Netflix

Introduction

- Neighborhood models은 localized relationship을 잘 capture하고 Latent factor model은 global relationship을 잘 capture함

→ 그렇다면 이 두가지를 combine하면 두 방법의 장점 모두 가져올 수 있지 않을까 제안한 논문

Feedback

- Explicit Feedback
  - 5-Stars Rating
  - Thumbs-up / down
    - 이러한 Explicit Feedback은 항상 구하기 어려울 뿐만 아니라 user biased, item biased된 형태를 가지고 있다
    - Implicit Feedback을 사용할 필요성 제시
- Implicit Feedback
  - Purchased History
  - browsing history
  - search patterns
  - mouse movements

현재까지 Explicit Feedback만 이용되는 경향이 큰데 이러한 Implicit Feedback까지 반영하면 더 좋지 않을까 제안

Neighborhood Model

Baseline model

$b_{ui} = \mu + b_u + b_i$

b_u : user-bias, b_i : item-bias, μ : global mean

Loss

$$\min_{b_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 (\sum_u b_u^2 + \sum_i b_i^2)$$

- SSE에 더해 Regularizing Term이 존재
- User-item Rating Matrix는 sparse matrix기 때문에 overfitting의 위험이 크다
- 즉, overfitting을 방지하기 위해 parameter의 magnitude를 제한

Neighborhood Models

Correlation Neighborhood Model

$s_{ij} := \frac{n_{ij}}{n_{ij}+\lambda_2}\rho_{ij}$

n_{ij} : # of users rated both i and j, $\lambda_2 : 100\ (typical\ value)$

- item i, j가 얼마나 유사한지를 단순히 Pearson Correlation Coefficient를 사용한 게 아니라 shrinkage factor를 곱해준 형태
- 이는 앞서 말했듯이 User-item Rating Matrix는 sparse한 특징을 가지고 있기 때문에 실제로 비슷하지 않음에도 불구하고 서로 적게 평가가 되어 있다면 비슷하다고 잘못 평가할 가능성이 있기 때문에 shrinkage factor를 곱해주는 형태
- 즉, shrinkage factor가 신뢰도를 의미함
- 서로 같이 평가가 많이 될수록 신뢰도가 높다고 평가하여 PCC를 전부 반영하고 평가된 수가 적을 때는 신뢰도가 낮다고 평가하여 0에 가깝게 반영

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

- 이러한 similarity를 이용해 weighted-average 방식을 취함

Interpolation weights Model

- 위의 Correlation Neighborhood Model과 같은 경우 직관적이고 구현하기 쉽다는 장점이 있지만

- Global effects를 잘 고려하지 못한다는 문제점
- item들간의 interaction을 고려하지 못한다는 문제점
- 분모를 $\sum s_{ij}$로 나눠주게 되면 결국 원래의 유사도보다 더 큰 비중으로 반영되어 정확도가 낮춰질 여지 있음
- 사람마다 # of rating이 다르다면 성능이 안 좋을 가능성 존재

위와 같은 문제점이 있기 때문에 Interpolation weight를 이용한 새로운 모델이 제시되었다.

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in \mathrm{S}^k(i;u)} \theta_{ij}^u (r_{uj} - b_{uj})$$

Latent Factor Model

Standard SVD

$$\hat{r}_{ui} = b_{ui} + p_u^T q_i$$

- 일반적인 SVD 모델
- User와 Item을 Latent vector로 representation

Paterek's NSVD Model

$$b_{ui} + q_i^T \left(\sum_{j \in \mathrm{R}(u)} x_j \right) / \sqrt{|\mathrm{R}(u)|}$$

- each user에 대해 explicitly parameterizing하기보다 user가 평가한 아이템을 바탕으로 모델링

Revised Model

Neighborhood Model

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) w_{ij} + \sum_{j \in \mathrm{N}(u)} c_{ij}$$

- 위에서 언급했던 Interpolation weight은 user-specific한 weight을 사용했기 때문에 global한 structure를 담기 어려웠다.
- 즉, specific user와 독립적인 global weight를 사용하여 global optimization을 용이하게 함
- 이 모델의 w_{ij}에 대해 저자는 offset를 계속해서 더해가는 형태라고 설명을 하였는데, 유저가 Explicit Feedback을 표현 아이템 사이에서 베이스라인과 얼마나 차이가 나는지를 유사도에 비례에서 반영해주는 형태다
- 사격의 개념으로 이해하자면 b_{ui}로 표현되는 baseline estimator에서 영점 사격을 통해 과녁을 맞춰나가는 방식 생각
- 또한 c_{ij}라는 새로운 term을 추가하여 Implicit feedback을 표현
  - 이때, $N(u)$는 user u의 Implicit Feedback을 표한 아이템 집합인데, Netflix 데이터셋은 Implicit Feedback 데이터가 없기 때문에 Rating의 유무를 1/0로 표시
  - 즉, 이 데이터셋에 한하여 $R(u) = N(u)$

$$\begin{aligned}\hat{r}_{ui} =& \mu + b_u + b_i + |\mathrm{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) w_{ij} \\ &+ |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} c_{ij}\end{aligned}$$

- 위와 같은 형태는 Feedback에 유무에 따라 estimate가 심하게 바뀌기 때문에 보정해줄 수 있는 장치인 $|R(u)^{-1/2}|, |N(u)^{-1/2}|$를 각각 곱해주었다
- 지수가 -1이 되어야 정확한 normalizing factor지만 -1/2를 사용한 것은 극심한 차이는 아니더라도 feedback을 표현 집합과 아닌 집합을 어느 정도는 구분하고 싶었던 저자의

의도가 아니었을까

$$\hat{r}_{ui} = \mu + b_u + b_i + |\mathrm{R}^k(i;u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}^k(i;u)} (r_{uj} - b_{uj}) w_{ij} + |\mathrm{N}^k(i;u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}^k(i;u)} c_{ij} \quad (10)$$

- 모든 아이템에 대해 계산하는 것이 아니라 K개의 가장 가까운 아이템에 대해서만 계산을 수행해 Parameter Pruning을 진행한다.

Comparison of Neighborhoods

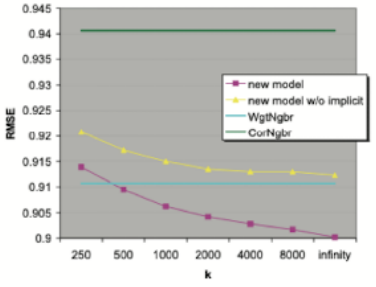


Figure 1: Comparison of neighborhood-based models. We measure the accuracy of the new model with and without implicit feedback. Accuracy is measured by RMSE on the Netflix test set, so lower values indicate better performance. RMSE is shown as a function of varying values of k, which dictates the neighborhood size. For reference, we present the accuracy of two prior models as two horizontal lines: the green line represents a popular method using Pearson correlations, and the cyan line represents a more recent neighborhood model.

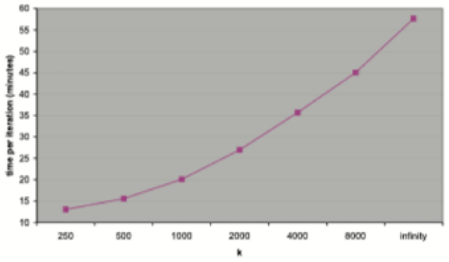


Figure 2: Running times (minutes) per iteration of the neighborhood model, as a function of the parameter k.

- 저자가 새롭게 제안한 모델이 k가 500일 때부터 계속 가장 좋은 성능을 내고 있음을 확인할 수 있음
- 단, 250일 때는 Interpolation weight을 사용한 모델이 가장 좋음
- 또한 k에 따른 1-iteration에 걸리는 시간을 오른쪽으로 나타내어 k값에 따라 RMSE Performance와 Running Time에 Trade off가 있음을 보여줌

→ 적절한 K값 선정이 필요함

Asymmetric-SVD

$$\hat{r}_{ui} = b_{ui} + q_i^T \left(|\mathrm{R}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{R}(u)} (r_{uj} - b_{uj}) x_j + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j \right)$$

- Paterek's NSVD를 Improve한 형태
  - 일반적으로 User가 Item에 비해 더 많기 때문에 Fewer Parameter를 구성할 수 있음
  - 새로운 유저가 들어와도 몇번만 그 유저가 몇 가지 아이템만 평가해도 쉽게 User에 대한 vector를 구성할 수 있음
  - 어떻게 rating에 대한 estimate가 일어나는지 쉽게 확인할 수 있기 때문에 explainability가 늘어남
  - Implicit Feedback 부분에 해당하는 y_j가 추가되어 효율적인 통합이 가능

SVD++

$$\hat{r}_{ui} = b_{ui} + q_i^T \left(p_u + |\mathrm{N}(u)|^{-\frac{1}{2}} \sum_{j \in \mathrm{N}(u)} y_j \right)$$

- 일반적인 SVD에 Implicit Feedback 부분만 추가한 모델

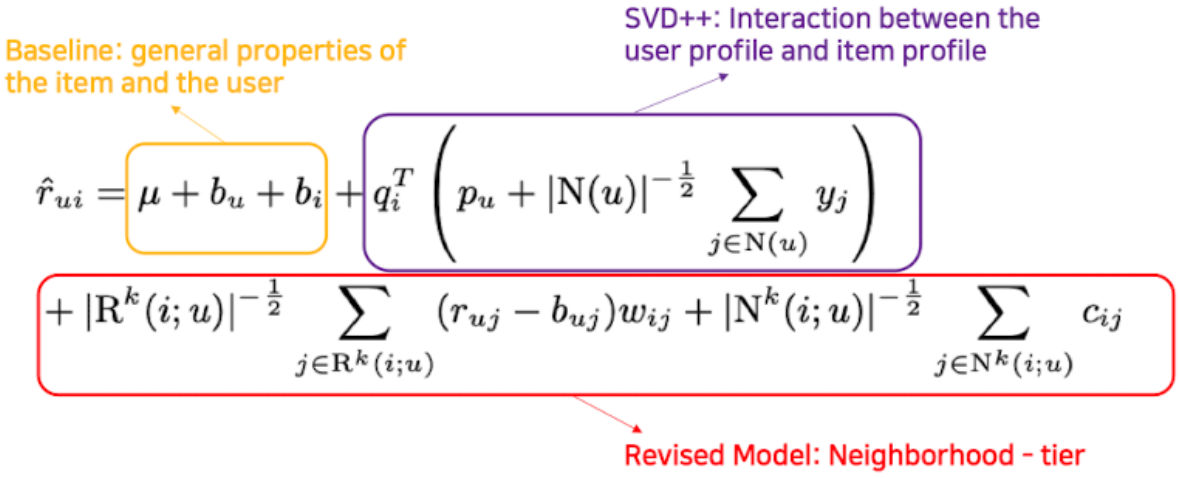
Comparison of SVDs

Model	50 factors	100 factors	200 factors
SVD	0.9046	0.9025	0.9009
Asymmetric-SVD	0.9037	0.9013	0.9000
SVD++	0.8952	0.8924	0.8911

Table 1: Comparison of SVD-based models: prediction accuracy is measured by RMSE on the Netflix test set for varying number of factors (f). Asymmetric-SVD offers practical advantages over the known SVD model, while slightly improving accuracy. Best accuracy is achieved by SVD++, which directly incorporates implicit feedback into the SVD model.

- SVD++가 가장 좋은 성능을 나타내고 있기 때문에 Integrated Model에서도 SVD++ 사용

Integrated Model



- Integrated Model은 3-tier로 합쳐져있음
  - Baseline: user와 item의 일반적인 특징을 반영
  - SVD++: User Profile과 item profile의 interaction을 반영
  - Revised Model: 아이템간의 Interaction을 반영

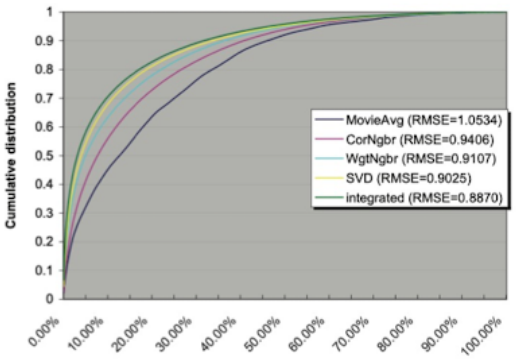
	50 factors	100 factors	200 factors
RMSE	0.8877	0.8870	0.8868
time/iteration	17min	20min	25min

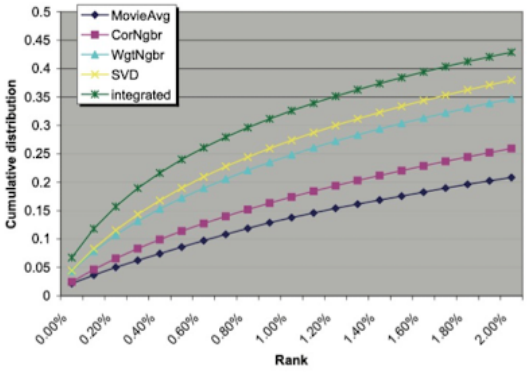
Table 2: Performance of the integrated model. Prediction accuracy is improved by combining the complementing neighborhood and latent factor models. Increasing the number of factors contributes to accuracy, but also adds to running time.

- 다른 모델에 비해 가장 낮은 RMSE를 기록함

Top-K Recommender

- 저자는 실질적으로 조금 낮아진 RMSE가 유저에게 정말 좋은 추천을 제공할지 의문이 들었다고 주장
- 이를 측정하기 위해 새로운 Metric인 Top-K recommender 방식 제안
  - 각 유저마다 5점을 준 Item을 하나 뽑음
  - 1000개를 아이템 집합에서 Random Sampling
  - 1001개의 데이터를 모델에 태워서 Rating을 바탕으로 Ranking Prediction
  - 처음에 뽑았던 5점짜리 아이템이 앞순위에 와야 옳게된 모델이라고 판단
  - 이러한 과정을 모든 유저, 모든 5점에 대해 반복적으로 진행





- 위의 과정을 Cumulative Distribution으로 나타낸 그래프

- 왼쪽 Figure를 통해 RMSE가 낮은 모델이 실질적으로 좋은 Recommendation을 제공하고 있음을 확인할 수 있음
- 오른쪽 Figure 같은 경우는 우리가 실질적으로 유저에게 추천되는 것은 1000개 중에 소수 (10개 내지 20개)기 때문에 그 부분에서 아이템이 잘 제공되고 있는지 확대
    - Integrated Model이 가장 좋은 성능을 내고 있음을 확인할 수 있음