✅

BPR

Domain	RecSys
tag	Optimization criterion Pair level ranking
Conference / Journal	UAI
정리 날짜	@2024년 1월 8일
AI summary	BPR-OPT은 Pair level optimization criterion으로, implicit feedback을 사용하여 학습하는 방식입니다. BPR-OPT를 사용하여 학습하는 LEARNBPR은 bootstrap sampling을 활용한 stochastic gradient descent 방식을 사용합니다. BPR-OPT는 Bayes theorem을 활용하여 만들어졌으며, MF와 KNN과 유사하지만 ranking에 최적화된 criterion을 사용합니다.
AI key info	Optimization criterion, Pair level ranking, Implicit feedback, BPR-OPT, LEARNBPR, Pairwise preference, Personalized ranking, BPR-OPT, BPR Optimization Criterion, Similarities to AUC, Learn-BPR, BPR, Adaptive k-nearest-neighbor, Relations to other methods, WR-MF, MMMF, Evaluation, Conclusion

Summary

Pair level ranking optimization criterion인 BPR-OPT 제공

- Implicit feedback을 사용하여 학습할 수 있는 방식
- Bayes theorem을 활용하여 만들었음

BPR-OPT를 사용하여 학습하는 방식인 LEARNBPR 제공

- bootstrap sampling을 활용한 stochastic gradient descent 방식 사용

Introduction

기존의 아이템 추천 methods는 아이템의 랭킹을 직접적으로 구하지 않았음

- Ranking을 구하기 위해 model parameter를 직접 조정하지 않음
- 대신 개별 아이템이 선택될지를 예측하는 것을 최적화함

랭킹을 직접 구하기 위한 방법을 제시

- BPR-OPT: personalized ranking criterion
- LearnBPR: BPR-OPT 사용한 model optimizing learning alogrithm

많은 최근 연구는 explicit feedback에 기반함

- 실제로는, implicit feedback의 접근성이 더 높음
- 유저가 직접 제공하는 explicit feedback은 수가 적지만, implicit은 무수히 생성됨

Methodology

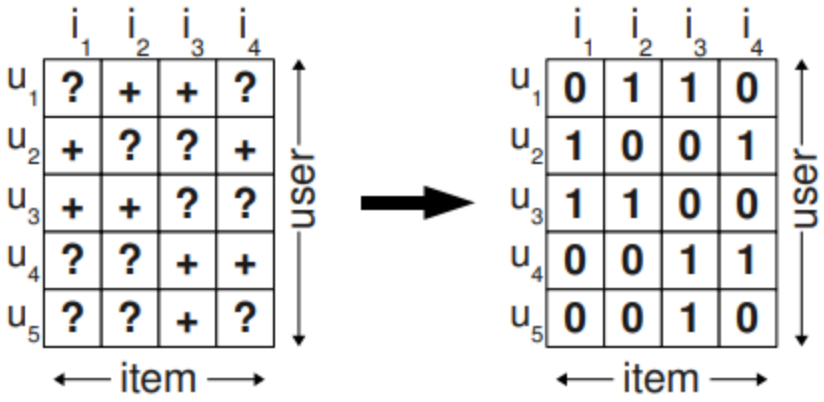
Pairwise preference

두 아이템 간에는 선호도 $>_u \subset I^2$ 가 있을 수 있음

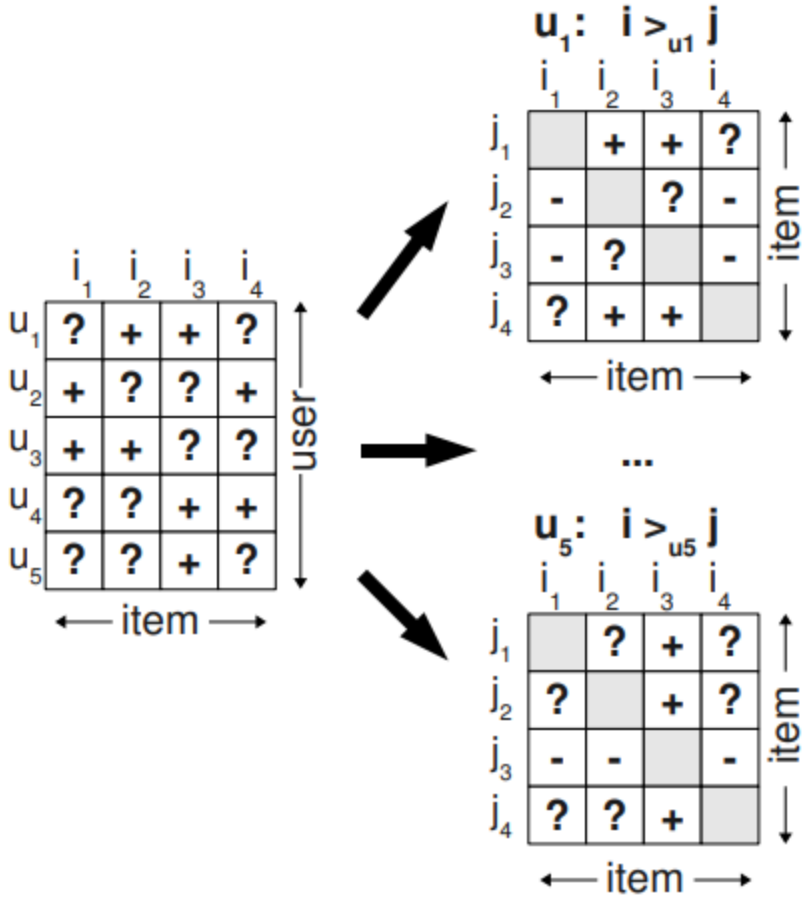
- 이 $>_u \subset I^2$에는 다음과 같은 성질이 적용됨
  - $\forall i, j \in I : i \neq j \Rightarrow i >_u j \vee j >_u i$ (totality)
  - $\forall i, j \in I : i >_u j \wedge j >_u i \Rightarrow i = j$ (antisymmetry)
  - $\forall i, j, k \in I : i >_u j \wedge j >_u k \Rightarrow i >_u k$ (transitivity)
- 편의를 위한 추가 정의
  - $I_u^+ := \{i \in I : (u, i) \in S\}$
  - $U_i^+ := \{u \in U : (u, i) \in S\}$

Personalized ranking

Implicit data의 문제점



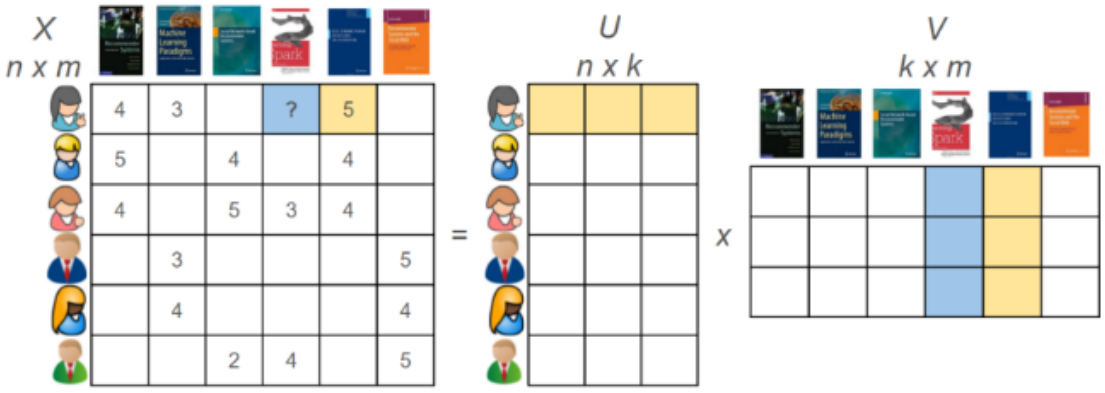
- Positive observation만 가능
  - Non-observed(Negative)는 Real Negative feedback과 missing values의 혼합
- 방법 1) Missing value 전부 무시
  - Positive와 Negative를 구분할 수 없어 ML 모델이 전혀 학습할 수 없음
- 방법 2) 전부 Negative 취급
  - Item별 Personalized score $\hat{x}_{ui}$를 예측하기
    - score 기준으로 정렬해 상위 Item 추천
  - $(u, i) \in S$ 인 Item을 1로, 이 외를 0으로 설정
  - 미래에 예측할 상호작용하지 않은 Item들을 모두 negative로 학습하는 문제 발생
    - 이런 모델들이 작동하는 이유는 regularization같은 overfitting 방지 때문임
- 해결: 개별 Item에 ranking 매기지 않고, pair로 만듬
  - Non-observed를 negative화 하지 않고, observed와 non-observed 사이의 선호도를 사용
    - Observed : $(u, i) \in S$ is preferable $>_u$ than $(U \times I) \backslash S$
    - $D_S := \{(u, i, j) \mid i \in I_u^+ \wedge j \in I \backslash I_u^+\}$
    - + 와 + 사이, ?와 ? 사이에는 선호도 없음



BPR-OPT

BPR Optimization Criterion

- 정확한 ranking을 매기는 것 == $p(\Theta| >_u)$를 높이는 것
  - user의 선호도를 잘 나타낼 수 있는 파라미터 Θ를 찾는 것
    - 주어진 user에 대해 학습 데이터를 가장 잘 만족(≒1)하는 Θ를 찾는 것
    - MF의 경우, 유저와 아이템으로 이루어진 matrix를 user-feature, feature-item으로 분해함
    - 이 feature factor를 잘 찾는 것 (아래 그림에서 k)



- $p(\Theta|>_u)$의 변환
  - $p(\Theta|>_u) \propto p(>_u|\Theta)p(\Theta)$ Bayes theorem
  - $p(>_u|\Theta)$ = $\prod_{u \in U} p(>_u|\Theta)$ user independent

    = $\prod_{(u,i,j) \in U \times I} p(i >_u j|\Theta)^{\delta((u,i,j) \in D_s)} \cdot (1 - p(i >_u j|\Theta))^{\delta((u,i,j) \notin D_s)}$ item independent

    = $\prod_{(u,i,j) \in D_s} p(i >_u j|\Theta)$ totality, antisymmetry

    D_s = training data

    $$\delta(b) := \begin{cases} 1 & \text{if } b \text{ is true,} \\ 0 & \text{else} \end{cases}$$

  - 실제 값 함수
    - $p(i >_u j|\Theta) := \sigma(\hat{x}_{uij}(\Theta))$
    - σ는 $\sigma(x) := \frac{1}{1+e^{-x}}$ 인 logistic sigmoid 함수
    - $\hat{x}_{uij}(\Theta)$: user, item사이 관계인 Θ의 실제 값 함수
    - 이후에는 $\hat{x}_{uij}$로 간략화
- prior probability $p(\Theta) \sim N(0, \Sigma_\Theta)$
  - $p(\Theta)$가 평균 0이고, 분산 Σ_Θ인 정규분포 따른다 가정
  - Σ_Θ : variance-covariance matrix, = $\lambda_\Theta I$ 라고 간소화

$$
\begin{aligned}
p(\Theta) &= \frac{1}{\sqrt{(2\pi)^d |\Sigma_\Theta|}} \exp\left(-\frac{1}{2}(\Theta-\mu)^T \Sigma_\Theta^{-1}(\Theta-\mu)\right) \\
&= (2\pi\lambda_\Theta)^{-\frac{d}{2}} \exp\left(-\frac{1}{2\lambda_\Theta}\Theta^T\Theta\right) \\
&= (2\pi\lambda_\Theta)^{-\frac{d}{2}} \exp\left(-\frac{1}{2\lambda_\Theta}\|\Theta\|^2\right)
\end{aligned}
$$

- BPR-OPT

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

수식 중간에 생략된 상수들 있음, λ_Θ 또한 hyperparameter이므로, $\frac{1}{2\lambda_\Theta}$를 대충 λ_Θ로 바꿈

λ_Θ는 model regularization parameter

Similarities to AUC (Justification)

AUC per user : True positive / 전체 item pair

$$
\text{AUC}(u) := \frac{1}{|I_u^+|\,|I \setminus I_u^+|} \sum_{i\in I_u^+} \sum_{j\in|I\setminus I_u^+|} \delta(\hat{x}_{uij} > 0)
$$

avg AUC

$$\text{AUC} := \frac{1}{|U|} \sum_{u \in U} AUC(u)$$

따라서, avg AUC는 다음과 같이 정리됨 (논문에서 AUC(u)로 표기됐는데 잘못 된 듯?)

$$\text{AUC}(u) = \sum_{(u,i,j) \in D_S} z_u \, \delta(\hat{x}_{uij} > 0) \qquad z_u = \frac{1}{|U| \, |I_u^+| \, |I \setminus I_u^+|}$$

$$\delta(x > 0) = H(x) := \begin{cases} 1, & x > 0 \\ 0, & \text{else} \end{cases}$$

BPR-OPT와 유사성

- normalizing constant z_u와 loss function만 다름
  - AUC: 미분 불가능한 Heaviside function $\delta(x > 0)$ 사용
  - BPR-OPT: 미분 가능한 $\ln \sigma(x)$사용

Learn-BPR

BPR-OPT의 gradient

$$\begin{aligned} \frac{\partial \text{BPR-Opt}}{\partial \Theta} &= \sum_{(u,i,j) \in D_S} \frac{\partial}{\partial \Theta} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \frac{\partial}{\partial \Theta} ||\Theta||^2 \\ &\propto \sum_{(u,i,j) \in D_S} \frac{-e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} - \lambda_\Theta \Theta \end{aligned}$$

Gradient descent의 문제

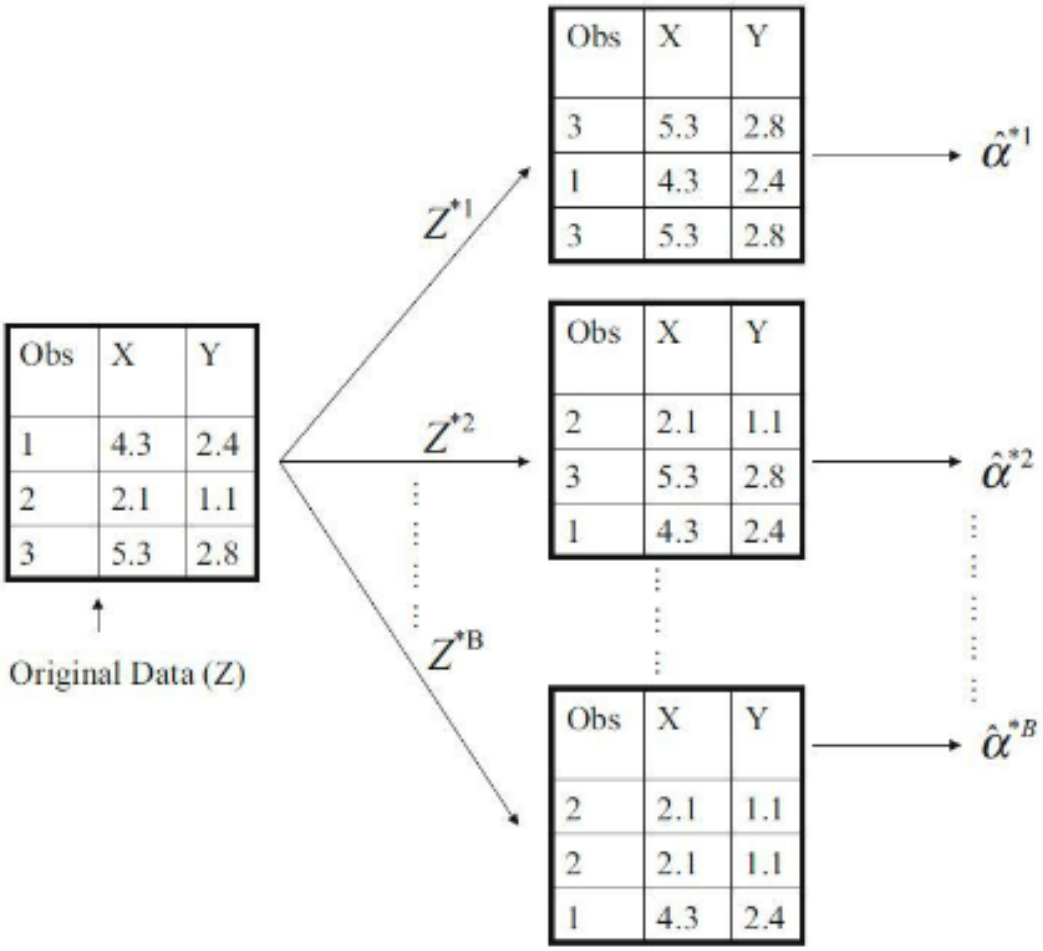
- 가장 기본적인 learning 방식이지만 한계가 있음
- Full gradient descent

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

  - $O(|S||I|)$의 모든 training triple(u,i,j)를 사용해야 해 convergence가 너무 오래걸림
  - skewness(왜곡)문제: 왜곡된 training pair가 poor convergence로 이끌 수 있음
    - 각 i가 모든 j와 비교되므로, 매우 큰 영향을 미침 ⇒ 아주 작은 α를 설정해야 함
- SGD(Stochastic Gradient Descent)

$$\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \Theta \right)$$

  - 전체 dataset batch아닌 일부 mini-batch씩 학습 진행
  - skew 문제에 좋은 해결책이지만, training 순서에 따라 poor convergence 되기도
    - item-wise, user-wise는 동일한 item이나 user에 대해 연속적인 학습 진행
- Bootstrap sampling 적용한 SGD
  - Bootstrap sampling(복원 랜덤추출)



- 이 논문에서는 하나의 triple마다 업데이트 실행
- 데이터가 매우 크므로, 일부분만 사용해도 수렴할 수 있음 → Earlystopping 가능
- 실험적으로 일찍 수렴함 보임

BPR 적용하기 (minor)

Decomposing estimator $\hat{x}_{uij}$

- MF랑 KNN은 개별 item에 대한 선호도 예측
- predictor $\hat{x}_{ul}$이 single number 가 아닌 prediction간의 차이에서 기인

$$\hat{x}_{uij} := \hat{x}_{ui} - \hat{x}_{uj}$$

- 같은 모델을 사용하더라도 ranking에 최적화된 criterion을 사용하였기 때문에 더 나음

Matrix factorizaion

- MF에서 $\hat{x}_{ui}$를 추정하는 건 행렬 $X : U \times I$ 를 추정하는 것임

$$\hat{X} := WH^t$$

  - X는 $W : |U| \times k$와 $H : |I| \times k$로 근사됨
    - k는 dimensionality(feature의 수)의 근사값
    - W의 각 행 w_u는 유저의 feature vector, H의 row vector도 아이템의 feature vector

$$\hat{x}_{ui} = \langle w_u, h_i \rangle = \sum_{f=1}^{k} w_{uf} \cdot h_{if}$$

  - MF의 model parameter는 $\Theta = (W, H)$로, 유저, 아이템의 non-observed property를 모델링하는 latent variable(잠재 변수)로 볼 수 있음
- 일반적으로 MF의 X에 대한 best approximation $\hat{X}$은 SVD로 구해짐
  - but overfitting issue: regularized least square opt, non-negatvie factorization 등 씀
- Ranking에서는 BPR-OPT가 가장 좋음
  - 모델 파라미터 Θ에 대한 $\hat{x}_{uij}$의 gradient만 구하면 됨
  - MF에서 이 gradient는 다음과 같음

$$\frac{\partial}{\partial\theta}\hat{x}_{uij} = \begin{cases} (h_{if} - h_{jf}) & \text{if } \theta = w_{uf}, \\ w_{uf} & \text{if } \theta = h_{if}, \\ -w_{uf} & \text{if } \theta = h_{jf}, \\ 0 & \text{else} \end{cases}$$

- Regularization 상수로 세 개 사용
  - λ_W: user feature인 W에 사용
  - λ_{H+}: item feature의 positive update에 사용
  - λ_{H-}: item feature의 negative update에 사용

Adaptive k-nearest-neighbor(?)

- 유사하게 작동하지만, item기반이 user기반 knn보다 잘 작동함
  - item의 예측이 과거 관찰한 item I_u^+에 기반함

$$\hat{x}_{ui} = \sum_{l \in I_u^+ \wedge l \neq i} c_{il}$$

  - c_{il}은 아이템 l의 i과의 유사도
  - C는 item-correlation/item-similarity matrix로, 모델 파라미터인 Θ = C 임.
- 일반적 C의 선택은 cosine vector similarity와 같은 heuristic similarirty를 사용함

$$c_{i,j}^{\text{cosine}} := \frac{|U_i^+ \cap U_j^+|}{\sqrt{|U_i^+| \cdot |U_j^+|}}$$

- 이보다 좋은 방법은 유사성 척도인 C를 학습하는 것
  - 방법 1) model parameter로 C를 직접 사용 : 이 방법 사용

- 이 때 gradient의 derivative는 다음과 같음

$$\frac{\partial}{\partial\theta}\hat{x}_{uij} = \begin{cases} +1 & \text{if } \theta \in \{c_{il}, c_{li}\} \wedge l \in I_u^+ \wedge l \neq i, \\ -1 & \text{if } \theta \in \{c_{jl}, c_{lj}\} \wedge l \in I_u^+ \wedge l \neq j, \\ 0 & \text{else} \end{cases}$$

- Regularization 상수로 두 개 사용
  - λ_+: update on c_{il}
  - λ_-: update on c_{jl}

- 방법 2) 아이템 수가 너무 많을 때, C를 HH^T로 factorization. $H : I \times k$

Relations to other methods (minor)

WR-MF(Weighted Regularized Matrix Factorization)

- implicit feedback으로 item prediction하므로, $\hat{X} := WH^t$와 같음
- but optimization criterion이 SVD + regularization임: least square
  - + error function에 weigth 줘서 positive feedback의 영향 증대
  - 따라서 optimization criterion이 다음과 같음

$$\sum_{u \in U} \sum_{i \in I} c_{ui}(\langle w_u, h_i \rangle - 1)^2 + \lambda ||W||_f^2 + \lambda ||H||_f^2$$

  - c_{ui}는 model parameter가 아니고, 조건에 따라 주어진 weight임
- Cons
  - Pair가 아니라 one item 기반임
  - item prediction은 regression(quantitative)이 아니라 classification(qualitative)이므로 logistic optimization보다 불리

        - MLE(Maximum Likelihood Estimator)에 대응하는 least squre 방식의 optimization이므로
        - MLE는 가장 설득력 있는 정규분포를 찾는 문젠데 이게 regression에 더 적합하단 뜻인가? **(?)**
            - 근데 이게 아래 MMMF에 쓰인걸 보면 반댄데
- Pros
    - $O(iter(|S|k^2 + k^3(|I| + |U|)))$ 내에 학습 가능
        - c_{ui}가 상수이므로
        - LEARNBPR은 $m \cdot |S|$의 subsample 후에 가능

MMMF(Maximum Margin Matrix Factorization)

- explicit feedback을 위해 만들어졌지만, observed에 1을, 나머지에 0을 줘서 implicit 적용 가능
    - 이 때, MF에 적용된 BPR과 유사해짐

$$\sum_{(u,i,j)\in D_s} \max(0, 1-\langle w_u, h_i - h_j\rangle)+\lambda_w||W||_f^2+\lambda_h||H||_f^2$$

    - Difference
        - 뭔소린지 이해 못함

> One difference is that the error functions differ – our hinge loss is smooth and motivated by the MLE. Ad-

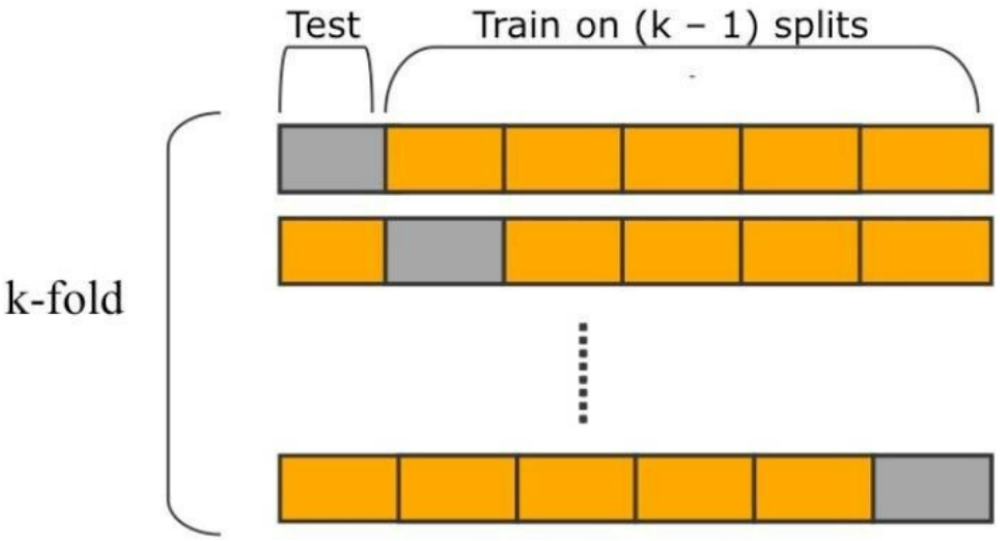
        - BPR-OPT는 많은 모델에 적용 가능한데 MMMF는 MF에만 가능
- Learning method가 LEARNBPR과 다름
    - sparse explicit data 목적으로 만들어져서 implicit data에 적용시 너무 오래걸림

Evalution

- 비교 대상
  - BPR-MF, SVD-MF, WR-MF, BPR-kNN, Cosine-kNN
- Dataset
  - Rossman online shop dataset
  - Netflix DVD rental dataset

Evaluation Methodology

- leave one out 사용
  - n개의 dataset에서 1개를 test set으로 돌아가면서 지정해 n개 sample 생성
  - 이 실험에서는 10개 sample로 테스트



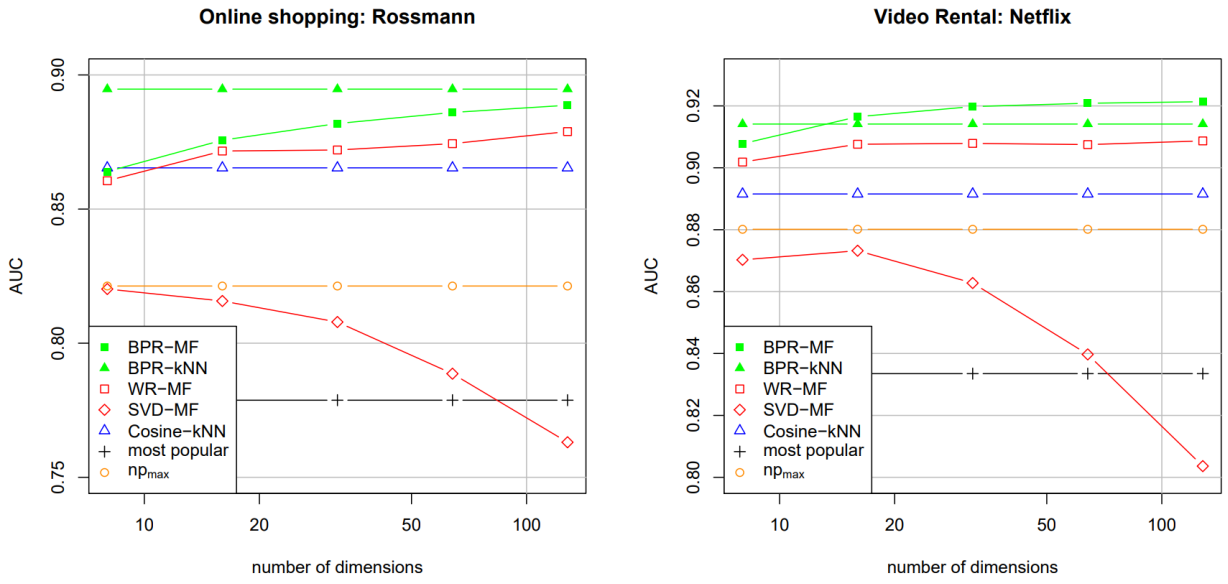
- avg AUC를 평가 지표로 사용

$$\text{AUC} = \frac{1}{|U|} \sum_{u} \frac{1}{|E(u)|} \sum_{(i,j) \in E(u)} \delta(\hat{x}_{ui} > \hat{x}_{uj})$$

where the evaluation pairs per user u are:

$$E(u) := \{(i,j)|(u,i) \in S_{\text{test}} \wedge (u,j) \notin (S_{\text{test}} \cup S_{\text{train}})\}$$

Result



- 다른 모든 방법보다 우수함
- thoretical uppder bound of non-personalized보다 우수함
  - 개별 user가 아닌 전체 user에 대해 예측하는 것

Conclusion

BPR-OPT

- Bayesian analysis를 활용한 optimizaiton criterion 제시

LEARNBPR

- bootstrap sampling 활용한 stochastic gradient descent 제시

Optimization criterion importance

- 실험 결과로 볼 때, model 뿐 아니라, opt 기준또한 중요함

개인적) Netflix의 데이터를 rating 할지 말지를 implicit으로 사용하는 게 설득력이 낮아 보임

- 이론상 상정한 implicit data는 물건을 클릭하는 등의 상호작용이지만, rating하는 행동 자체를 implicit으로 간주하기에는 무리가 있어 보임. 같은 user behavior로 간주하기 어려움
    - 이 때문에 수치가 잘 나왔을 수 있음
- 하지만, 다른 데이터셋에도 적용이 잘 되는 것을 보니, 이 경우에도 사용할 수 있음

AUC 계산할때 그냥 라이브러리 참조해보기 : scikit