의 구현은 유저 매트릭스, 아이템 매트릭스 2개로 이루어짐

SVD로 분해된 행렬의 Σ를 U나 V^T에 곱해서 흡수시킨다면 MF와 같음

(N x K) x (K x K) x (K x M) = (N x K) x (K x M) = (N x M)

$$R \approx U\Sigma_k V^T$$

$$= (U\Sigma_k)V^T = U'V^T$$

$$= U(\Sigma_k V^T) = UV'^T$$

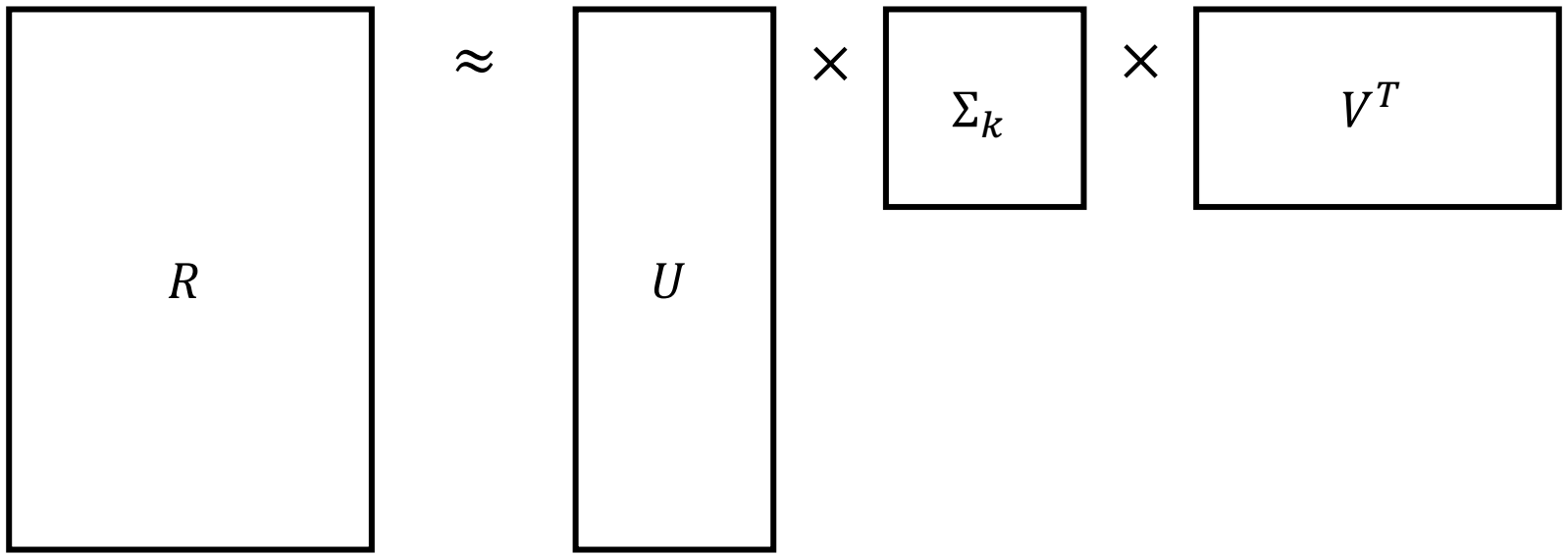
$$\hat{r}_{u,i} = \sum_k u_{ik}s_{kk}v_{kj} = \sum_k (u_{ik}s_{kk})v_{kj} = \sum_k u_{ik}'v_{kj} = u_i'^T v_j$$

$$\sum_k u_{ik}(s_{kk}v_{kj}) = \sum_k u_{ik}v_{kj}' = u_i^T v_j'$$

➔ SVD를 통해 분해된 행렬은 결국 유저 매트릭스와 아이템 매트릭스의 곱이 됨

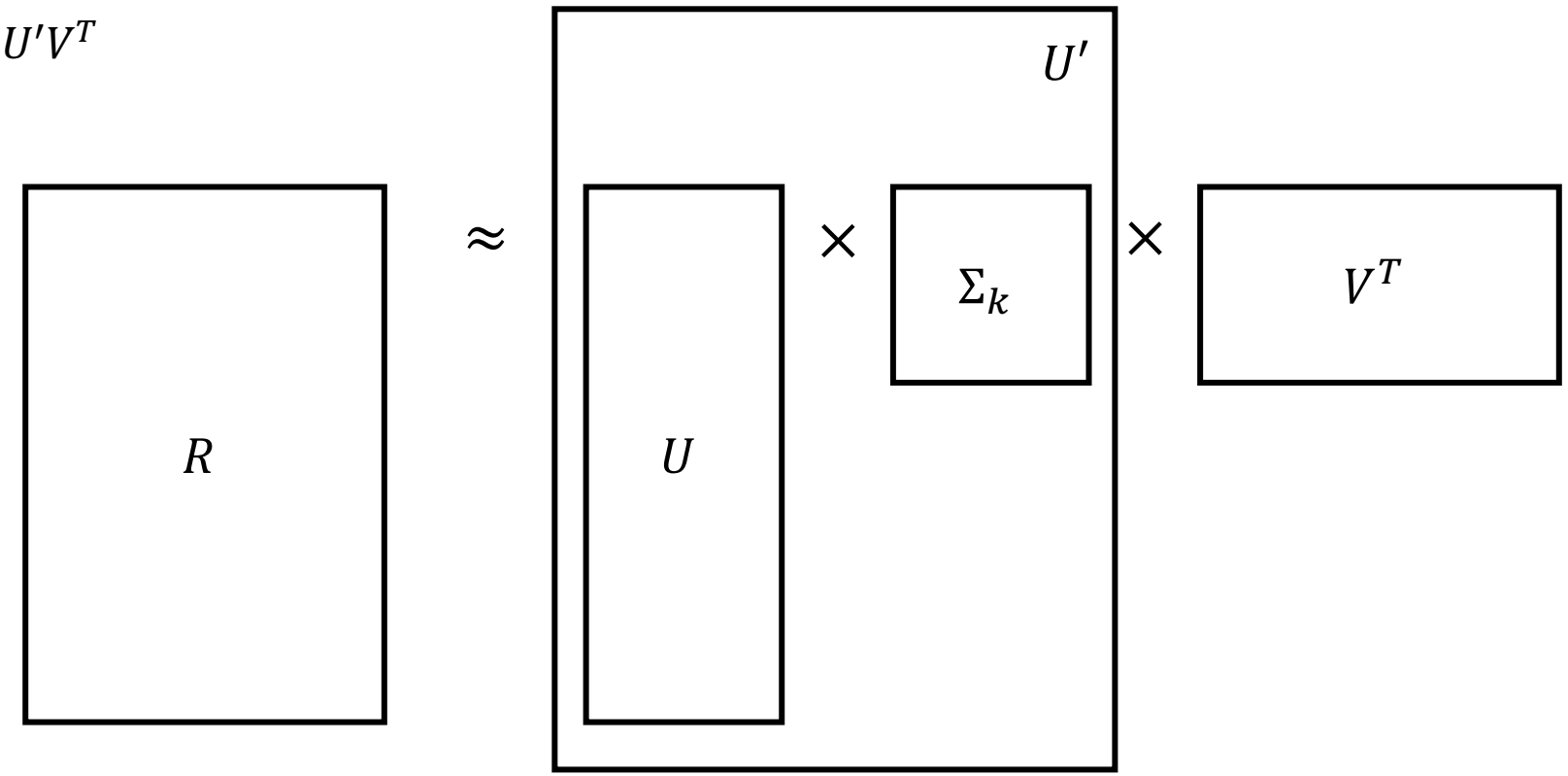
SVD와 Matrix Factorization

$$R \approx U\Sigma_k V^T$$
$$= (U\Sigma_k)V^T = U'V^T$$



SVD와 Matrix Factorization

$$R \approx U\Sigma_k V^T$$
$$= (U\Sigma_k)V^T = U'V^T$$



Matrix Factorization

Rating Matrix를 P와 Q로 분해

$R \approx P \times Q^T = \hat{R}$

$P \rightarrow |U| \times k$

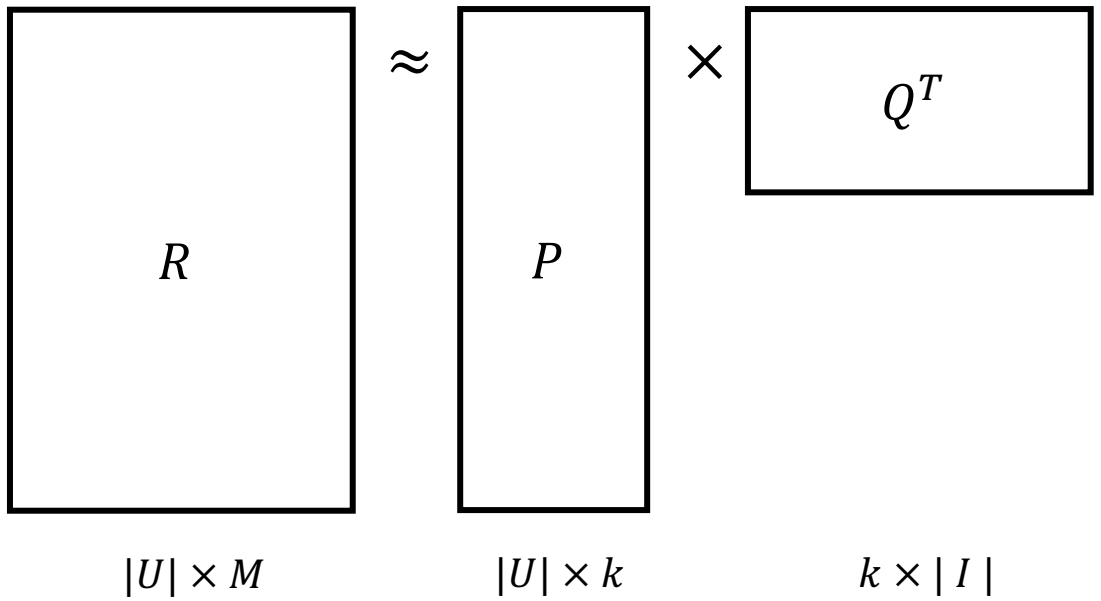
$Q \rightarrow |I| \times k$

평점 예측치: $\hat{r}_{u,i} = p_u^T q_i$

MF 학습

R 과 $\hat{R}$ 이 최대한 유사하도록 P, Q를 학습하는 과정이다

$$\min_{P,Q} \sum_{observed\ r_{u,i}} (r_{u,i} - p_u^T q_i)^2$$



SVD를 직접 구현해보자

3. Matrix Factorization과 최적화 기법

Matrix Factorization

$R \approx P \times Q^T = \hat{R}$

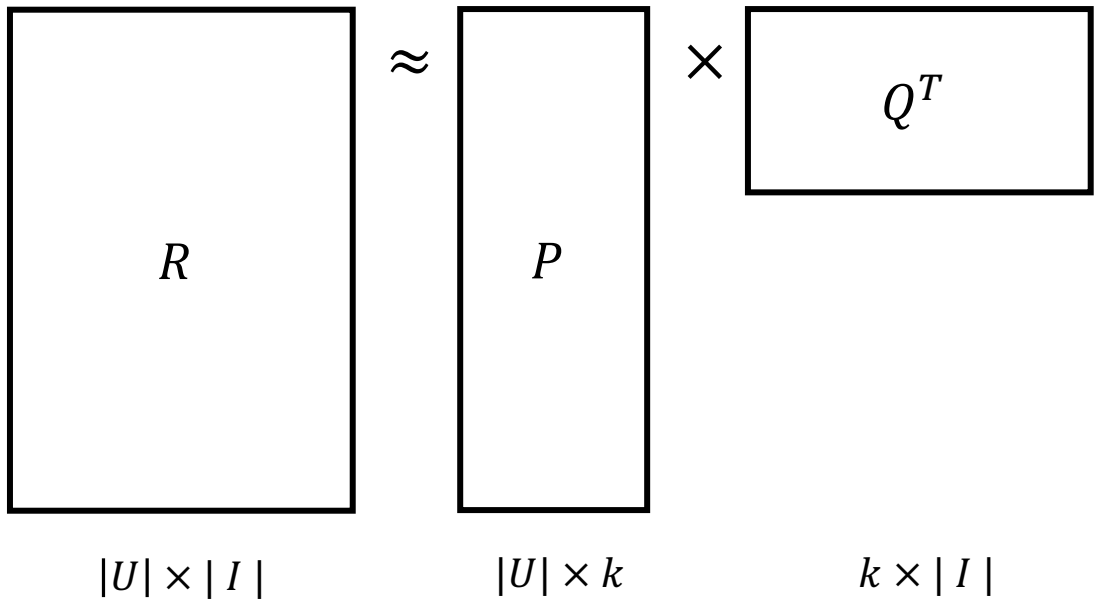
R 과 최대한 유사하게 $\hat{R}$ 을 추론하는 것이 목표

true rating: $r_{u,i}$

predicted rating: $\widehat{r_{u,i}} = p_u^T q_i$

Objective Function을 정의하고 이를 Minimize하는 Optimization을 수행함

$$\min_{P,Q} \sum_{observed\ r_{u,i}} (r_{u,i} - p_u^T q_i)^2$$



Objective Function

$$\min_{P,Q} \sum_{observed\ r_{u,i}} (r_{u,i} - p_u^T q_i)^2 + \lambda(||p_u||_2^2 + ||q_i||_2^2)$$

$r_{u,i}$: 학습 데이터에 있는 유저 u의 아이템 i에 대한 실제 rating

p_u: 유저 u의 latent vector, q_i: 아이템 i의 latent vector

이는 최적화 문제를 통해 업데이트되는 파라미터

실제 관측된 데이터만을 사용하여 모델을 학습

SVD는 행렬 분해를 위해 결측 entry를 채워넣었음

λ term은 L2 Regularization(정규화)을 의미

학습 데이터에 과적합되는 것을 방지함

잠깐, 여기서 Regularization(정규화)란 무엇인가?

학습 데이터에 지나치게 overfitting되는 경우 weight의 값이 커지게 됨

➔ weight의 크기를 loss function에 넣어주면 weight에 제한이 걸리게 됨

Regularization Term에 곱해지는 λ에 크기에 따라 영향도가 달라짐

λ가 너무 크면 weight가 제대로 변하지 않아서 underfitting이 일어남

대표적인 방법으로 L1, L2 정규화가 있음

$$L(w) = \sum_{i=1} \left(h_w\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_j |w_j|$$

$$L(w) = \sum_{i=1} \left(h_w\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_j w_j^2$$

Stochastic Gradient Descent in MF

실제 rating과 예측된 rating의 차이를 에러로 정의

Error

$$e_{ui} = r_{ui} - p_u^T q_i$$

Gradient

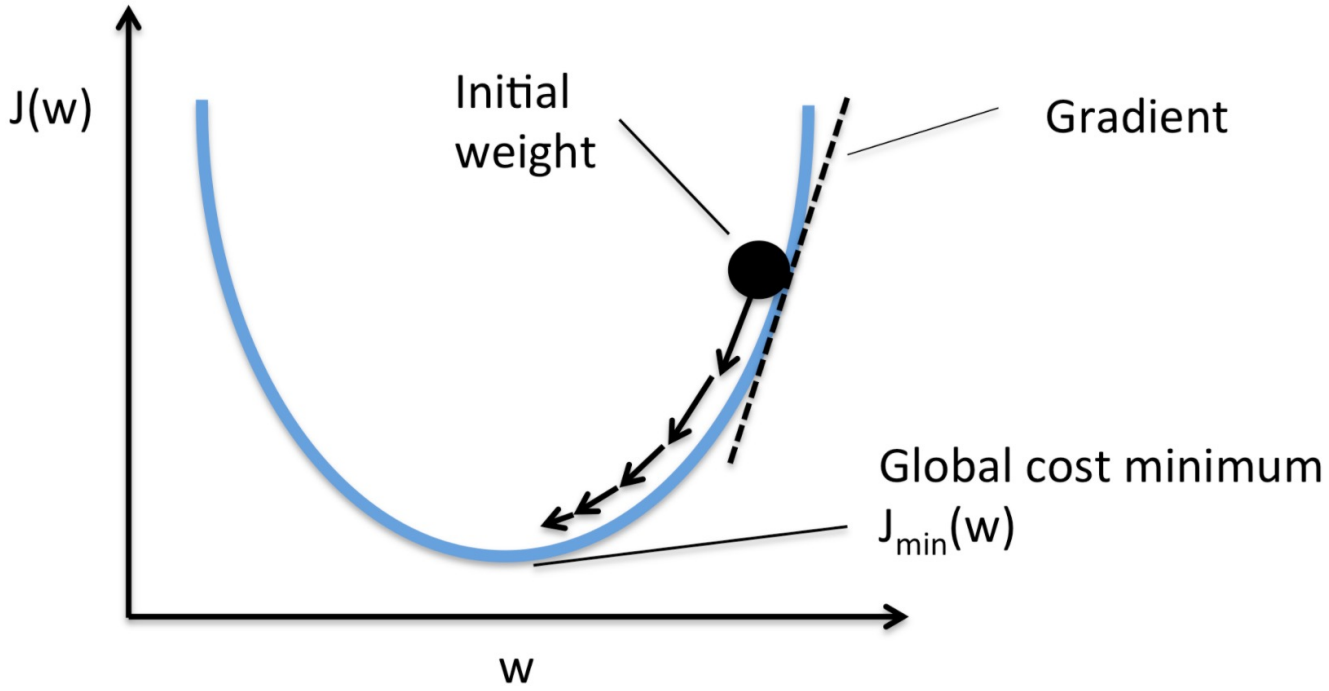
$$\frac{\partial L}{\partial p_u} = \frac{\partial (r_{ui} - p_u^T q_i)^2}{\partial p_u} + \frac{\partial \lambda ||p_u||_2^2}{\partial p_u} = -2(r_{ui} - p_u^T q_i) q_i + 2\lambda p_u = -2(e_{ui} q_i - \lambda p_u)$$

Gradient의 반대방향으로 p_u, q_i 를 업데이트

η는 learning rate

$$p_u \leftarrow p_u + \eta \cdot (e_{ui} q_i - \lambda p_u)$$
$$q_i \leftarrow q_i + \eta \cdot (e_{ui} p_u - \lambda q_i)$$

Gradient Descent 예시



$$J(\mathbf{w}) = \frac{1}{2}\sum_i (\text{target}^{(i)} - \text{output}^{(i)})^2$$

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j}$$

$$w_j := w + \Delta w_j$$

Paper Review

Matrix Factorization 기반 추천으로 가장 널리 알려진 논문

기본적인 MF에 다양한 테크닉을 추가하여 성능을 향상시킴

Adding Biases

Objective Function

$$\min_{P,Q} \sum_{observed\, r_{u,i}} (r_{u,i} - p_u^T q_i)^2 + \lambda(||p_u||_2^2 + ||q_i||_2^2)$$

Bias 추가된 Function

$$\min_{P,Q} \sum_{observed\, r_{u,i}} (r_{u,i} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda(\left|\left|p_u\right|\right|_2^2 + \left|\left|q_i\right|\right|_2^2 + b_u^2 + b_i^2)$$

어떤 유저는 모든 영화에 대해서 평점을 짜게 줄 수도 있음

아이템도 편향이 생길 수 있는 것은 마찬가지

➔ 전체 평균, 유저/아이템의 bias를 추가하여 예측 성능을 높임

Adding Biases

Error

$$e_{ui} = r_{ui} - \mu - b_u - b_i - p_u^T q_i$$

Gradient의 반대방향으로 b_u, b_i, x_u, y_i 를 업데이트

γ는 learning rate

$$b_u \leftarrow b_u + \gamma \cdot (e_{ui} - \lambda b_u)$$
$$b_i \leftarrow b_i + \gamma \cdot (e_{ui} - \lambda b_i)$$
$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} q_i - \lambda p_u)$$
$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} p_u - \lambda q_i)$$

Temporal Dynamics

시간에 따라 변화는 유저, 아이템의 특성을 반영하고 싶음

아이템이 시간이 지남에 따라 인기도가 떨어짐

유저가 시간이 흐르면서 평점을 내리는 기준이 엄격해짐

시간을 반영한 평점 예측

학습 파라미터가 시간을 반영하도록 모델을 설계함

$$\widehat{r_{ui}(t)} = \mu + b_u(t) + b_i(t) + p_u^T q_i(t)$$

ex) $b_u(t) = b_u + \alpha_u \cdot sign(t - t_u) \cdot |t - t_u|^\beta$

Inputs with Varying Confidence Level

Objective Function

$$\min_{P,Q} \sum_{observed\, r_{u,i}} (r_{u,i} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda \left(||p_u||_2^2 + ||q_i||_2^2 + b_u^2 + b_i^2 \right)$$

Objective Function with confidence

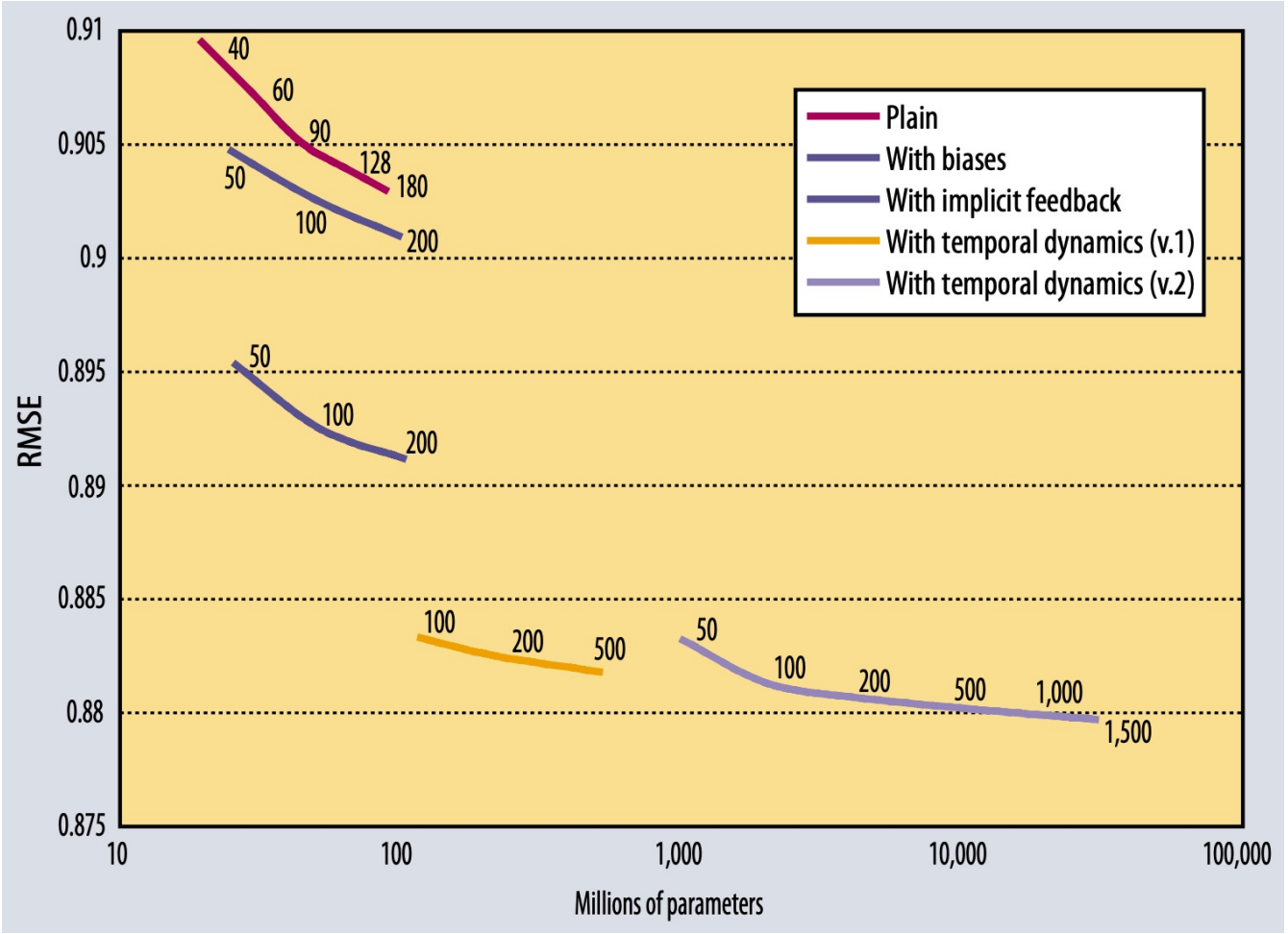
$$\min_{P,Q} \sum_{observed\, r_{u,i}} c_{u,i}(r_{u,i} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda(||p_u||_2^2 + ||q_i||_2^2 + b_u^2 + b_i^2)$$

$c_{u,i}$는 $r_{u,i}$에 대한 신뢰도를 의미함, 모든 평점이 동일한 신뢰도를 갖지 않음

대규모 광고 집행과 같이 특정 아이템이 많이 노출되어 클릭되는 경우

유저가 아이템에 대한 평점이 정확하지 않은 경우 (Implicit Feedback)

Model Performance



Paper Review

Implicit Feedback 데이터에 적합하도록 MF 모델을 설계하여 성능을 향상시킨 논문

Collaborative Filtering for Implicit Feedback Datasets

Yifan Hu
AT&T Labs – Research
Florham Park, NJ 07932

Yehuda Koren*
Yahoo! Research
Haifa 31905, Israel

Chris Volinsky
AT&T Labs – Research
Florham Park, NJ 07932

Abstract

A common task of recommender systems is to improve customer experience through personalized recommendations based on prior implicit feedback. *These systems passively track different sorts of user behavior, such as purchase history, watching habits and browsing activity, in order to model user preferences. Unlike the much more extensively researched* explicit feedback, *we do not have any*

tent based approach creates a profile for each user or product to characterize its nature. As an example, a movie profile could include attributes regarding its genre, the participating actors, its box office popularity, etc. User profiles might include demographic information or answers to a suitable questionnaire. The resulting profiles allow programs to associate users with matching products. However, content based strategies require gathering external information that might not be available or easy to collect.

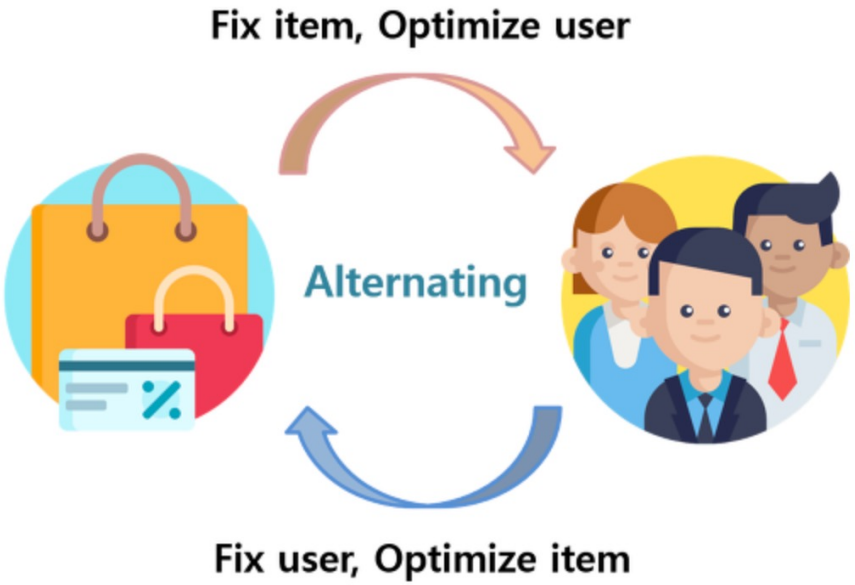
Alternative Least Square

Basic Concept

유저와 아이템 매트릭스를 번갈아가면서 업데이트함

두 매트릭스 중 하나를 상수로 놓고 나머지 매트릭스를 업데이트함

p_u, q_i 가운데 하나를 고정하고 least-square 문제를 푸는 것



Sparse한 데이터에 대해 SGD보다 Robust한 모습을 보임

SGD와 다르게 병렬처리가 가능

Spark와 같은 분산 처리 시스템에서 대용량 데이터에 대해 빠른 학습 가능

참고

p_u 나 q_i를 상수로 고정할 경우 objective(loss function)가 quadratic form이 되어 convex

Solving ALS by Matrix Multiplication

Objective Form

$$\min_{P,Q} \sum_{observed\, r_{u,i}} (r_{u,i} - p_u^T q_i)^2 + \lambda(||p_u||_2^2 + ||q_i||_2^2)$$

아래 수식을 사용해 P, Q를 번갈아 가면서 업데이트

$$p_u = (Q^T Q + \lambda I)^{-1} Q^T r_u$$
$$q_i = (P^T P + \lambda I)^{-1} P^T r_i$$

Using Preference / Confidence

Implicit Feedback을 처리하는 방법

Preference

유저 u가 아이템 i를 선호하는지 여부를 binary로 표현

$$f_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases}$$

Confidence

유저 u가 아이템 i를 얼마나 선호하는지 나타내는 increasing function

α는 positive feedback과 negative feedback 간의 상대적인 중요도를 조정하는 하이퍼 파라미터

$$c_{ui} = 1 + \alpha \cdot r_{ui}$$

참고

c_{ui}와 실제 평점 r_{ui}과 α에 의해 결정되는 constant이므로 학습 알고리즘의 로직이 바뀌지 않음

Using Preference / Confidence

기존 Objective Function

$$\min_{P,Q} \sum_{observed\, r_{u,i}} (r_{u,i} - p_u^T q_i)^2 + \lambda(||p_u||_2^2 + ||q_i||_2^2)$$

새로운 Objective Function

$$\min_{P,Q} \sum_{observed\, r_{u,i}} c_{u,i}(f_{u,i} - p_u^T q_i)^2 + \lambda(||p_u||_2^2 + ||q_i||_2^2)$$

참고

SGD나 ALS로 동일하게 풀 수 있음

논문에서는 ALS 매트릭스 연산을 통해 좀 더 효율적으로 연산하는 방법을 제안

Solving ALS by Matrix Multiplication

기본 Objective Form

$$p_u = (Q^T Q + \lambda I)^{-1} Q^T r_u$$
$$q_i = (P^T P + \lambda I)^{-1} P^T r_i$$

Confidence / Preference 분리

$$p_u = (Q^T C^u Q + \lambda I)^{-1} Q^T C^u f_u$$
$$q_i = \left(P^T C^i P + \lambda I\right)^{-1} P^T C^i f_i$$

SGD, ALS 기반의 MF 모델을 구현해보자

4. BPR Optimization with MF

Paper Review

Implicit Feedback 데이터를 활용해 MF를 학습할 수 있는 새로운 관점을 제시한 논문

베이지안 추론에 기반하고 있으며 Implicit data를 통해서도 서로 다른 아이템에 대한 유저의 선호도를 반영함

BPR: Bayesian Personalized Ranking from Implicit Feedback

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme
{srendle, freudenthaler, gantner, schmidt-thieme}@ismll.de
Machine Learning Lab, University of Hildesheim
Marienburger Platz 22, 31141 Hildesheim, Germany

Introduction

사용자의 클릭, 구매 등의 로그는 Implicit Feedback 데이터

평점과 같이 아이템에 대한 선호가 분명하게 드러나지 않음

Implicit Feedback 데이터의 경우 0/1 binary로 이루어져 있음

일반적으로 유저가 아이템을 클릭/구매 할 확률을 예측하는 문제

논문에서는 AUC를 모델 성능 지표로 비교함

Ranking을 고려한 최적화

유저가 item i보다 j를 좋아한다면 이 정보를 사용해 MF의 파라미터를 학습함

유저 u에 대해 item i > item j 라면 이는 유저 u의 Personalized Ranking

Personalized Ranking

사용자에게 순서(Ranking)가 있는 아이템 리스트를 제공하는 문제

equal to 아이템 추천 문제

Implicit Feedback 데이터만을 통해서 추론해야 함

Implicit Feedback 추천 시스템은 positive observation만 존재

관측되지 않은 데이터에 대해서 아래 두 가지를 모두 고려

유저가 아이템에 관심이 없는 것인가

유저가 실제로 관심이 있지만 아직 모르는 것인가

Personalized Ranking

Formalization

유저 U, 아이템 I 집합에 대해서,유저 별로 아이템의 선호도 순서($>_u \subset I^2$)를 정의해보자

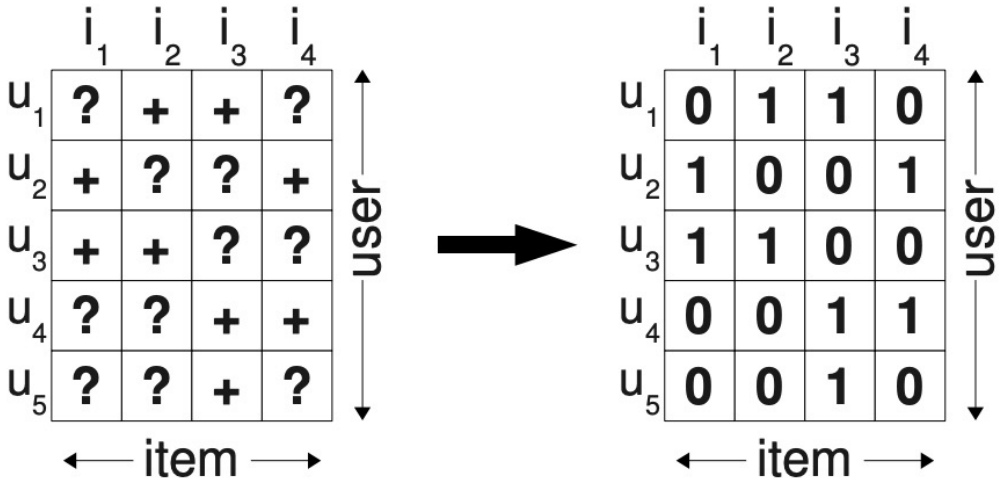
$$\forall i, j \in I : i \neq j \Rightarrow i >_u j \vee j >_u i \qquad (totality)$$
$$\forall i, j \in I : i >_u j \wedge j >_u i \Rightarrow i = j \qquad (antisymmetry)$$
$$\forall i, j, k \in I : i >_u j \wedge j >_u k \Rightarrow i >_u k \qquad (transitivity)$$

Personalized Ranking

Analysis of the problem setting

기존에는 관측되지 않은 entry를 0으로 채웠음

➔ 실제로 관심있지만 아직 관측되지 않은 데이터도 0으로 처리됨



Personalized Ranking

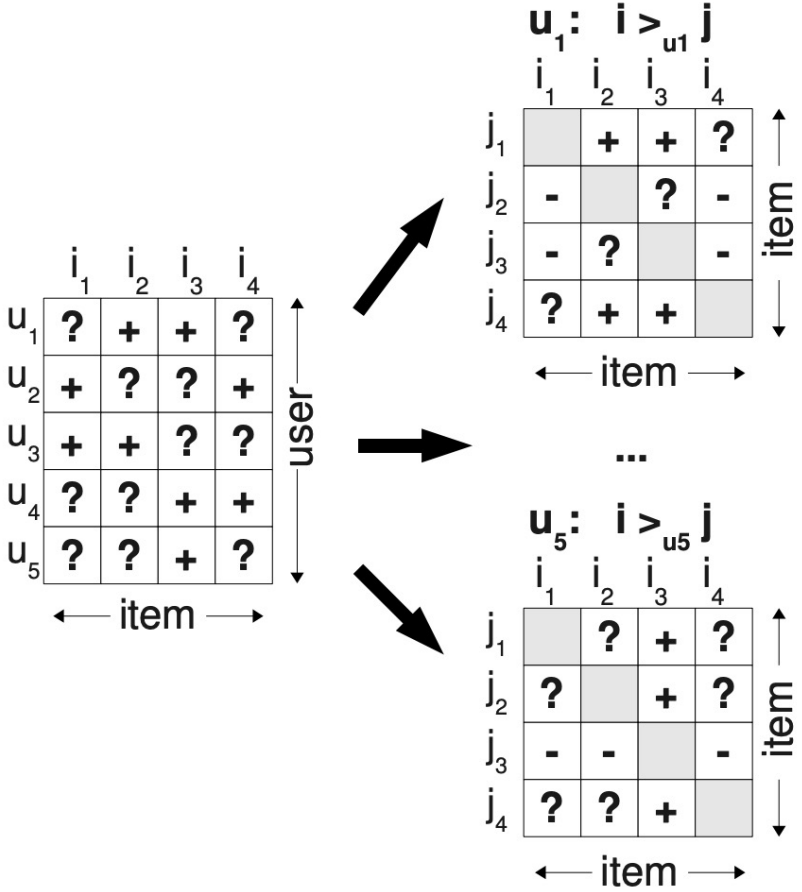
Analysis of the problem setting

가정

1. 관측된 item을 관측되지 않은 item보다 선호
2. 관측된 아이템끼리는 선호도를 추론할 수 없음
3. 관측되지 않은 아이템끼리도 선호도를 추론할 수 없음

특징

1. 관측되지 않은 item들에 대해서도 정보를 부여하여 학습함
2. 관측되지 않은 item들에 대해서도 ranking이 가능



Personalized Ranking

Analysis of the problem setting

학습 데이터 생성

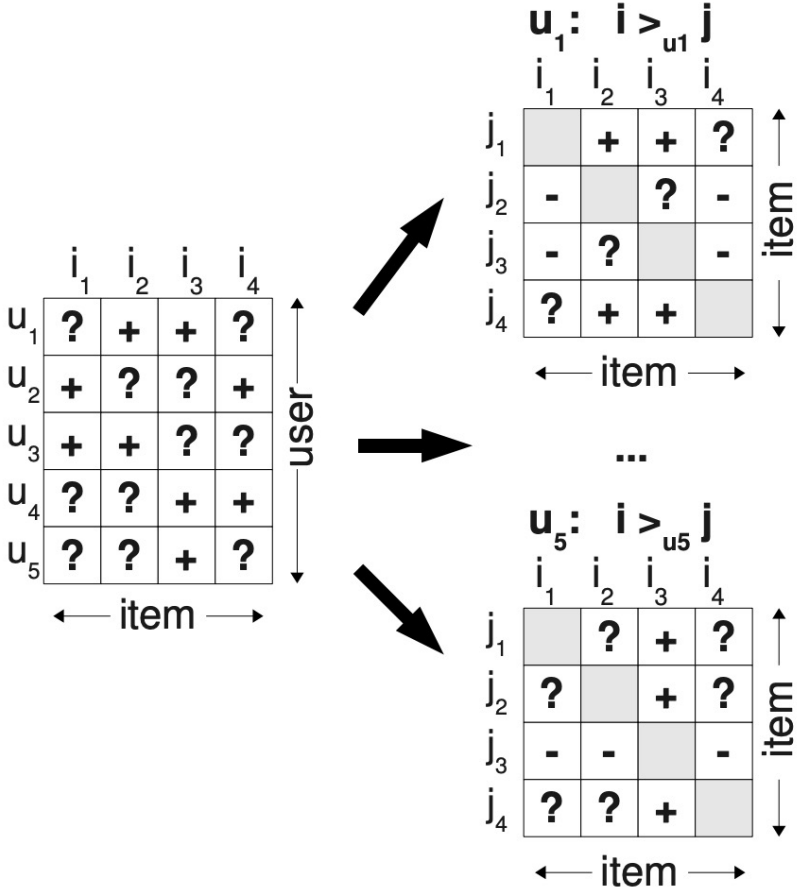
유저 u가 선호하는 아이템들을 I_u^+라고 하면,

$D_S := \{(u, i, j) \mid i \in I_u^+ \ \wedge \ j \in I \setminus I_u^+\}$

유저 u_1의 선호도 데이터 예시

$i_2 >_{u1} i_1, \qquad i_3 >_{u1} i_1$

$i_2 >_{u1} i_4, \qquad i_3 >_{u1} i_4$



BPR Optimization

최대 사후 확률 추정 (Maximum A Posterior)

Bayes 정리에 의해,

$$p(\Theta| >_u) \propto p(>_u |\Theta)\, p(\Theta)$$

사전 확률(prior): $p(\Theta)$, 파라미터에 대한 사전 정보

사후 확률(posterior): $p(\Theta| >_u)$, 주어진 유저의 선호 정보에 대한 파라미터에 대한 확률

가능도(likelihood): $p(>_u |\Theta)$, 주어진 파라미터에 대한 유저의 선호 정보에 대한 확률

➔ posterior를 최대화 한다는 것은 주어진 유저 선호 정보를 최대한 잘 나타내는 파라미터를 추정하는 것!

BPR Optimization

Likelihood: $p(>_u | \Theta)$

유저의 선호 정보 $>_u := (u, i, j) \in D_s$로 표현하면, 유저 선호 정보($>_u$)에 대한 likelihood

$$\prod_{u \in U} p(>_u | \Theta) = \prod_{(u,i,j) \in D_s} p(i >_u j) = \prod_{(u,i,j) \in D_s} \sigma\left(\widehat{x_{uij}}(\Theta)\right)$$

사용자 u가 아이템 i 를 j 보다 좋아할 확률

$$p(i >_u j) := \sigma\left(\widehat{x_{uij}}(\Theta)\right), \qquad \sigma(x) := \frac{1}{1 + e^{-x}} \text{ (sigmoid function)}$$

참고. 유저 u의 벡터를 p_u 아이템 i의 벡터를 q_i 라고 하면

$$\hat{r}_{ui} = p_u^T q_i, \qquad \hat{r}_{uj} = p_u^T q_j, \qquad \widehat{x_{uij}} = \hat{r}_{ui} - \hat{r}_{uj} = p_u^T q_i - p_u^T q_j$$

BPR Optimization

Prior: $p(\Theta)$

파라미터에 대한 사전 확률은 정규분포를 따른다고 가정

평균이 모두 0이고 공분산 행렬이 Σ_Θ인 정규분포

공분산 행렬 Σ_Θ는 $\lambda_\Theta I$로 설정하여 하이퍼파라미터의 개수를 조정함

$$p(\Theta) \sim N(0, \Sigma_\Theta) = N(0, \lambda_\Theta I)$$

BPR Optimization

BPR-OPT

$$
\begin{aligned}
\text{BPR-OPT} &:= \ln p(\Theta| >_u) \\
&= \ln p(>_u |\Theta)\, p(\Theta) \\
&= \ln \prod_{(u,i,j) \in D_S} \sigma(\hat{x}_{uij})\, p(\Theta) \\
&= \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta) \\
&= \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2
\end{aligned}
$$

where λ_Θ are model specific regularization parameters.

BPR Optimization

Gradient 계산

$$\frac{\partial \text{BPR-OPT}}{\partial \Theta} = \sum_{(u,i,j) \in D_S} \frac{\partial}{\partial \Theta} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \frac{\partial}{\partial \Theta} ||\Theta||^2$$

$$\propto \sum_{(u,i,j) \in D_S} \frac{-e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} - \lambda_\Theta \Theta$$

$$\Theta \leftarrow \Theta - \alpha \frac{\partial \text{BPR-OPT}}{\partial \Theta}$$

Objective인 BRP-OPT는 미분가능하기 때문에 Gradient로 Optimization을 수행함

그러나 일반적인 Gradient Descent 가 적절한 학습 방법이 아님

BPR Optimization

LEARNBPR

1: **procedure** LEARNBPR(D_S, Θ)
2: initialize Θ
3: **repeat**
4: draw (u, i, j) from D_S
5: $\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \cdot \Theta \right)$
6: **until** convergence
7: **return** $\hat{\Theta}$
8: **end procedure**

Bootstrap 기반의 SGD를 사용

D_s에서 triples를 랜덤 샘플링한 뒤 이를 사용하여 파라미터를 업데이트

($D_S := \{(u,i,j) \mid i \in I_u^+ \ \wedge \ j \in I \setminus I_u^+\}$)

일반적인 GD, SGD를 사용하면,

보통 i 보다 j 가 훨씬 많기 때문에 학습의 비대칭 발생

동일한 u, i 에 대해 계속 업데이트가 되므로 수렴이 잘 안되고 성능이 떨어짐

But! triples 단위로 랜덤 샘플링하면,

i 와 j 의 비대칭 학습 해소

동일한 u, i 가 계속 등장하지 않기 때문에 성능 우수

BPR Optimization

Matrix Factorization 모델에 적용

1: **procedure** LEARNBPR(D_S, Θ)
2: initialize Θ
3: **repeat**
4: draw (u, i, j) from D_S
5: $\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \cdot \Theta \right)$
6: **until** convergence
7: **return** $\hat{\Theta}$
8: **end procedure**

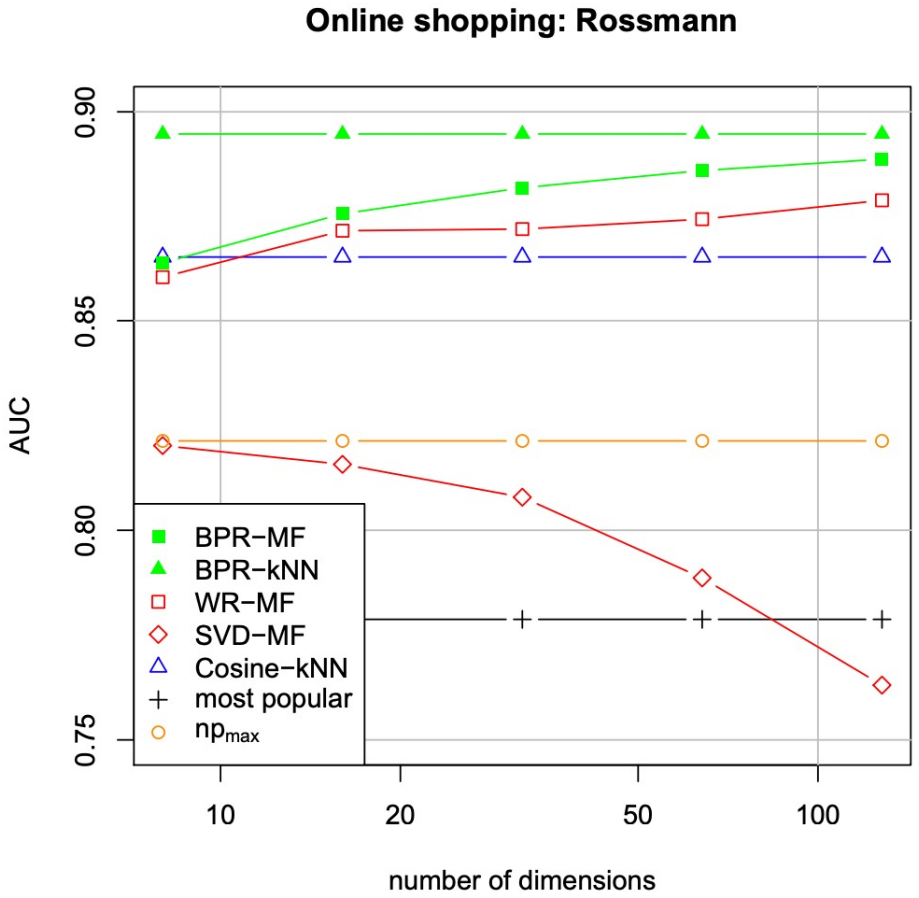
유저 u의 벡터를 p_u 아이템 i의 벡터를 q_i 라고 하면,

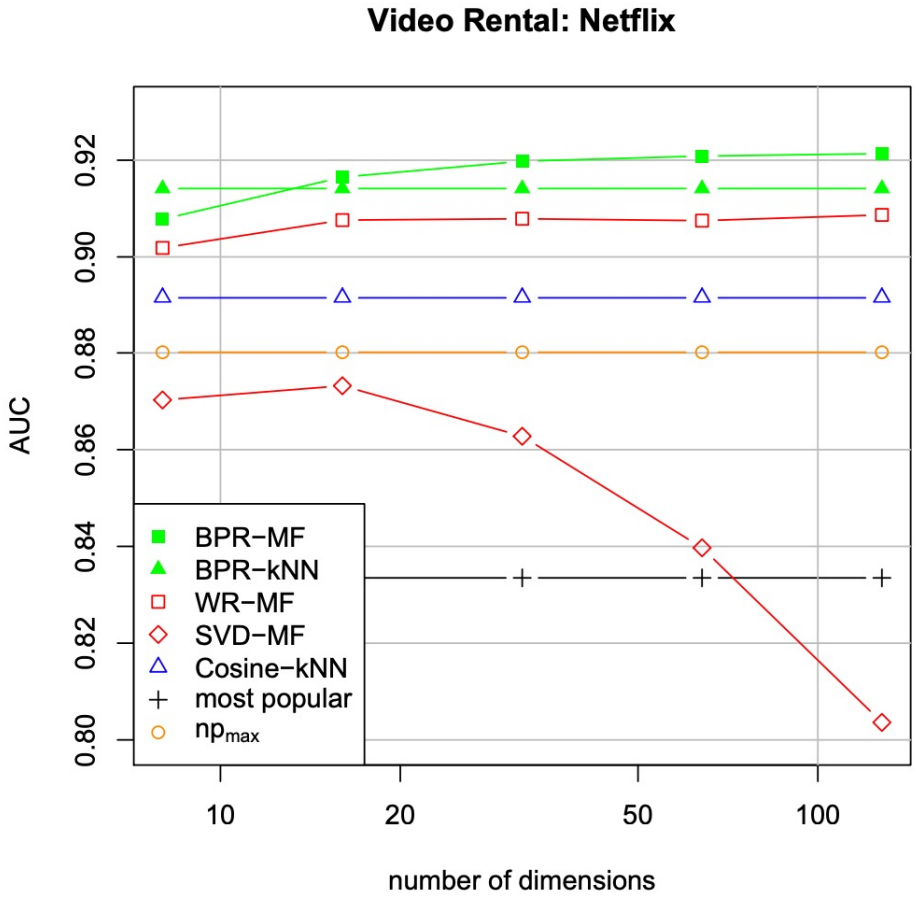
$$\hat{r}_{ui} = p_u^T q_i, \qquad \hat{r}_{uj} = p_u^T q_j,$$

$$\hat{x}_{u,i,j} = \hat{r}_{ui} - \hat{r}_{uj} = p_u^T q_i - p_u^T q_j$$

$$\frac{\partial \hat{x}_{u,i,j}}{\partial \theta} = \begin{cases} (q_{ik} - q_{jk}) & if\ \theta = p_{uk} \\ p_{uk} & if\ \theta = q_{ik} \\ -p_{uk} & if\ \theta = q_{jk} \end{cases}$$

BPR Performance





BPR 요약

Implicit Feedback 데이터만을 활용해 아이템 간의 선호도 도출

Maximum A Posterior 방법을 통해 파라미터를 최적화

LEARNBPR이라는 Bootstrap 기반의 SGD를 활용해 파라미터를 업데이트

Matrix Factorization에 BPR Optimization을 적용한 결과 성능 우수

5. Annoy를 활용한 서빙

MF 서빙 이슈

특정 유저 u에게 추천을 제공하기 위해서 해당 유저 벡터와 후보 아이템 벡터들의 연산이 필요함

$\hat{r}_{ui} = p_u^T q_i \quad for\ i \in I$, 추천 스코어는 두 벡터의 내적 (inner product)

후보 아이템들에 대해서 rating 값을 구하고 내림차순 정렬하여 상위 아이템을 추천

특정 아이템 i와 비슷한 아이템(i')을 추천하기 위해서 해당 아이템 벡터와 후보 아이템 벡터의 유사도 연산이 필요함

특정 아이템(i)에 대해서 유사도 $\cos(q_i, q_i')$를 구하여 내림차순 정렬하고 상위 아이템을 추천

만약 아이템의 개수가 엄청 많다면? (1M 이상)

모든 아이템에 대해서 스코어를 계산하게 될 경우 오랜 시간과 높은 연산량이 필요함

실시간 서빙에 있어서 특히 더 취약함

➔ 많은 후보 아이템 벡터 가운데 스코어가 가장 큰 n개의 아이템을 유사하게 추출할 수 없을까?

Annoy

수많은 데이터가 있는 n차원 벡터 공간에서 주어진 벡터와 가장 유사한 벡터들을 찾는 알고리즘

spotify에서 추천 시스템의 nearest-neighbor 문제를 해결하기 위해 개발

작동 방식

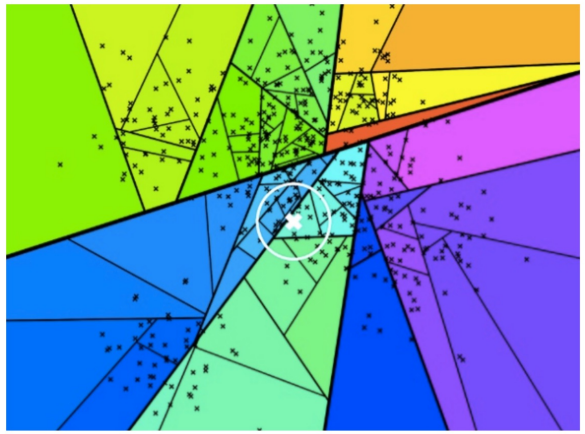
벡터 공간은 여러개의 권역으로 나누어 binary tree의 형태로 구성함

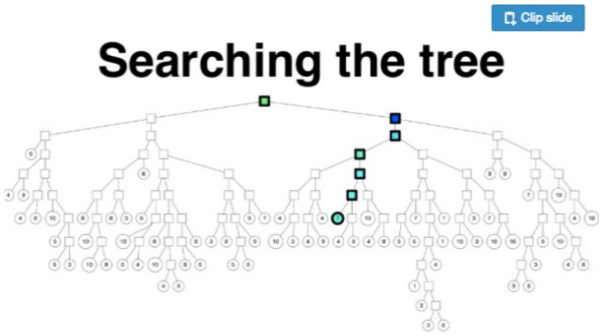
주어진 벡터가 속한 공간을 tree search로 찾고, 그 권역 안에서만 nearest neighbor 연산을 한다

제공 유사도 기준

angular distance = $\sqrt{2(1-\cos(i,i'))}$

dot product





Annoy 사용 예시

유저에게 아이템 추천

주어진 유저 벡터에 대해서 내적이 최대가 되는 아이템 벡터 N개를 찾아서 추천

이때 사용하는 유사도는 dot product

아이템과 비슷한 다른 아이템 추천

주어진 아이템 벡터와 가장 비슷한 아이템 벡터 N개를 찾아서 추천

이 때 사용하는 유사도는 angular distance

Implicit, Annoy 라이브러리 실습