**OCCF 발표**

제가 이번에 발표할 논문은 OCCF : collaborative Filtering for Implcit Feedback Datasets 입니다. 2008년에 IEEE. ICDM에서 발표된 논문입니다.

이 논문에 대해 리뷰하기 전에, 이 논문의 background인 implicit feedback과 collaborative Filtering에 대해 먼저 설명하겠습니다. User는 특정 iitem에 대한 feedback을 데이터로 남기게 되는데, feedback은 두 가지로 나뉘어집니다. 평점, 좋아요, 리뷰등의 직접적인 explicit feedback과, 구매이력, 클릭, 방문 횟수 등의 간접적인 형태의 implicit feedback으로 나뉘어집니다.

추가적으로 이 논문에서 다루는 implicit feedback 4가지의 특징이 있습니다.

Implicit feedback에는 부정적인 평가가 담겨져 있지 않다. 유저의 의도를 파악하기 힘들다, 데이터 값 자체가 선호도라고 판단할 수 없다. 단순한 metric으로는 평가하기 힘들다. 라는 점들이 있습니다.

recommendation system은 이러한 feedback 데이터를 이용해서 user가 좋아할만한 item을 추천하는건데, 이 논문 이전에는 Collaborative filtering 방법을 주로 사용하였습니다. 이 방법은 neighborhood model, latent factor model 2가지로 나뉘어지게 됩니다. neighborhood model은 말 그대로 유사한 유저, 유사한 아이템들을 고려해서 추천하는 모델입니다. user-oriented는 비슷한 user들의 선호도를 보는 방법이고, item-oriented는 user가 좋아하는 item과 유사한 Item들을 추천하는 방법입니다. 수식으로 표현하였을 때, 추천 item의 선호도는 다른 Item과의 유사도 곱하기 선호도의 합으로 표현됩니다.

collaborative filtering의 두 번째 방법은 latent factor model입니다. 이 방법은 user와 item rating의 matrix를 SVD를 통해 user와 item을 f차원의 벡터로 표현하는 방법입니다. 그렇게되면 user의 item에 대한 선호도는 두 벡터의 inner product로 표현할 수 있게 됩니다. loss function은 보이는 것 처럼 구성되어있는데, 좌측은 실제값과 prediction의 차이이고, 우측은 overfitting을 방지하기 위한 regularization입니다.

그런데 이제, collaborative filtering은 explicit feedback dataset을 이용하는 recommedation system입ㄴ디ㅏ . 하지만 explicit feedback dataset은 user가 직접 평가해야하기 때문에 항상 얻을 수 있는 데이터가 아니고, 수집하는 양에 대해서도 어느 정도 한계가 있습니다. 그래서 이 논문에서는 implicit feedback dataset을 이용하는 collaborative filtering 방법을 제시하고 있습니다.

이 논문에서 제시한 모델은 CF의 두번째 방법인 latent factor model을 발전시킨 버전입니다.

이 방법은 총 3가지 단계로 나눌 수 있습니다.

첫 단계는 implicit feedback의 특징인 numerical value가 preference를 표현하지 않는다는 점을 보완하는 부분입니다. 그렇기 때문에 observed value를 두 형태로 재표현하게됩니다. user가 item을 한 번이라도 소비한 적이 있다면, preference를 1로 표현하고, 소비한 적이 없다면 0으로 표현합니다. 그리고 여기에 user가 해당 item에 대해 선호한다는 것을 표현하는 confidence라는 값을 도입하였습니다. 이 값은 observation 값을 반영해서 해당 식으로 표현됩니다. alpha는 하이퍼파라미터인데, 논문에서 40의 값을 사용하였을 때 가장 잘 되었다고 소개하고 있습니다.

Preference와 confidence 값을 반영하기 위해, latent factor model에서 사용하였던 loss function은 다음과 같이 바뀌게 됩니다. 기존의 explicit feedback이 들어가는 곳에 preference로 바뀌고 앞에 confidence 값이 들어가게 됩니다.

두 번째 단계는 loss function computation 입니다.

loss function computation을 이제 모든 user, item pair에 대해서 진행을 해야하는데, 그 pair가 수십억은 넘게 됩니다. 그래서 gradient descent 방법으로 optimization을 할 수가 없는데, 이 논문에서는 alternative least square라는 다른 방법을 사용합니다. user를 나타내는 x, item을 나타내는 y 중 하나를 상수로 취급하는 편미분을 통해 optimization 지점을 찾습니다. 논문에서는 10번 정도 반복하여 찾은 값들로 충분히 optimization이 되었다고 판단합니다.

세 번째 단계는 computation을 줄이기 위한 또 다른 트릭입니다.

저 식을 계산하기 위해서는 한 user에 대해서 O(f^2 n)이 걸립니다. 하지만 다음 처럼 식을 바꿀 수 있는데, 이렇게 바꾸게 되면 기존의 C 값들은 observed value에 1을 더한 값이기 때문에 1을 빼게 되면 각 user와 item pair에 대해 nonzero value만 남기 떄문에 n이 아니라, nonzero value 갯수 만큼 줄어들기 때문에 더 빠른 연산이 가능합니다. 일반적으로는 nonzero value가 전체 갯수에 비해 매우 적기 때문에 computation이 절약되게 됩니다.

그리고 이 논문에서 해당 모델의 강점이라고 말하는 것은 explainable한 recommendation이라는 것입니다. user의 item에 대한 선호도는 이처럼 표현되는데 이 때 Wu를 이렇게 정의하고, similarity i,j는 이처럼 표현됟기 때문에 이 식을 해당 모델의 factor 식에 대입하면 이렇게 표현됩니다. 이 때 중간 부분을 u에 대한 Weighting matrix로 표현하고 뒤쪽은 특정 user에게 타켓 item과 다른 item간의 유사도로 표현되므로 최종적인 식은 아래와 같이 표현됩니다. 이는 결국 유사한 item과 confidence의 곱이기 때문에 결국 explainable하다고 말합니다.

논문에서는 TV show 시청 시간에 대한 데이터를 이용해서 4개의 실험을 진행하였습니다.

그리고 실험 이전에 data preprocessing 과정을 거쳤는데,

첫 번째는 ~~

두 번째는 implicit feedback data의 noisy 함을 줄이기 위해서 사용한 방법입니다. 채널을 변경하지 않고 프로그램이 계속 시청되는 경우 가중치를 작게 하는 방법입니다.

세 번째는 프로그램의 절반도 보지 않은 경우에는 선호가 있다고 판단하기 힘들기에 0으로 처리

네 번째는 recommendation의 목표는 user가 경험하지 않은 item에 대한 추천이기 때문에, user가 시청한 프로그램은 test set에서 지우는 작업을 했습니다.

앞서 말했던 log scaling과 momentum issue는 다음과 같은 식을 사용하였고, momentum의 경우 실험적으로 a=2, b=6일 때 최적의 결과를 얻었다고 합니다.

해당 논문에서 evaluation metirc은 rank bar 라는 값을 이용했는데, 이는 false negative를 계산할 수 없는 implicit feedback으로 인해 만들어진 평가 지표 입니다. 추천한 item의 순위와 실제 test set 값의 곱을 평균내어 계산한 값으로, rank bar 값이 작을 수록 높은 성능을 가진 것이고, 무작위 랜덤의 경우는 50%의 값을 가지게 됩니다.

본격적으로 실험에서는

그냥 인기 많은 TV show를 추천하는 모델

모든 TV show에 대해서 유사도를 매겨서 추천하는 모델

그리고 논문에서 제시한 모델

3가지를 가지고 진행됩니다.

이 3가지 모델을 가지고 4개의 실험을 진행하였는데

첫 번째는 각자의 모델에 대해서 rank bar score를 뽑는데, paper model의 경우는 factor 값을 10 ~ 200까지 바꾸어가면서 실험을 진행하였습니다.

두 번째는 추천의 quality를 평가하는 실험인데, 이해하기 쉽게 설명을 하면 추천한 item list 중 상위 5%가 실제로 소모됐을 확률을 계산하여 비교하였습니다.

세 번째는 모델의 performance가 item의 popularity와 watch time에 의해 영향을 받는지 알아보는 실험입니다. test set의 item을 popularity에 따라 15개의 bin으로 나누고 각 bin에 대해 rank bar score를 뽑았고, 이와 같은 방법으로 watch time에 대해서 진행하였습니다.

네 번째는 paper model에서 채택한 loss function 뿐만 아니라 기존 latent factor model의 식을 사용했을 때의 결과와 비교하는 실험입니다.

첫번째 실험의 경우 factor가 높아가면서 paper model의 성능이 점차 안정되고 다른 model에 비해 매우 높은 성능을 보여주었습니다.

두 번째 실험도 마찬가지로 top 5% 이내에 어느 구간에 대해서나 paper model이 타 모델에 비해 추천 quality가 좋았음을 보여주었습니다.

세 번째 실험에서는 파란 line이 popularity인데, popularity가 높아 지면서 굉장히 높은 performance를 보여줍니다. 하지만 watch time에 대해서는 큰 영향이 없음을 보여줍니다.

네 번째 실험에서는

paper model 외로. latent factor model의 식을 사용하였을 때, 네이버후드 모델보다 낮은 값이 나왔고, confidence value를 사용하지 않았을 때는 네이버후드보다는 나았지만, paper model 보다는 성능이 안 좋게 나왔습니다. 이 실험을 통해서 preference 값과, confidence 값을 사용하는 해당 논문의 방법을 검증하였습니다.

BPR

BPR은 선호도가 아닌 ranking을 부여하는 것에 집중하는 모델이고 item pair끼리 순위를 비교…

기존에는 소비되지 않은 item에 대해 0이라는 값으로 치부했으나, 논문에서는 소비된 item ranking > 소비되지 않은 item ranking 이라는 pair로 묶어서 데이터로서 활용

-> ranked pair 들을 이용해서 bayesian optimizing을 formularization 하고 이를 하나의 criterion으로 정리하고 이를 MF, kNN에 적용해서 실제 데이터에 실험하여 좋은 결과를 얻음