협력적 필터링과 콘텐츠 정보를 결합한 영화 추천 알고리즘

(A Movie Recommendation Algorithm Combining Collaborative Filtering and Content Information)

김 상 화 † 오 병 화 †† 김 문 종 ††† 양 지 훈 ††††
(Sanghwa Kim) (Byonghwa Oh) (Moonjong Kim) (Jihoon Yang)

요 약 추천 시스템은 사용자의 관심을 고려하여 새로운 추천 아이템을 예측한다. 대부분의 추천 시스템은 협력적 필터링 또는 콘텐츠 기반의 방법을 사용한다. 두 방법은 각자의 장점을 가지고 있지만, 하나만 사용할 경우에는 상황에 따라 알맞은 추천 결과를 내지 못하기도 한다. 두 방법의 이점들을 통합한 하이브리드 추천 시스템은 각 방법들의 단점을 극복할 수 있다. 본 논문에서는, 콘텐츠 기반의 방법과 협력적 필터링을 효과적으로 통합할 수 있는 구조를 소개한다. 제안한 방법은 하나의 기법만 이용한 모델에 비해 높은 성능을 보여주었다.

키워드 : 추천 시스템, 협력적 필터링, 콘텐츠 기반 추천, 영화 추천

Abstract Recommender Systems attempt to predict new items considering interest for a user. Most recommender systems use Collaborative Filtering or Content-based methods. While both methods have their own advantages, individually they fail to provide good recommendations in many situations. Incorporating components from both methods, a hybrid recommender system can overcome these shortcomings. In this paper, we present an effective framework for combining content-based filtering and collaborative filtering. The proposed algorithm performs better than a pure content-based predictor or pure collaborative filter.

Key words : Recommender System, Collaborative Filtering, Content-based Recommendation, Movie Recommendation

· 본 연구는 정보통신산업진흥원의 IT/SW 창의연구과정의 연구결과로 지식경제부와 마이크로소프트에 의해 지원된 과제로 수행되었음(NIPA-2011-43010)
· 이 논문은 제38회 추계학술발표회에서 '협력적 필터링과 콘텐츠 정보를 결합한 영화 추천 알고리즘'의 제목으로 발표된 논문을 확장한 것임

† 비 회 원 : 다음커뮤니케이션 FT기술팀 팀원
kojak17@hanmail.net
†† 학생회원 : 서강대학교 컴퓨터공학과
mrfive@sogang.ac.kr
††† 비 회 원 : 서강대학교 컴퓨터공학과
penbell@naver.com
†††† 종신회원 : 서강대학교 컴퓨터공학과 교수
yangjh@sogang.ac.kr
(Corresponding author임)

논문접수 : 2011년 12월 5일
심사완료 : 2012년 1월 16일

정보과학회논문지: 소프트웨어 및 응용 제39권 제4호(2012.4)

1. 서 론

인터넷 서비스 시장의 확장과 서비스 업체들의 성장에 따라 사용자에게는 선택의 기회가 매우 다양해진 반면 기업 간의 경쟁은 더욱 치열해졌다. 특히 인터넷의 양방향 특성으로 말미암아 사용자가 서비스 또는 제품에 대해 내린 평가 및 사용기록 정보를 기업체가 체계적으로 관리할 수 있게 됨으로써, 기업들은 이 데이터를 통해 개인고객에게 특화된 제품 및 서비스를 제공할 방법을 모색하게 되었다. 이러한 전략이 성공하기 위해서는 무엇보다 고객의 요구를 정확하게 파악하는 것이 중요하기 때문에 여러 다양한 해결 방법이 연구되었다.

특히, 고객에게 개인화된 서비스를 제공하기 위해 개개인의 특성, 취향에 맞추어 추천할 콘텐츠를 파악하는 방법을 추천 시스템(Recommender System)[1]이라 한다.

협력적 필터링(Collaborative Filtering)[2]을 기반으로 한 추천 시스템은 아이템과 사용자의 관계를 이용하여 사용자의 흥미에 맞는 아이템을 추천한다. 반면, 콘텐츠

기반의 시스템은 아이템의 내용을 이용하여, 사용자가 어떤 아이템을 좋아한다면 해당 아이템의 속성을 파악하여 동일 속성을 가진 아이템을 추천한다. 즉, 협력적 필터링은 잠재적 특성(Latent Feature)을 기반으로 아이템 및 사용자의 유사도를 측정하지만, 콘텐츠 기반 방법은 가시적 특징들을 파악하여 유사도 측정에 사용한다.

위 두 방법을 결합하려는 시도들[3,4]이 있었지만 이들은 콘텐츠-사용자, 콘텐츠-아이템 사이의 관계 정보를 단편적으로 파악하거나, 또는 제각각 살펴보는 한계로 인해 전체적인 시야를 가질 수 없었다. 본 논문에서는 위 두 정보를 통합적으로 파악하는 협력적 필터링 기반의 추천 시스템을 제시한다. 먼저 이를 위해 제안하는 알고리즘의 기반 구조를 형성하는 특이값 분해(Singular Value Decomposition, SVD)[5]의 개념 및 기존에 이를 사용한 알고리즘들을 설명하고, 콘텐츠 기반의 SVD를 제안한다. 이후 협력적 필터링에서 가장 성능이 뛰어나다고 알려진 기존 알고리즘과의 결합을 통해 향상된 성능의 통합 알고리즘을 제안한다.

2. 협력적 필터링

협력적 필터링은 사용자의 선호도와 관심 표현의 패턴을 분석하여, 비슷한 패턴을 가진 고객들을 선별하고, 이를 기반으로 아이템을 추천하거나 서비스를 제공한다.

2.1 행렬 인수분해 모델(Matrix Factorization)

행렬 분해 모델은 모든 사용자가 부여한 모든 영화의 점수로 구성된 관찰된 목표 행렬(Observed Target Matrix) A를 K차원의 특성 행렬 U와 M으로 분해한다. 그리고 이 두 개의 특성 행렬을 이용하여 결측치(Missing Values)를 예측하게 된다. 영화의 개수를 m, 사용자의 수를 n이라고 하자. 행렬 분해 모델의 목표는 행렬 A에 근사하는 행렬 $\tilde{A}= UM$을 찾는 것이다. 여기서 U와 M은 각각 $U\in\mathbb{R}^{m\times K}$, $M\in\mathbb{R}^{K\times n}$이고, K는 특성 차원이다.

$$\begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mn} \end{bmatrix} \cong \begin{bmatrix} U_{11} & \cdots & U_{1k} \\ \vdots & \ddots & \vdots \\ U_{m1} & \cdots & U_{mk} \end{bmatrix} \cdot \begin{bmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{k1} & \cdots & M_{kn} \end{bmatrix}$$

그림 1 행렬 인수분해 모델

영화 추천 시스템에서 분해된 특성 행렬 U와 M은 각각 사용자와 영화의 특성 벡터로 구성되어진다. 사용자 특성 행렬 U는 각 행이 모든 사용자 i의 특성 벡터 U_i로 이루어져 있고, 영화 특성 행렬 M은 각 열이 모든 영화 j의 특성 벡터 M_j로 이루어져 있다. 이 두 개의 특성 벡터의 내적을 통해 사용자 i가 영화 j에 부여한 점수를 예측할 수 있다. 이 개념을 수식화한 것이 식 (1)이다.

$$\tilde{a}_{ij} = \sum_{k=1}^{K} U_{ik} M_{kj} = U_i^T M_j \quad (1)$$

행렬 분해 방법은 여러 가지가 있지만, 특이값 분해는 명확한 평가 또는 관계 정보가 있을 경우, 이를 바탕으로 행렬을 분해하였을 때 잠재적 요인(Latent Factor)을 잘 정의한다고 알려져 있다[5]. 하지만 전통적인 SVD 풀이법은 사용자가 시청한 영화가 있지만 그에 대한 평가 정보가 없는 경우, 예를 들어 어떤 사용자가 시청한 영화가 있지만 그것에 대한 평가 정보가 없는 경우, 해결 방법이 정의되어 있지 않다. 이로 인해 추천 시스템에서 사용되는 행렬 분해 모델은 기존의 수학적인 방법으로 해결하는 것이 힘들다. 따라서 임의의 특성 행렬 U와 M을 구성하여 관찰된 목표 행렬 A와의 차이 값을 기울기 강하(Gradient Descent)[6] 기법을 사용함으로써 최소화함으로써 특성 행렬을 학습하는 방법이 많이 사용된다.

2.2 특이값 분해

SVD는 추천 시스템에서 사용되는 여러 기법들 중 가장 우수한 알고리즘 중에 하나이다. 여기서는 제안 알고리즘에 사용될, 기울기 강하를 이용한 SVD의 학습 및 이의 확장에 대해 설명한다[6-8].

2.2.1 조정된 특이값 분해(Regularized SVD)

SVD는 잠재 요인을 구하기 위한 저계수(Low-rank) 분해 기법으로, 완전히 관찰된 목표 행렬 A로부터 사용자 및 영화 특성 행렬으로 분해를 시도한다. 여기서 U와 M은 관찰된 목표 행렬 A와의 합-제곱 거리(Sum-Squared Distance)를 최소화하고, K개의 특성의 차원을 가지는 행렬이 된다. 이 설명을 수식으로 표현하면 다음과 같다.

$$A \cong \tilde{A}= UM, \ (U\in\mathbb{R}^{m\times K}, M\in\mathbb{R}^{K\times n}) \quad (2)$$

이 근사식을 계산하기 위해 오류를 실제 점수 a_{ij}와 식 (1)의 예측된 점수의 차로 정의하고 오류의 합-제곱 거리를 최소화한다.

$$\min \sum_{(i,j)\in\Gamma} \left(a_{ij} - \sum_{k=1}^{K} U_{ik} M_{kj} \right)^2 + \lambda(\| U_i \|^2 + \| M_j \|^2) \quad (3)$$

식 (3)은 경사 강하 기법의 갱신 규칙의 기본이 된다. 여기서 Γ는 데이터에서 나타난 모든 사용자, 아이템 쌍이며, $\lambda(\| U_i \|^2 + \| M_j \|^2)$는 과적합 문제(Overfitting Problem)을 막기 위한 조정(Regularization) 작용을 한다.

추천 시스템에 사용되는 데이터는 각 사용자들과 아이템들이 가지는 특성과 관계들이 서로에게 영향을 주어 얻어진 결과이므로, 어떤 사용자가 자신과 관련된 아이템들에 대해 다른 이보다 특히 높은 점수를 주는 경우, 또는 어떤 아이템이 특히 높은 점수를 받는 경우 등 사용자와 아이템 개개의 고유 특성들이 두드러지게 나타나는 경우가 있다. 이런 효과를 반영하기 위해 다음 식과

같은 기준선 예측(Baseline Estimates)을 시도한다.

$$b_{ij} = \mu + b_i + b_j \tag{4}$$

b_{ij}는 사용자 i와 아이템 j간의 관계를 예측하는데 있어서 기본 값이 된다. b_{ij}는 전체 데이터에서 점수의 평균 μ와 b_i, b_j의 합으로 이루어지며, b_i는 각 사용자 i가 영화에 부여한 점수의 평균과 전체 점수의 평균의 차이이며, b_j는 모든 사용자가 영화 j에 부여한 점수와 전체 영화 점수와의 차이이다. 즉 b_i, b_j는 사용자 i, 아이템 j의 평균 점수에서 관찰된 편향 값을 반영한 결과이다. 식 (3)에서는 평점만을 근사하지만, 평점에서 b_{ij}를 뺀 잔차(Residual)를 근사함으로써 고유 특성을 적용한 모델을 학습할 수 있다. 즉, 영화 평점에서 b_{ij}를 뺀 값을 식 (4)를 식 (3)에 적용시키면 식 (5)와 같다. 이렇게 학습된 U와 M을 곱한 후 b_{ij}를 각각 더해줌으로써 평점을 예측할 수 있다.

$$\min \sum_{(i,j)\in\Gamma} \left(a_{ij} - b_{ij} - \sum_{k=1}^{K} U_{ik} M_{kj} \right)^2 + \lambda (b_{ij}^2 + \| U_i \|^2 + \| M_j \|^2) \tag{5}$$

이렇게 정리된 수식은 볼록 문제(Convex Problem)이므로 경사 강하 기법을 이용하여 최적화가 가능하다. 매 학습마다 RMSE(Root Mean Squared Error)를 측정하여, 그 값이 일정 값에 수렴할 때까지 학습을 반복한다. 이러한 접근 방법은 이론적인 수렴성을 보장한다[6]. 훈련 데이터에 대해서 반복적으로 학습하기 때문에 SVD는 과적합 문제에 빠지기 쉽고, 이를 예방하기 위해 λ 이하의 항을 이용한다. 다음은 조정된 SVD의 알고리즘을 나타낸 것이다.

2에서 5번까지의 줄은 각 변수에 대한 갱신 규칙이다. 이들을 유도하기 위해, 최적화할 식 (5)를 각 변수에 따라 편미분하여 구한 변화량에 학습률 γ_1, γ_2를 곱하여 원래 변수의 값에 더하였다. λ_1과 λ_2는 조정 인자(Regularizing Factor)로서 각 변수에 대한 조정 작용의 정도를 조절한다. 알고리즘은 반복을 통해 더 이상 변수들의 값이 변화되지 않는 극소점을 찾는다.

Regularized SVD Algorithm

Initialize: $U \in R^{m\times k}$, $M \in R^{k\times n}$
Repeat until convergence
  For each i, j
1. $e_{ij} = a_{ij} - b_{ij} - \sum_{k=1}^{K} U_{ik} M_{kj}$
2. $b_i \leftarrow b_i + \gamma_1 (e_{ij} - \lambda_1 b_i)$
3. $b_j \leftarrow b_j + \gamma_1 (e_{ij} - \lambda_1 b_j)$
4. $U_i \leftarrow U_i + \gamma_2 (e_{ij} M_j - \lambda_2 U_i)$
5. $M_j \leftarrow M_j + \gamma_2 (e_{ij} U_i - \lambda_2 M_j)$
  End For
End Repeat

2.2.2 NSVD1

NSVD1[9]은 Paterek이 고안한 알고리즘으로 조정된 SVD의 성능을 향상시킨 비대칭 인수 모델(Asymmetric Factor Model)이다. 이의 가장 큰 특이점은 사용자의 특성 행렬을 대신하여 아이템의 특성 행렬을 사용한다는 것이다. 보통 한정된 아이템의 수보다 사용자의 수가 많기 때문에, 다루는 데이터의 크기를 많이 줄일 수 있으며, 더 나아가 성능을 더욱 향상 시킬 수 있음을 보였다.

기본 발상은 사용자의 특성 벡터 U_i를 또 다른 아이템의 특성 벡터들의 합으로 근사 시킬 수 있다는 것이다. $R(i)$는 사용자 i가 점수를 부여한 아이템들의 집합이다. W_j는 $W_j \in \mathbb{R}^K$인 벡터로서, 사용자가 점수를 부여한 아이템 j가 가진 K개의 특성들을 나타낸다.

$$U_i \approx |R(i)|^{-0.5} \sum_{j\in R(i)} W_j \tag{6}$$

이 방법은 어떤 사용자의 특성을 추상화시킨 특성 벡터를 구성하는 대신, 사용자의 과거의 반응, 즉 사용자가 어떤 영화에 부여한 점수만으로 예측을 시도하기 때문에 기존의 SVD와는 다른 관점으로 데이터의 양상을 살펴 볼 수 있다는 특징이 있다.

2.2.3 SVD++

Koren[10]은 기존의 SVD와 NSVD1 알고리즘을 확장시킨 하이브리드 알고리즘을 제안하였다. 사용자가 어떤 아이템을 좋아한다면 높은 점수를, 아니면 낮은 점수를 부여할 것이다. 이를 명백한 피드백(Explicit Feedback)으로 보고, 기존 알고리즘은 이 정보를 이용해 추천을 한다. 암시적인 피드백(Implicit Feedback)은 여기서 더 확장된 개념으로, 어떤 사용자가 점수를 부여한 아이템은 사용자와 아이템 사이 어떤 관계를 맺을 이유가 있을 것이고, 반대의 경우, 즉 점수를 부여하지 않은 아이템들에 대해서도 그러할 것이다. 따라서 사용자 i가 어떤 영화에 부여한 점수 $\tilde{a}_{ij}$는 위 두 요인들이 결합된 결과로 정의한다. 이러한 개념을 식 (6)의 발상을 이용하여 적용하면 다음과 같은 모델을 만들 수 있다.

$$\tilde{a}_{ij} = b_{ij} + M_j^T \left(U_i + |N(i)|^{-0.5} \sum_{j\in N(i)} W_j \right) \tag{7}$$

여기서 $N(i)$는 사용자 i가 점수를 부여하지 않은 아이템들의 집합이며, W는 아이템 특성 행렬이다. 이 수식은 SVD+NSVD1의 형태로 표현된다. 기존 SVD 부분은 사용자의 명백한 피드백을 반영하며, NSVD1 부분은 사용자가 선택하지 않은 아이템들의 특성 벡터들을 평균을 이용하여, 사용자의 또 다른 특성을 이용하여 예측한 점수를 보정해 주는 역할을 한다(Koren은 $N(i)$를 사용자 i가 점수를 부여한 집합으로 정의하여, 선택한 아이템들에 대한 점수를 보정하였으나, 위와 같이 사용할 수도 있다). 식 (7)을 따르는 갱신 규칙은 다음 식

(8)과 같은데, 이는 앞의 SVD에서 M_j에 대해 변경이 적용된 것으로, U_i도 자연히 변경된다. 각 변수에 따라 갱신 규칙을 수렴될 때까지 반복적으로 적용함으로써 오류를 최소화하는 변수들의 값을 찾을 수 있다.

$$\begin{aligned}
&\bullet b_i \leftarrow b_i + \gamma_1 (e_{ij} - \lambda_1 b_i) \\
&\bullet b_j \leftarrow b_j + \gamma_1 (e_{ij} - \lambda_1 b_j) \\
&\bullet U_i \leftarrow U_i + \gamma_2 (e_{ij} M_j - \lambda_2 U_i) \\
&\bullet M_j \leftarrow M_j + \gamma_2 \left(e_{ij} \left(U_i + |N(i)|^{-0.5} \sum_{j \in N(i)} W_j \right) - \lambda_2 M_j \right) \\
&\bullet j \in N(i),\ W_j \leftarrow W_j + \gamma_3 (e_{ij} |N(i)|^{-0.5} M_j - \lambda_3 W_j)
\end{aligned} \tag{8}$$

식 (8)에서, 첫 세 개의 항은 조정된 SVD 알고리즘의 갱신 규칙과 동일하나, 다섯 번째 항에서 W_j에 대한 갱신 규칙이 추가됨으로써 네 번째 항 또한 U_i 대신 변경된 식이 적용되었음을 알 수 있다.

3. 콘텐츠 정보를 결합한 영화 추천 알고리즘

본 장에서는, 콘텐츠 정보와 사용자-영화 관계 정보를 통괄적으로 처리하는 시스템을 제안한다. 이를 위해 콘텐츠 모델을 구성하여 SVD++와 결합한다.

3.1 데이터 생성

MovieLens에서 제공하는 데이터는 영화ID, 사용자ID, 사용자가 본 영화에 부여한 점수 및 영화의 제작연도, 장르, 태그 정보가 포함되어 있다(10,681 Movies, 71,567 Users, 10,000,054 Ratings, 95,580 Tags)[11]. 여기에 영화에 대한 부가정보를 더 포함시키기 위해, 네이버 개발자 센터에서 제공하는 영화 API[12]와 다음 DNA에서 제공하는 영화 API[13]를 사용하였다.

데이터 생성을 위해 사용한 웹 API의 경우 한국에서 개봉된 영화의 정보를 우선으로 하기 때문에 검색이 되지 않는 경우가 많았다. 또한 영화 제목이 'Seven'과 같이 짧고, 동명의 영화가 많은 경우, 웹 API에서 연관어 검색을 수행한 후 API 출력 결과의 영문 제목을 이용해 똑같은 영화 제목일 경우 해당 영화라고 판단하였다. 또한 영화 제목이 국내에서 개봉될 때 바뀌는 경우나, 검색에서 누락될 경우에는 판단이 애매해지므로 부가정보를 포함시키지 않았다. 이렇게 생성한 최종 데이터는 원래의 MovieLens 10M 데이터(평점, 장르)에 감독, 배우에 대한 정보를 포함시켰다. 최종 데이터에 포함된 콘텐츠 데이터의 클래스와 레코드 수는 표 1과 같다.

표 1 콘텐츠 실험 데이터

콘텐츠 종류	클래스수	레코드수
Genre	20	21,564
Director	3,801	9,867
Actor	10,203	27,410

3.2 콘텐츠 정보를 이용한 SVD의 확장

콘텐츠 정보를 이용하여 어떤 사용자와 영화 간의 숨겨진 관계들을 찾아내기 위해 2장에서 설명한 기울기 강하 기법을 이용하여 학습하는 SVD를 확장한다. 콘텐츠 데이터는 모두 영화에 종속되므로, 영화와 콘텐츠 사이의 경향성을 이용하는 것보다 사용자와 콘텐츠 사이의 경향성을 파악하여, 이를 영화와 사용자 사이의 숨겨진 관계를 파악하는데 이용하였다. 우선 사용자-콘텐츠 경향성 파악을 위해 행렬 형태로 관계를 표현한다.

$$A_{contents} = \begin{cases} A_{contents}(x,y) = 1, \text{ if } movie\ x\ has\ contents\ y \\ A_{contents}(x,y) = 0, otherwise \end{cases} \tag{9}$$

이렇게 구성된 행렬을 SVD를 이용하여 각 사용자의 콘텐츠에 대한 경향성과 콘텐츠의 경향성을 표현하면 아래 식과 같다.

$$A_{contents} \cong \tilde{A}_{contents} = YH,\ (Y \in R^{m \times k}, H \in R^{k \times c}) \tag{10}$$

Y는 사용자의 특성행렬, H는 콘텐츠의 특성행렬이다. 하지만 영화의 점수 예측이 목표이므로 위 공식을 그대로 사용할 수는 없고, 사용자와 콘텐츠의 특징행렬을 이용하여 $R^{m \times n}$행렬의 결과가 나오도록 수정할 필요가 있다. 이를 위해 식 (10)을 수정한 아래 수식을 이용한다.

$$A \cong \tilde{A} = Y(|R(j)|^{-0.5} \sum_{c \in R(j)} H_c) \tag{11}$$

$R(j)$는 영화 j에 포함된 콘텐츠 집합을 뜻한다. 식 (11)은 영화 j에 포함된 콘텐츠 특성 벡터의 합을 영화 특성 벡터 대신 사용한다. 그러나 콘텐츠 정보가 누락되었을 경우, 위의 공식을 그대로 사용하면 학습이 불가능하다. 이를 처리하기 위해 식 (12)와 같이 기준선 추정 기법을 추가한다. 이로써 콘텐츠 정보를 얻지 못한 영화에 대해서도 다른 사용자의 경향성을 이용하여 학습이 가능하다.

$$a_{ij} \cong \tilde{a}_{ij} = b_{ij} + Y_i^T (|R(j)|^{-0.5} \sum_{c \in R(j)} H_c) \tag{12}$$

3.2.2 콘텐츠 모델 학습 단계

2.2.1절의 조정된 SVD 학습과 동일하게, 합-제곱 거리를 이용하여 식 (12)를 학습하기 위해 다음과 같이 표현할 수 있다. + 앞부분의 식은 총 오류의 합을 제곱한 것이고(실제 평점에서 예측 평점을 뺀 값), 뒷부분은 조정 부분(예측에 사용되는 변수들의 크기의 제곱의 합)이다.

$$\begin{aligned}
\min \sum_{(i,j) \in \Gamma} &\left(a_{ij} - b_{ij} - Y_i^T \left(|R(j)|^{-0.5} \sum_{c \in R(j)} H_c \right) \right)^2 \\
&+ \lambda \left(b_{ij}^2 + \| Y_i \|^2 + \| H_c \|^2 \right)
\end{aligned} \tag{13}$$

실험에 의하면 콘텐츠의 양에 비해 관찰된 관계의 수가 적기 때문에(0.001%) 단독 모델의 기대 성능은 낮다. 하지만 다른 방법과 함께 사용될 경우, 기존의 방법과는 다른 데이터의 양상을 함께 고려할 수 있다. 하지만 콘텐츠 모델은 한 아이템에 대해 해당되는 모든 콘텐츠 벡터에 대해 연산을 시도하므로, 기존의 사용자-아이템

관계만을 고려한 알고리즘에 비해 학습 시간이 길어진다는 단점이 있다.

3.3 콘텐츠 모델과 SVD++ 통합

콘텐츠 모델 식 (13)을 SVD++와 통합하면 식 (14)와 같다. 식에서 등호 오른쪽 두 번째 항은 SVD, 세 번째 항은 SVD++ 부분이다.

$$\widetilde{a_{ij}} = b_{ij} + Y_i^T(|R(j)|^{-0.5}\sum_{c\in R(j)} H_c) + M_j^T(S_i + |N(i)|^{-0.5}\sum_{j\in N(i)} W_j) \quad (14)$$

또한 이를 학습하기 위해 식 (15)과 같이 합-제곱 거리 형태로 나타낸다.

$$\min \sum_{(i,j)\in I} \begin{pmatrix} a_{ij} - b_{ij} - Y_i^T(|R(j)|^{-0.5}\sum_{c\in R(j)} H_c) \\ - M_j^T(S_i + |N(i)|^{-0.5}\sum_{j\in N(i)} W_j) \end{pmatrix} + \lambda\begin{pmatrix} b_{ij}^2 + \| Y_i \|^2 + \| H_c \|^2 \\ + \| M_j \|^2 + \| S_i \|^2 + \| W_j \|^2 \end{pmatrix} \quad (15)$$

식 (15)에 기울기 강하를 사용하되, W와 H는 다른 갱신 규칙을 먼저 수행한 후, 계산된 모든 오류 값들의 평균을 이용하여 사용자가 점수를 부여한 아이템에 대해서만 최종적으로 수정하여 계산 부담을 줄인다. 최종 알고리즘은 다음과 같다.

Integrated Model Algorithm

Init: $U, Y, S \in R^{m\times k},\ M, W, C \in R^{k\times n},\ H \in R^{k\times c}$
Repeat until convergence
  For each i
1. $U_i = S_i + |N(i)|^{-0.5}\sum_{j\in N(i)} W_j$
2. $C_j = |R(j)|^{-0.5}\sum_{c\in R(j)} H_c$
  For each j
3. $e_{ij} = a_{ij} - b_{ij} - \sum_{k=1}^{K} Y_{ik}C_{kj} - \sum_{k=1}^{K} U_{ik}M_{kj}$
4. $b_i \leftarrow b_i + \gamma_1(e_{ij} - \lambda_1 b_i)$
5. $b_j \leftarrow b_j + \gamma_1(e_{ij} - \lambda_1 b_j)$
6. $Y_i \leftarrow Y_i + \gamma_2(e_{ij}C_j - \lambda_2 Y_i)$
7. $S_i \leftarrow S_i + \gamma_2(e_{ij}M_j - \lambda_2 S_i)$
8. $M_j \leftarrow M_j + \gamma_2(e_{ij}U_i - \lambda_2 M_j)$
  End For
  For each $j \in N(i)$
9. $W_j \leftarrow W_j + \gamma_3(e_{ij}|N(i)|^{-0.5}M_j - \lambda_3 W_j)$
  End For
  For each $c \in R(j)$
10. $H_c \leftarrow H_c + \gamma_4(e_{ij}|R(j)|^{-0.5}Y_i - \lambda_4 H_c)$
  End For
 End For
End Repeat

4. 실험 및 결과

4.1 실험 환경

비교에 사용되는 알고리즘은 모두 Matlab에서 구현하였다. 모든 실험은 인텔 쿼드 Q6600 2.4GHz와 8GB RAM의 PC환경에서 수행되었다.

통합 모델에는 여러 매개변수가 있는데, 크게 3가지 종류로 분류할 수 있다. 첫 번째는 학습률(γ)으로, 기울기 강하 기법을 이용하여 한 단계에 특성 벡터의 값들을 수정할 정도를 가리킨다. 두 번째는 λ값으로 과적합 문제를 해결하기 위해 조절해 주어야 한다. 마지막으로 SVD 기법을 사용하는데 있어 숨겨진 요인 벡터의 차원 k를 결정하여야 한다. 이러한 매개변수들의 최적 값을 찾아내는 방법은 실험적인 방법뿐으로 매우 어렵다. 따라서 최적 값을 찾기 위해 반복 실험을 통해 값을 수정하였으며, 학습 속도에 상관없이 최소의 RMSE값을 찾는데 중점을 두었다. 실험적으로 결정된 값들은 각각 γ_1=0.0002, γ_2=γ_3=0.001, γ_4=0.003, λ_1=0.005, λ_2=0.002, λ_3=λ_4=0.001이다. 비교군인 SVD 및 SVD++ 알고리즘은 직접 구현하여 해당 논문에서 제시한 매개변수를 이용하여 실험하였다.

SVD의 차원 K는 너무 클 경우 과적합 문제를 야기하며, 너무 작을 경우에는 잠재 요인을 충분히 학습하지 못해 시스템이 제대로 된 성능을 발휘하기 어렵게 한다. 이에 적합한 차원 K 또한 실험적으로 결정하였다. 그림 1은 K의 값에 따른 통합 모델의 학습 결과이다. 차원을 높일수록 RMSE가 줄어들었으나 차원의 개수가 40을 넘어가면서 학습되는 시간에 비해 얻을 수 있는 성능의 향상이 미미했다. 이러한 점을 고려하여 K는 40으로 설정하였다.

4.2 실험 결과 및 분석

4.2.1 학습 시간 및 복잡도 분석

우선 공간 복잡도의 경우, SVD++은 $O(NK+MK)$개의 변수를 가진다. 통합 모델은 여기에 컨텐츠의 정보를 포함하기 때문에 $O(NK+MK+CK)$개의 변수를 가진다. 여기서 N은 영화의 개수, M은 사용자 수, C는 컨텐츠의 개수, K는 모델에서 사용된 차원의 개수이다. 특히 컨텐츠의 경우, 포함되는 속성에 따라 차원의 수가 기하급수적으로 늘어나는 문제가 있다. 때문에, 통합 모델을 사용하기 위해서는 적절한 개수의 속성을 선택할 필요성이 있다.

시간 복잡도 분석은 다음과 같다. R을 학습 데이터에 포함된 사용자-영화 쌍의 수라고 할 경우, 1회 학습의 시간 복잡도는 $O(R\times K)$이다. SVD++는 사용자가 점수를 부여하지 않은 모든 아이템들에 대해 업데이트를 시도하기 때문에 시간 복잡도는 $O(R\times N\times K)$이다. 통합 모델의 경우, 아이템에 포함된 콘텐츠의 특성 벡터를 업데이트하는 과정이 추가되므로, 시간 복잡도는 $O(R\times K\times (N+C))$와 같지만, 실제로 콘텐츠에서 관찰되는 관계의

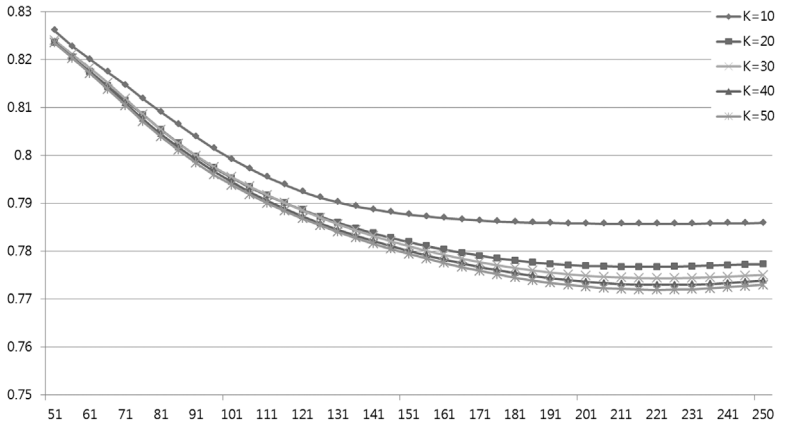

그림 2 학습 차원의 개수에 따른 RMSE 변화

표 2 알고리즘 1회 학습 시간

Algorithm	Learning Time (Sec)
SVD++ (K=30)	2.89
SVD++ (K=40)	3.15
SVD++ (K=50)	3.99
Integrated Model (K=30)	30.12
Integrated Model (K=40)	39.24
Integrated Model (K=50)	47.90

개수는 앞서 언급했듯이 매우 희박하다. 실제로 대부분의 아이템에서 10개 이하의 관계만이 관찰되므로, 콘텐츠의 크기에 따른 변화보다는 평균 콘텐츠 내에서 발견되는 관계들의 비율과 학습 데이터의 양에 따른 차이가 크다. 1회 학습에 걸린 시간은 표 2와 같이 측정되었다.

통합 모델의 학습 시간은 K가 40일 때 SVD++에 비해 10배 이상(3.15초 vs 39.24초) 증가하나, 계산이 오프라인으로 진행되어 행렬 분해가 완료되었다면 점수 예측은 실시간으로 이루어질 수 있으므로 여기서는 정확도의 향상에 더 큰 가중치를 부여한다.

4.2.2 성능 평가

성능 비교 평가를 위해 5-폴드 교차 검증(5-Fold Cross Validation)을 이용하였다. 5-폴드 교차 검증의 실험 결과로 나온 5개의 값의 평균을 이용하여 표 3에 기록하였으며, 학습 횟수에 따른 RMSE변화는 그림 2와 같다. 콘텐츠 정보는 통합 모델 및 콘텐츠 모델에서만 사용하였다.

표 3에서 BaseLine은 기준선 예측만을 사용한 결과로, 학습 데이터에서 단순 계산으로 사용자와 영화 바이어스(Bias)를 구하였다. 이 값을 그대로 점수 예측에 사용하였으며, 이는 알고리즘 최저 성능의 기준치이자 다른 알고리즘들의 초기 값으로 사용된다. 콘텐츠 모델은 BaseLine과 콘텐츠 정보만을 이용한 알고리즘으로 다른 알고리즘에 비해 성능이 많이 떨어지는 것을 확인할 수

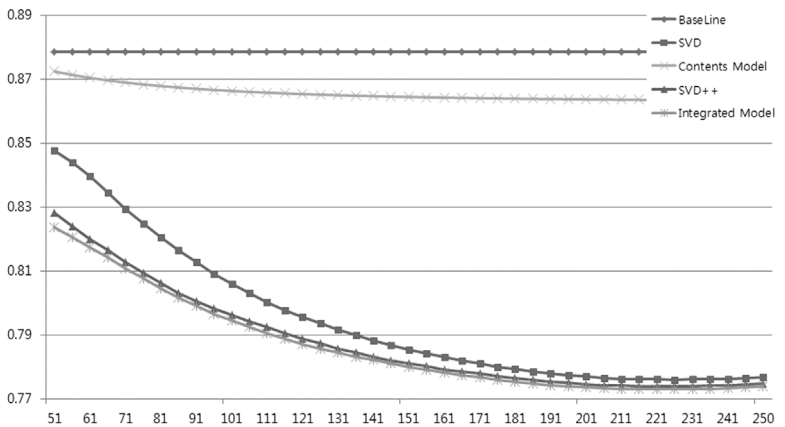

그림 3 알고리즘별 RMSE 변화

표 3 RMSE 성능 평가

Algorithm	RMSE	(표준편차)
BaseLine	0.8786	(0)
SVD	0.7760	(0.0007)
Contents Model	0.8643	(0.0010)
SVD++	0.7736	(0.0007)
Integrated Model (Proposed)	0.7729	(0.0006)

있다. 이 결과는 콘텐츠 정보에서 많은 부분이 오히려 노이즈로 작용한다는 것을 보여 준다.

그림 3에서 보이듯이, SVD와 SVD++ 및 통합 모델은 학습 횟수와 차원 개수를 늘릴수록 성능 차이가 줄어들지만, 콘텐츠 모델은 그렇지 않다. 이는 데이터의 희소성 및 노이즈 때문인데, 그렇지만 기존 방법과 함께 협력적 필터링에 적용함으로써 사용자와 영화 사이의 관계 예측에 도움을 주는 특성을 반영할 수 있다.

4.3 결론 및 향후 과제

본 연구에서는 영화 추천 시스템을 구성하는 문제에 대해서 콘텐츠 정보를 이용하여, 사용자의 경향성을 파악 할 수 있도록 사용자-콘텐츠의 행렬을 행렬 분해 알고리즘인 SVD를 통해 특성 행렬로 변환시켰으며, 이 때 간단한 기울기 강하 기법을 이용하여 비교적 빠르고, 쉽게 특성 행렬을 생성하였다. 그리고 이러한 특성 행렬을 사용자-영화 사이의 관계를 유추하기 위해 콘텐츠 특성 행렬의 부분 집합을 활용하는 방법을 이용하여, 사용자-영화 사이의 숨겨진 관계를 예측하는 알고리즘을 제안하였다. 마지막으로 SVD++ 알고리즘과 통합하는 과정을 통해 콘텐츠 정보를 비롯하여, 사용자-영화 사이의 관계를 통합적으로 고려할 수 있는 통합 모델을 제안하였다. MovieLens 데이터와 직접 생성한 데이터를 이용한 실험 결과, SVD++ 및 기타 알고리즘에 비해서 예측 결과가 우월하다는 것을 보였다.

본 논문의 연구 결과를 바탕으로 하여 더 좋은 성능의 영화 추천 알고리즘을 개발하기 위해 생각해 볼 수 있는 향후 과제는 크게 두 가지이다.

첫 번째로 데이터의 수집이다. 현재 웹 콘텐츠 사이트를 통해 생성한 콘텐츠 데이터는 전체 행렬에서 0.001%의 관계만이 관찰된다. 이렇게 비율이 작은 정보를 SVD를 통해 학습할 경우 반복된 학습을 통해 과적합 문제를 겪는다. 이를 위해서 엄선된 콘텐츠 데이터를 수집하고, 생성한 콘텐츠 데이터를 전처리를 통해 데이터의 희소성을 줄여줄 필요가 있다. 그리고 콘텐츠별 데이터를 늘리는 작업을 통해, 조금 더 다양한 방법을 시도해 볼 수 있다. 예를 들어 각 콘텐츠별 데이터를 따로 분류하여 각각의 모델을 이용하여 결측치를 예측하는 방법 등이 있다. MovieLens 데이터로 이러한 방법을 시도해 보았으나, 각 콘텐츠별 데이터의 수가 부족하여 학습이 잘 이루어지지 않았다. 무비렌즈 데이터와 함께 비슷한 형식으로 이루어진 Netflix 경진 대회[14]의 데이터를 함께 사용할 경우 이러한 문제를 해결할 수 있을 것이다.

두 번째로 계산량과 계산 시간이 콘텐츠 데이터의 양에 따라 기하급수적으로 증가하는 문제를 해결하여야 한다. 그러므로 학습 및 예측에 사용되는 콘텐츠의 수를 제한할 필요가 있다. 콘텐츠의 수를 제한하기 위해서 KNN 알고리즘 등을 이용하여 유사도를 비교하는 방법이 있는데, 이 부분에 대해서는 앞으로 연구가 필요하다.

참 고 문 헌

[1] G. Adomavicius and A. Tuzhilin, "Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," *IEEE Transactions on Knowledge and Data Engineering*, no.17, pp.634-749, 2005.

[2] R. Bell and Y. Koren, "Lessons from the Netflix Prize Challenge," *SIGKDD Explorations*, vol.9, no.2, 2007.

[3] P. Melville, R. M. Monney and R. Nagarajan, "Content-boosted Collaborative Filtering for Improved Recommendations," *Proceedings of 18th National Conference on Artificial Intelligence*, 2002.

[4] J. Salter and N. Antonopoulos, "CinemaScreen Recommender Agent: Combining Collaborative Filtering and Content-Based Filtering," *IEEE Intelligent Systems*, vol.21, no.1, pp.35-41, 2006.

[5] Y. Koren, R. Bell and C. Volinssky, "Matrix Factorization Techniques for Recommender System," *IEEE Computer*, vol.42, no.8, pp.30-37, 2009.

[6] S. Funk, "Netflix Update: Try This At home," http://sifter.org/~simon/journal/20061211.html, 2006.

[7] Gabor Takacs, Istvan Pilaszy, Bottyan Nemeth and Domonkos Tikk, "On the Gravity Recommendation System," *Proceedings of KDD Cup and Workshop*, 2007.

[8] R. Salakhutdinov and A. Mnih, "Probabilistic Matrix Factorization," *Advances in Neural Information Processing*, 2008.

[9] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," *Proceedings of KDD Cup and Workshop*, 2007.

[10] Y. Koren, "Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model," *Conference on Knowledge Discovery and Data Mining*, 2008.

[11] MovieLens, "www.movielens.org".

[12] "http://dev.naver.com/openapi/apis/data/movie".

[13] "http://dna.daum.net/apis/contents/movie".

[14] Netflix Prize, "www.netflixprize.com".

김 상 화

2009년 서강대학교 컴퓨터공학과 졸업(학사). 2011년 서강대학교 컴퓨터공학과 졸업(석사). 2011년~현재 다음커뮤니케이션 FT기술팀 팀원. 관심분야는 기계학습, 추천 시스템, 음성 인식 등

오 병 화

2007년 서강대학교 컴퓨터학과 졸업(학사). 2009년 서강대학교 컴퓨터공학과 졸업(석사). 2009년~현재 서강대학교 컴퓨터공학과 재학(박사). 관심분야는 기계학습, 추천 시스템, 진화 알고리즘, 네트워크 과학, 베이지안 추론 등

김 문 종

2009년 성결대학교 정보통신공학부 졸업(학사). 2011년~현재 서강대학교 컴퓨터공학과 재학(석사). 관심분야는 기계학습 강화 학습, 무인 자동차 등

양 지 훈

1987년 서강대학교 전자계산학과 졸업(학사). 1989년 ISU Department of Computer Science(석사). 1999년 ISU Department of Computer Science(박사). 1999년~2000년 HRL Laboratories, LLC Malibu, CA 연구원. 2000년~2002년 SRA International, Inco, Fairfax, VA 연구원. 2002년~현재 서강대학교 컴퓨터공학과 교수. 관심분야는 기계학습, 바이오인포매틱스 등