Introduction

추천 시스템은 두 가지 전략으로 나눌 수 있다.

1. Content based approach

   : 유저 또는 아이템의 특징으로 프로파일 생성, 매칭

   - 예시 : 장르, 예시, 인기도 등
   - 단점 : 각각의 프로파일을 생성하기 위한 데이터 수집, 구성하려는 노력이 필요, 유저의 content profile 벗어나는 추천은 어렵다

2. Collaborative Filtering

   : 유저-아이템 사이 상호관계를 이용한 유사성 기반 매칭

   - 예시 : 유저의 아이템 구매 내역
   - 단점 : Cold Start(초기 데이터 부족)

입력 데이터를 **Explicit / Implicit** 로 구분.

1. Explicit Feedback

   : 직접 표현한 선호도

   - 예시 : 평점, 좋아요 기능
   - 단점 : 데이터 확보 어려움(평가하지 않고 시청 등)

2. Implicit Feedback

   : 유저 행동 기반 관측치

   - 예시 : 검색 패턴, 마우스 움직임

Explicit Feedback는 데이터 수집 측면에서 비효율적이라 Implicit Feedback으로 구상함

그러기 위해 고려해야할 Implicit Feedback 의 4가지 특성

1. No Negative Feedback : 선호에 대해서는 알 수 있지만 비선호는 나타내기가 어렵다.
2. Implicit feedback is inherently noisy : 유저 행동을 추적한다고 해서 정확한 동기와 선호도 추측이 어렵다.

3. The numerical value of implicit feedback : 한 번의 이벤트는 다양한 이유로 인해 발생되기 때문에 implicit feedback 의 수치는 preference 가 아닌 confidence를 나타냄
4. Evaluation of implicit-feedback recommender requires appropriate measures : Explicit는 예측값과 실제값을 비교해도 되지만 Implicit는 실제값을 정해야 하기 때문에 평가 척도에 대한 고민이 필요함

Previous work

1. Neighborhood models

rui : 해당 user(u)가 item(i)에 대한 평가 (비슷한 취향의 데이터 가중평균)

sij : 아이템 i, j 의 유사도 (피어슨 상관계수)

S^k(i;u) : 아이템 i와 유사한 아이템 k개의 유저 u의 선호도

$$\hat{r}_{ui} = \frac{\sum_{j \in S^k(i;u)} s_{ij} r_{uj}}{\sum_{j \in S^k(i;u)} s_{ij}}$$

user-oriented < item-oriented (정확도 높고, 예측 모델에 대한 설명이 용이해서 자주 이용된다)

Explicit의 경우, user <-> item 간의 bias를 보정하여 향상시킬 수 있지만 Implicit에는 적용이 어렵다. (스케일이 모두 다르고, 유사도 계산이 명백하지 않기 때문)

2. Latent factor models

user <- item 상관관계를 Item latent vector / User latent vector의 행렬 분해로 설명

$$\min_{x_\star, y_\star} \sum_{r_{u,i} \text{ is known}} (r_{ui} - x_u^T y_i)^2 + \lambda(\|x_u\|^2 + \|y_i\|^2)$$

Xu : user 에 대한 latent vector

Yi : item 에 대한 latent vector

λ : 내적의 과적합을 위한 regularized term

Our Model

Implicit Feedback 에 적용가능 한 **Latent factor model**을 제안

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

유저 u가 아이템 i에 대한 preference를 나타내는 binary variable Pui 를 정의. (행동 했으면 1, 아니면 0)

선호도 Pui 만으로 관측치에 대한 설명이 부족하기 때문에 confidence Cui 필요. (아이템의 존재를 모르거나 구입할 수 있는 환경이 안 됐을수도 있고 선호하지 않아도 어떤 이유로 구입했을 수 있다.)

하지만 반복적인 행동은 선호의 증거가 될 수 있어 rui 가 커짐에 따라 비례하는 confidenct cui를 정의.

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

α 에 따라, rui에 대한 변동을 통제. 논문에서는 α = 40으로 정했지만, confidence cui는 고정된 수치로 계산되는 값이기 때문에, 최적화해야 할 parameter는 아님.

preference pui 와 **confidence cui** 를 반영하여 최적의 **user,item latent factor**를 찾기 위해 **loss function**을 정의.

$$\min_{x_\star, y_\star} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

이 loss function은 SGD(Stochastic Gradient Descent) 또는 ALS(Alternating Least Squares)를 통해 최적화할 수 있음.

- 행렬의 내적으로 이뤄지는 SGD 방법은 Non-convex 문제가 된다. (Explicit 에서는 사용 가능)
- 논문에서는 ALS방식을 사용하는데, (X 학습 시 Y 상수 취급 / Y 학습 시 X 상수 취급) convex으로 바꾸기 위함이다.

Y(item vector) 고정 후 X(user vector)에 대한 미분의 결과는 아래와 같다.

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u)$$

Y에 대한 미분도 같은 방식으로 이루어진다.

Explaining recommendations

이 논문의 강점은 설명가능한 추천이다. (왜, 어떠한 user한테 특정 item이 좋은지 나타낼 수 있기 때문)

기존 neighborhood model들과 비교하여 latent factor model들은 설명이 까다로웠다. (모든 과거 유저들의 행동과 추천 결과의 관계가 user-factor들을 경유해 추출되기 때문)

이 논문은 아래와 같은 방법으로 compute하는 방법을 설명한다.

user(u)의 item(i)에 대한 preference 는

$$\hat{p}_{ui} = y_i^T x_u$$

이므로 pui 를

$$y_i^T (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u)$$

으로 표현할 수 있다.

$$(Y^T C^u Y + \lambda I)^{-1} \text{ as } W^u$$

user(u)에 대한 가중치 매트릭스를 Wu라고 표현하면

user(u)의 관점에서 item i, j의 가중 유사도 suij 는

$$s_{ij}^u = y_i^T W^u y_j$$

즉 preference pui는 아래와 같이 표현된다.

$$\hat{p}_{ui} = \sum_{j:r_{uj}>0} s_{ij}^u c_{uj}$$

분리된 pui를 구함으로서 사용자의 이전 행동들의 공헌도를 가질 수 있게 된다.

- 사용자의 이전 행동 : user(u) - Cui 관계의 중요성 / item(i) - Suij의 유사성

Experimental study

TV의 시청기록에 관한 데이터로 실험을 하였고 4주 간의 학습 데이터로, 그 후 1 주를 테스트 데이터로 구성 후 전처리 3단계를 거쳤다.

1. 각 유저마다 test data에 있는 item이 train에도 있으면 제거.

   `주기적 시청의 추천은 의미가 없다`

2. test data에서 0.5시간 이하는 0으로 처리.

   `유저의 호감으로 보기에 약하다`

3. 채널을 바꾸지 않고 재생되는 연속적인 program은 down-weighting

   `단순히 채널을 변경하지 않았을 수 있음`

rui가 정확한 값이라고 하기엔 Implicit Feedback이기 때문에 rank를 제안했음.

rank는 여러 item의 percentile_ranking 수치를 나타내고,

0 % = item(i)에 대해 가장 추천된 item

50 % = 중간 정도로 추천된 item

$$\overline{rank} = \frac{\sum_{u,i} r^t_{ui} rank_{ui}}{\sum_{u,i} r^t_{ui}}$$

평가 결과

본 논문의 Our model의 factor 수를 10~200까지 시도했고, 다른 2개의 모델과 같이 나타내 보았다.

1. popularity : user 상관없이 program 인기 순대로 추천하는 모델
2. neighborhood based : 모든 neighbor set 이용하여 cosine similarity 사용

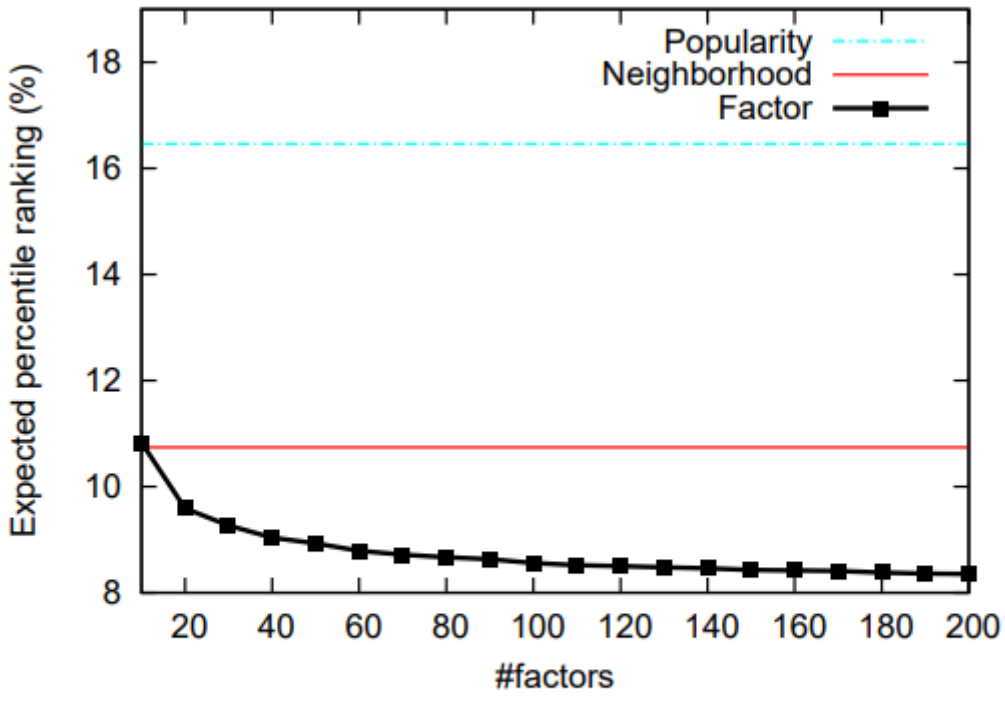

Figure 1. Comparing factor model with popularity ranking and neighborhood model.

결과에 대한 설명

- popularity : factor 수와 관계없이 rank = 16.46 %
- neighborhood based : factor 수과 관계없이 rank = 10.74 %
- Our model : factor가 증가할수록 rank가 줄며 200일 경우 rank = 8.35 %

  또한, 상위 1 %의 추천 program이 전체 시청시간의 27 %

TV 시청 데이터는 주마다 반복시청의 경향을 감안하여 주기적 시청의 영향을 데이터에서 제거하는 전처리 첫 번째 과정을 생략한다면 성능은 더욱 좋아질 것으로 예측

결론

- ALS를 적용하여 학습을 효율화했고 기존 latent 모델과 달리 추천의 설명이 가능한 모델.
- Implicit Feedback의 문제점을 개선하기 위해 기존의 latent 모델에서 rui 대신에 pui 와 cui를 도입하여 모델의 성능을 개선.
- 0과 관련된 모든 user-item 쌍들을 모델에 반영할 수 있음