The Korean Journal of Applied Statistics
2021, Vol. 34, No. 3, 329–345

DOI: https://doi.org/10.5351/KJAS.2021.34.3.329
Print ISSN1225-066X / Online ISSN 2383-5818

Comparison of deep learning-based autoencoders for recommender systems

Hyo Jin Lee[a], Yoonsuh Jung[1,a]

[a]Department of Statistics, Korea University

Abstract

Recommender systems use data from customers to suggest personalized products. The recommender systems can be categorized into three cases; collaborative filtering, contents-based filtering, and hybrid recommender system that combines the first two filtering methods. In this work, we introduce and compare deep learning-based recommender system using autoencoder. Autoencoder is an unsupervised deep learning that can effective solve the problem of sparsity in the data matrix. Five versions of autoencoder-based deep learning models are compared via three real data sets. The first three methods are collaborative filtering and the others are hybrid methods. The data sets are composed of customers' ratings having integer values from one to five. The three data sets are sparse data matrix with many zeroes due to non-responses.

Keywords: autoencoder, deep learning, recommender system, sparse data matrix

1. 서론

인터넷과 휴대폰 앱과 같이 웹(web)기반의 플랫폼 사용이 활발해짐에 따라 온라인상에 방대한 양의 정보가 존재한다. 기업은 이 중에서 의미 있는 정보만을 이용하여 고객 만족 및 매출 증진을 목표로 재화와 서비스 제공을 위한 의사결정을 수행하고자 한다. 그러기 위해서는 고객의 니즈를 명확히 파악하고 그에 맞는 상품 및 서비스를 제공함으로써 고객의 구매 욕구를 끌어올리는 것이 중요하다. 따라서 일률적인 마케팅이 아닌, 각각의 고객의 특성을 반영한 마케팅 전략의 수립이 필요하며 이는 최근 추천 시스템(recommender system)이 주목을 받는 이유이다. 용어의 일반화를 위하여 고객을 사용자(user)로, 재화 및 서비스를 아이템(item)으로 지칭한다. 추천 시스템은 사용자의 선호도, 아이템의 특징 그리고 사용자와 아이템 상호 간의 교호작용 등의 정보를 이용함으로써 개인 맞춤화(personalization)전략을 실현하고자 한다. 추천 시스템은 콘텐츠 기반 필터링(content-based filtering), 협업 필터링(collaborative filtering) 그리고 하이브리드 방법(hybrid recommender system)으로 구성된다 (Zhang 등, 2019).

첫 번째로 콘텐츠 기반 필터링 방법은 비슷한 특성을 가진 아이템이라면 사용자로부터 유사한 평가를 받을 것이라고 가정한다. 따라서 사용자로부터 긍정적인 평가를 받았거나 구매 이력이 있는 아이템과 유사한 것을 추천한다 (Schafer 등, 2007). 아이템 간의 유사성은 사용자 혹은 아이템의 고유 정보(profile)를 이용하여 유클리디안 거리, 코사인 거리, 피어슨 상관계수 등을 계산하여 파악한다. 아이템의 고유 정보로는 해당

Jung's work has been partially supported by National Research Foundation of Korea (NRF) grants funded by the Korean government (MIST) 2019R1F1A1040515 and 2019R1A4A1028134.

[1] Corresponding Author: Department of Statistics, Korea University, 145 Anam-ro Seongbuk-Gu, Seoul 02841, Korea. E-mail: yoons77@korea.ac.kr

Published 30 June 2021 / journal homepage: http://kjas.or.kr

아이템이 속한 범주, 기능, 가격, 지명도 등을 예로 들 수 있다. 사용자의 고유 정보로는 연령, 성별, 거주지 등 인구 통계학적 정보를 예로 들 수 있다 (Koren 등, 2009). 콘텐츠 기반 필터링 방법의 장점은 콜드 스타트 (cold start)문제의 발생을 어느 정도 피할 수 있다. 콜드 스타트 문제란 새로운 사용자의 경우 평가와 구매 이력이 존재하지 않으며 새로운 아이템의 경우 사용자로부터 받은 평가나 구매 이력이 없는 상황을 의미한다. 콘텐츠 기반 필터링 방법은 사용자 혹은 아이템의 고유 정보에 의존하기 때문에 새로운 사용자나 아이템에 대해서도 추천이 가능하다 (Ali 등, 2018). 그러나 개인의 사생활 침해 우려로 인해 사용자 고유 정보의 이용이 제한적이고, 수많은 아이템에 대해 각각의 고유 정보 파악이 어렵다는 한계가 있다.

두 번째로 협업 필터링 방법은 비슷한 취향을 가진 사용자라면 비슷한 아이템을 선호할 것이라고 가정한다. 따라서 사용자의 평가 성향과 비슷한 다른 사용자로부터 긍정적인 평가를 받은 아이템을 추천한다. 이는 콘텐츠 기반 필터링 방법과는 달리 도메인 지식이 불필요하기 때문에 데이터를 광범위하게 많이 수집할 필요가 없다는 장점이 있어 Amazon과 Netflix를 비롯한 많은 기업에서 사용하고 있다 (Koren, 2008). 협업 필터링 방법에서 이용하는 정보는 크게 명시적 피드백(explicit feedback)과 내재적 피드백(implicit feedback)으로 나뉜다. 명시적 피드백이란 사용자가 아이템에 대한 선호도를 직접적으로 표현한 것을 의미하며 대표적으로 평점이 이에 속한다. 반면, 내재적 피드백이란 사용자의 선호도를 간접적으로 표현한 것으로 구매 이력, 검색 기록 및 패턴, 심지어는 컴퓨터 화면상 마우스의 움직임 등이 이에 속한다. 협업 필터링 방법은 이 두 가지의 피드백 정보를 사용하며 인구 통계학적 정보를 사용하지 않아도 되기 때문에 개인 정보가 침해될 위험이 적다. 그러나 콜드 스타트 문제를 동반한다는 점과 각 사용자가 실제 평점을 매긴 아이템이 몇몇 개에 불과하여 평점 행렬에서 평점이 없는 부분이 많은 비율을 차지하는 문제 즉, 높은 희박성(sparsity)을 갖는다는 한계가 있다 (Schafer 등, 2007).

세 번째로 하이브리드 방법은 콘텐츠 기반 필터링 방법과 협업 필터링 방법을 결합함으로써 각각의 장점을 취하는 방법이다. 예를 들면, 협업 필터링 방법을 적용할 때 아이템에 대한 고유 정보를 이용하여 콘텐츠 기반 필터링 방법을 결합하는 접근 방법이 이에 속한다. 그러므로 하이브리드 방법을 이용하여 콜드 스타트 문제와 높은 희박성 문제를 해결할 수 있다.

본 논문의 구성은 다음과 같다. 딥러닝을 이용한 추천 시스템 모형에 대해서는 2장에서 다루며 특히 오토인코더 기반의 다섯 가지 방법에 대하여 소개한다. 그 중 2.1장부터 2.3장까지는 협업 필터링 방법을 이용한 모형이며, 2.4장과 2.5장는 하이브리드 방법을 이용한 모형에 해당한다. 3장에서는 실제 데이터를 소개하고 여기에 소개한 방법을 적용한 결과에 대하여 살펴본다. 마지막으로 4장에서는 본 논문이 시사하는 바에 대하여 다루도록 한다.

2. 오토인코더 기반 방법들

2.1. 오토인코더

기계학습의 종류는 크게 지도학습(supervised learning)과 비지도학습(unsupervised learning)으로 나뉜다. 지도학습이란 문제와 그에 대한 정답을 훈련 데이터를 이용하여 학습시킨 후 문제에 해당하는 부분만 있는 시험 데이터의 정답을 맞히도록 하는 방법이다. 비지도학습이란 정답에 대한 정보 없이 여러 문제를 학습함으로써 훈련 데이터의 패턴 및 특성을 파악하여 시험 데이터의 특징을 찾는 방법이다. 실제 자료에서 문제와 그에 대한 정답이 항상 동반되기는 어렵다는 점에서 비지도학습 방법이 유용하게 사용될 수 있다. 오토인코더는 비지도 학습 모형 중 하나로, 차원 축소를 통해 원래 자료를 가장 잘 대표하는 핵심 요소(feature)를 추출하는 것이 목적이다. 선형 차원 축소 방법에 해당하는 주성분 분석과 다르게 오토인코더는 비선형 차원 축소 방법으로, 고차원 자료의 분석에 유용하게 적용된다.

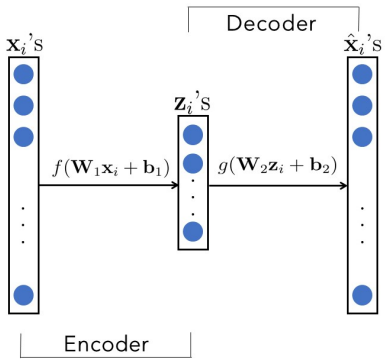

Figure 1: *The structure of autoencoder consisting of encoder and decoder considering a non-linear relationship as an activation function.*

Figure 1은 기본적인 오토인코더의 구조를 나타낸다. 입력층의 차원 축소를 통해 핵심 요소만 추출하는 부분을 인코더(encoder)라고 하며 여기서 추출된 핵심 요소들의 집합이 있는 층을 은닉층(hidden layer)이라고 한다. 추출된 핵심 요소를 이용하여 입력값과 동일한 차원의 출력값을 산출하는 부분을 디코더(decoder)라고 한다. 인코더는 $\mathbf{x}_i \in \mathbb{R}^M$을 입력값으로 받아 활성화 함수(activation function)인 $f(\cdot)$ 를 거쳐 차원을 축소시킨 출력값을 산출한다. 이렇듯 인코더를 통해 추출된 핵심 요소인 잠재 변수(latent variable)를 $\mathbf{z}_i \in \mathbb{R}^k$라고 지칭한다. 이때 $\mathbf{z}_i$는 $k \ll M$이라는 가정하에 차원이 축소된 형태이다. 디코더 함수는 입력값으로서 $\mathbf{z}_i$를 받으며 활성화 함수 $g(\cdot)$를 거쳐 원래의 입력값인 $\mathbf{x}_i$의 차원으로 복구하는 작업을 수행한다. 이러한 과정을 각각 N개의 $\mathbf{x}_i$벡터에 적용한다.

오토인코더의 인코더와 디코더는 대칭을 이룬다. 식 (2.1)은 인코더 부분을, 식 (2.2)는 디코더 부분을 표현한 것이다 (Wu 등, 2016)

$$\mathbf{z}_i = f(\mathbf{W}_1\mathbf{x}_i + \mathbf{b}_1), \quad i = 1, \ldots, N, \tag{2.1}$$

$$\hat{\mathbf{x}}_i = g(\mathbf{W}_2\mathbf{z}_i + \mathbf{b}_2) = g(\mathbf{W}_2 f(\mathbf{W}_1\mathbf{x}_i + \mathbf{b}_1) + \mathbf{b}_2), \quad i = 1, \ldots, N. \tag{2.2}$$

여기서 입력값 $\mathbf{x}_i \in \mathbb{R}^M$에 대하여 가중치 행렬(weight matrix)은 $\mathbf{W}_1 \in \mathbb{R}^{k\times M}$, $\mathbf{W}_2 \in \mathbb{R}^{M\times k}$로, 편향 벡터는 $\mathbf{b}_1 \in \mathbb{R}^k$, $\mathbf{b}_2 \in \mathbb{R}^M$으로 정의하며 $k \ll M$이라는 가정하에 잠재 변수 $\mathbf{z}_i \in \mathbb{R}^k$는 차원이 축소된 형태이다. 디코더를 거쳐 재구성된 $\mathbf{x}_i$를 $\hat{\mathbf{x}}_i$이라고 하면 차원이 축소된 $\mathbf{z}_i$는 디코더에 의해 $\hat{\mathbf{x}}_i \in \mathbb{R}^M$으로 $\mathbf{x}_i$와 동일한 차원을 출력한다. 인코더와 디코더에서 적용하는 활성화 함수 $f(\cdot)$와 $g(\cdot)$로는 비선형 관계식에 해당하는 tangent hyperbolic function (Tanh), rectified linear unit (ReLU), exponential linear unit (ELU), leaky rectified linear unit (LReLU), scaled exponential linear unit (SeLU) 등을 사용한다 (Kuchaiev와 Ginsburg, 2017).

오토인코더는 디코더를 통해 산출한 출력값이 최대한 입력값과 비슷하도록 학습하는 것을 목표로 한다. 따라서 입력값과 출력값의 차이를 계산하는 손실함수를 최소화하는 것이 오토인코더의 목적함수가 되며, 손실함수로서 제곱근 평균 제곱 오차(root mean square error; RMSE)를 정의할 경우 오토인코더의 목적함수는 식 (2.3)으로 표현할 수 있다

$$\underset{\mathbf{W}_1,\mathbf{W}_2,\mathbf{b}_1,\mathbf{b}_2}{\operatorname{argmin}} L\left(\mathbf{X}, \hat{\mathbf{X}}\right) = \underset{\mathbf{W}_1,\mathbf{W}_2,\mathbf{b}_1,\mathbf{b}_2}{\operatorname{argmin}} \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M} \frac{\left(x_{ij} - \hat{x}_{ij}\right)^2}{NM}}. \tag{2.3}$$

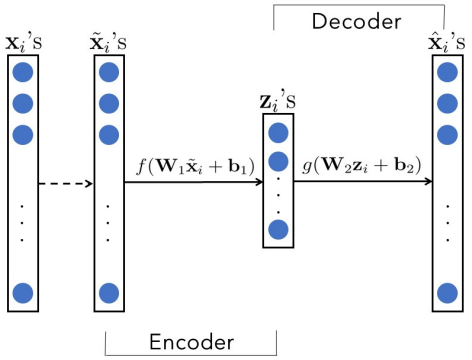

Figure 2: *The structure of denoising autoencoder containing input value with added noise* $\tilde{\mathbf{x}}$*, which is distinct from the basic autoencoder.*

이때 $\hat{\mathbf{X}} \in \mathbb{R}^{N\times M}$이고, i번째 행은 $\hat{\mathbf{x}}_i = \{\hat{x}_{i1}, \ldots, \hat{x}_{iM}\}^T$로 정의한다. 그리고 $\hat{\mathbf{x}}_i = g(\mathbf{W}_2 f(\mathbf{W}_1\mathbf{x}_i + \mathbf{b}_1) + \mathbf{b}_2)$로 계산한다.

2.2. 잡음 제거 오토인코더

기본적인 오토인코더가 입력값과 최대한 비슷한 출력값을 산출하도록 학습하는 것이 목표인 반면, 잡음 제거 오토인코더(denoising autoencoder)는 잡음이 추가된 입력값을 최대한 잡음이 추가되기 전의 입력값으로 재구성하도록 학습하는 것을 목표로 하는 모형이다 (Vincent 등, 2008). 잡음 제거 오토인코더를 제안한 논문에 따르면 이 모형의 발상 원천은 'robustness to partial destruction of the input'이라고 설명한다 (Vincent 등, 2008). 즉, 입력값에 일부 누락된 부분이 있더라도 핵심적인 요소를 파악한다면 입력값을 복원해 내는 효과를 기대할 수 있는 것이다. 따라서 잡음이 추가된 입력값으로부터 잡음이 제거된 값을 출력하도록 학습시키기 때문에 기본 오토인코더에 비해 입력값의 대표성이 더욱 높은 요소를 학습한다. Figure 2는 잡음 제거 오토인코더의 구조이다. 전체 평점 행렬(rating matrix) $\mathbf{X} \in \mathbb{R}^{\mathrm{M}\times\mathrm{N}}$에서 잡음이 추가된 i번째 행은 $\tilde{\mathbf{x}}_i = (\tilde{x}_{i1}, \ldots, \tilde{x}_{iM})^T$로 정의한다. 기본적인 오토인코더와 다르게 입력값으로서 $\tilde{\mathbf{x}}_i \in \mathbb{R}^{\mathrm{M}}$이 입력되는 것을 확인할 수 있다. 따라서 인코더와 디코더의 활성화 함수에서 입력값은 $\tilde{\mathbf{x}}_i$가 되어 식 (2.4)로 정의된다.

$$\begin{aligned} \mathbf{z}_i &= f(\mathbf{W}_1\tilde{\mathbf{x}}_i + \mathbf{b}_1), \quad i = 1, \ldots, N, \\ \hat{\mathbf{x}}_i &= g(\mathbf{W}_2\mathbf{z}_i + \mathbf{b}_2) = g(\mathbf{W}_2 f(\mathbf{W}_1\tilde{\mathbf{x}}_i + \mathbf{b}_1) + \mathbf{b}_2), \quad i = 1, \ldots, N. \end{aligned} \tag{2.4}$$

Figure 3은 매니폴드 학습(manifold learning)관점에서 잡음 제거 오토인코더의 개념을 나타낸 것이다. 매니폴드 학습이란 고차원 공간상의 데이터를 저차원의 공간상에 생성해내는 것을 의미한다 (Zheng과 Xue, 2009). 매니폴드 학습을 통해 데이터를 대표하는 요소를 저차원의 공간상으로 매핑(mapping)시킴으로써 고차원의 데이터가 갖는 내재적 구조를 찾을 수 있다. 매니폴드 학습 관점에서 보면 Figure 3에서 실선이 원래의 입력값 $\mathbf{x}_i$를 곡선 형태로 연결한 것일 때, 점선으로 표현된 것처럼 잡음이 추가된 입력값인 $\tilde{\mathbf{x}}_i$를 $\mathbf{x}_i$로 최대한 매핑할 수 있도록 매니폴드 학습시키는 과정으로써 잡음 제거 오토인코더를 이해할 수 있다. 여기서 $q_{\mathcal{D}}(\tilde{\mathbf{x}}_i|\mathbf{x}_i)$는 잡음이 추가된 입력값 $\tilde{\mathbf{x}}_i$의 의미로써 $\mathbf{x}_i$를 손상시킨 값이라고 이해하였을 때 $\tilde{\mathbf{x}}_i \sim q_{\mathcal{D}}(\tilde{\mathbf{x}}_i|\mathbf{x}_i)$라고 정의한 것이다. 또한, $g_{\theta'}(f_\theta(\tilde{\mathbf{x}}_i))$에서 $\theta = \{\mathbf{W}_1, \mathbf{b}_1\}$을, $\theta' = \{\mathbf{W}_2, \mathbf{b}_2\}$를 의미한다 (Vincent 등, 2008). 일반적으로 기존의 입력값에 추가할 잡음으로서 가우시안 잡음(Gaussian noise) 또는 마스크 아웃/드롭 아웃 잡음(mask-out/drop-

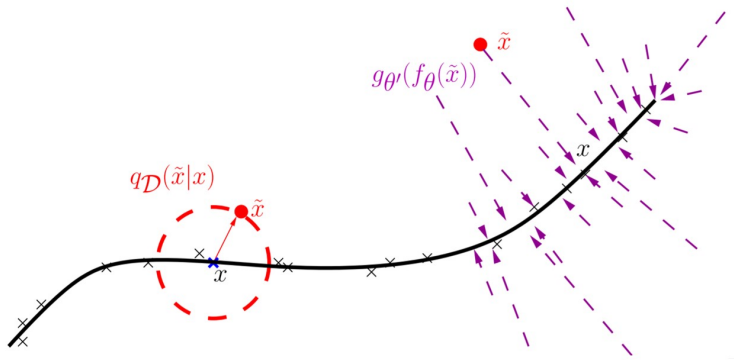

Figure 3: *Understanding denoising autoencoder from manifold learning perspective (Vincent et al. 2008).* $\tilde{\mathbf{x}}_i$ *is generated from* $q_{\mathcal{D}}(\tilde{\mathbf{x}}_i|\mathbf{x}_i)$ *and the denoising autoencoder tries to closely map* $g_{\theta'}(f_\theta(\tilde{\mathbf{x}}_i))$ *to real input value* $\mathbf{x}_i$.

out noise)이 사용된다 (Wu 등, 2016). 특히 마스크 아웃/드롭 아웃 잡음의 경우 본 논문에서는 사용자가 실제 평가를 한 점수 중 일부를 임의로 선택하여 0점으로 가림으로써 잡음의 역할을 수행하도록 하였다. 잡음 제거 오토인코더는 식 (2.1)에서 $\mathbf{x}_i$ 대신 잡음이 추가된 $\tilde{\mathbf{x}}_i$를 대입하여 모형을 학습시키며 기본적인 오토인코더와 마찬가지로 제곱근 평균 제곱 오차를 손실함수로 가정할 경우의 목적함수는 식 (2.3)로 표현할 수 있다. 단, 오토인코더에서는 $\hat{\mathbf{x}}_i = g(\mathbf{W}_2 f(\mathbf{W}_1\mathbf{x}_i + \mathbf{b}_1) + \mathbf{b}_2)$라고 정의하는 것과 다르게 잡음 제거 오토인코더에서는 $\hat{\mathbf{x}}_i = g(\mathbf{W}_2 f(\mathbf{W}_1\tilde{\mathbf{x}}_i + \mathbf{b}_1) + \mathbf{b}_2)$로 정의한다

$$\underset{\mathbf{W}_1,\mathbf{W}_2,\mathbf{b}_1,\mathbf{b}_2}{\operatorname{argmin}} L\left(\mathbf{X}, \hat{\mathbf{X}}\right) = \underset{\mathbf{W}_1,\mathbf{W}_2,\mathbf{b}_1,\mathbf{b}_2}{\operatorname{argmin}} \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M}\frac{\left(x_{ij} - \hat{x}_{ij}\right)^2}{NM}}. \tag{2.5}$$

이때 $\hat{\mathbf{X}}$의 i번째 행은 $\hat{\mathbf{x}}_i = g(\mathbf{W}_2 f(\mathbf{W}_1\tilde{\mathbf{x}}_i + \mathbf{b}_1) + \mathbf{b}_2)$로 계산한다.

2.3. 변분 오토인코더

오토인코더는 주어진 데이터로부터 차원 축소를 통하여 핵심 요소를 추출하는 것이 목적인 반면, 변분 오토인코더(variational autoencoder; VAE)는 추출된 핵심 요소의 분포를 찾아내고 이 분포로부터 새로운 자료를 생성하는 것이 목적이다. 즉, 오토인코더는 입력과 출력이 유사하도록 학습시키는 모형이고 변분 오토인코더는 입력의 분포를 따르는 새로운 출력을 생성하는 생성 모형(generative model)이다. 이렇듯 변분 오토인코더가 일반적인 오토인코더와 목적상 차이가 있음에도 불구하고 '오토인코더'라는 명칭을 사용하는 이유는 인코더와 디코더로 이루어져 있고 상호 대칭형 구조를 이룬다는 점이 유사하기 때문이다.

변분 오토인코더의 발상 원천은 현실에서 접하는 데이터의 경우 그 분포를 다루는 것이 어렵고 계산에 시간이 많이 소요된다는 점이다 (Kingma와 Welling, 2013). 예를 들어 θ를 생성 모형 모수(generative model parameter)라 하고, ϕ를 변분 모수(variational parameter)라고 할 때, $p_\theta(\mathbf{x}_i) = \int p_\theta(\mathbf{z}_i)p_\theta(\mathbf{x}_i|\mathbf{z}_i)d\mathbf{z}_i$를 계산하기 위해서는 $p_\theta(\mathbf{x}_i|\mathbf{z}_i)$를 알아야 하는데 현실적으로 불가능하며 알고 있더라도 그것의 적분을 계산하기까지 오랜 시간이 소요된다. 또한, 조건부 사후 분포인 $p_\theta(\mathbf{z}_i|\mathbf{x}_i) = p_\theta(\mathbf{x}_i|\mathbf{z}_i)p_\theta(\mathbf{z}_i)/p_\theta(\mathbf{x}_i)$의 경우도 동일한 어려움이 있다. 이를 해결하기 위하여 $p_\theta(\mathbf{z}_i|\mathbf{x}_i)$를 대신하여 다루기 쉬운 분포로서 $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$라고 가정하여 분포를 구성하는 모수들을 조정함으로써 실제 확률분포와 유사하도록 근사시킨다. 이를 변분 추론(variational inference)이라고 한다. 주로 $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$를 가우시안 확률분포로 가정하며 입력값 $\mathbf{x}_i$로부터 잠재 변수 $\mathbf{z}_i$를 추출해 내는 부분을 의미

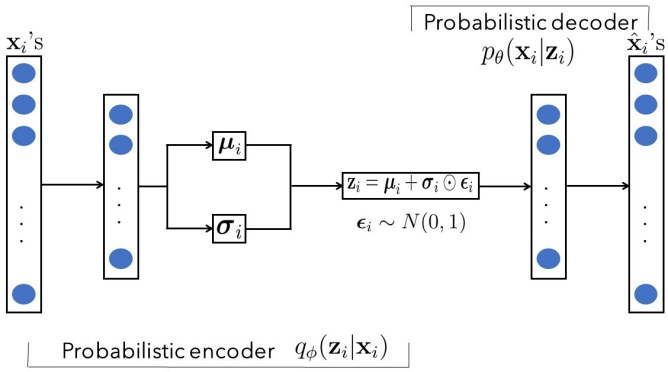

Figure 4: *The structure of variational autoencoder.*

하므로 확률적 인코더(probabilistic encoder)라고 한다. 마찬가지로, 추출된 $\mathbf{z}_i$로부터 $\mathbf{x}_i$를 생성해내는 부분인 $p_\theta(\mathbf{x}_i|\mathbf{z}_i)$를 확률적 디코더(probabilistic decoder)라고 부른다.

Figure 4는 변분 오토인코더의 구조를 나타낸 것이다. 여기서 가우시안 분포를 가정한 인코더 부분인 $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$로부터 직접적으로 $\mathbf{z}_i$를 추출하는 것이 아니라 $\mathbf{z}_i$의 분포를 알기 위해 샘플링하여 $\mathbf{z}_i$의 평균 $\boldsymbol{\mu}_i \in \mathbb{R}^k$와 표준편차 $\boldsymbol{\sigma}_i \in \mathbb{R}^k$를 출력한다. 이때 역전파 알고리즘 과정 중에 미분이 불가능한 점이 발생하기 때문에 이를 방지하기 위하여 $\mathbf{z}_i$를 대신하여 $\boldsymbol{\mu}_i + \boldsymbol{\sigma}_i \odot \boldsymbol{\epsilon}_i$를 디코더의 입력값으로 입력하며 이러한 방법을 reparameterization trick이라고 한다. 여기서 $\odot$은 요소별 곱셈 연산(element-wise product)을 수행하는 것을 의미하며 $\boldsymbol{\epsilon}_i \in \mathbb{R}^k$의 각각의 원소는 평균이 0, 표준편차가 1인 정규분포를 따른다고 가정한다. 변분 오토인코더의 목적은 $p_\theta(\mathbf{z}_i)$와 $q_\phi(\mathbf{z}_i|\mathbf{x}_i)$를 유사하게 근사시키는 것이기 때문에 두 분포의 유사성을 측정하는 척도인 쿨백-라이블러 발산(Kullback-Leibler divergence; KLD)을 최소화하고자 한다. 이는 입력값인 $\mathbf{x}_i$의 분포를 최대한 파악하고자 확률분포함수에 로그를 취한 $\log p_\theta(\mathbf{x}_i)$를 최대화하는 것으로 생각할 수 있다. 이러한 변분 오토인코더의 목적함수를 유도하는 과정은 다음과 같다. 식 (2.6)은 베이즈 규칙(Bayes rule)을 이용하여 $\log p_\theta(\mathbf{x}_i)$를 표현한 것이다

$$\begin{aligned}\log p_\theta(\mathbf{x}_i) &= \log \frac{p_\theta(\mathbf{x}_i|\mathbf{z}_i)p_\theta(\mathbf{z}_i)}{p_\theta(\mathbf{z}_i|\mathbf{x}_i)} \\ &= \log p_\theta(\mathbf{x}_i|\mathbf{z}_i) + \log p_\theta(\mathbf{z}_i) - \log p_\theta(\mathbf{z}_i|\mathbf{x}_i). \end{aligned} \tag{2.6}$$

또한, $\log p_\theta(\mathbf{x}_i)$는 확률밀도함수의 적분값은 1이라는 성질인 $\int q_\phi(\mathbf{z}_i|\mathbf{x}_i)d\mathbf{z}_i = 1$임을 이용하면 식 (2.7)로 표현할 수 있다

$$\log p_\theta(\mathbf{x}_i) = \int q_\phi(\mathbf{z}_i|\mathbf{x}_i)\ \log p_\theta(\mathbf{x}_i)d\mathbf{z}_i. \tag{2.7}$$

이제 식 (2.7)에서 우변의 $\log p_\theta(\mathbf{x}_i)$를 대신하여 식 (2.6)을 대입하면 식 (2.8)이 유도된다.

$$\begin{aligned}\log p_\theta(\mathbf{x}_i) &= \int q_\phi(\mathbf{z}_i|\mathbf{x}_i)\Big[\log p_\theta(\mathbf{x}_i|\mathbf{z}_i) + \log p_\theta(\mathbf{z}_i) - \log p_\theta(\mathbf{z}_i|\mathbf{x}_i)\Big]d\mathbf{z}_i \\ &= E_{q_\phi}\left(\log p_\theta(\mathbf{x}_i|\mathbf{z}_i)\right) - \int q_\phi(\mathbf{z}_i|\mathbf{x}_i)\ \log \frac{q_\phi(\mathbf{z}_i|\mathbf{x}_i)}{p_\theta(\mathbf{z}_i)}d\mathbf{z}_i + \int q_\phi(\mathbf{z}_i|\mathbf{x}_i)\ \log \frac{q_\phi(\mathbf{z}_i|\mathbf{x}_i)}{p_\theta(\mathbf{z}_i|\mathbf{x}_i)}d\mathbf{z}_i. \end{aligned} \tag{2.8}$$

쿨백-라이블러 발산의 정의에 따르면 $D_{\mathrm{KL}}(q_\phi(\mathbf{z}_i|\mathbf{x}_i)\|p_\theta(\mathbf{z}_i)) = E_{q_\phi}(\log q_\phi(\mathbf{z}_i|\mathbf{x}_i)/p_\theta(\mathbf{z}_i))$이다. 또한 쿨백-라이블러 발산은 항상 0 이상의 값을 갖는다는 성질에 따라 식 (2.9)의 부등식과 같이 표현할 수 있고 이를 evidence

lower bound (ELBO)라고 한다 (Kingma와 Welling, 2013).

$$\begin{aligned}\log p_\theta(\mathbf{x}_i) &= E_{q_\phi}\left(\log p_\theta(\mathbf{x}_i|\mathbf{z}_i)\right) - D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}_i|\mathbf{x}_i)\|p_\theta(\mathbf{z}_i)\right) + D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}_i|\mathbf{x}_i)\|p_\theta(\mathbf{z}_i|\mathbf{x}_i)\right)\\ &\geq E_{q_\phi}\left(\log p_\theta(\mathbf{x}_i|\mathbf{z}_i)\right) - D_{\mathrm{KL}}\left(q_\phi(\mathbf{z}_i|\mathbf{x}_i)\|p_\theta(\mathbf{z}_i)\right).\end{aligned} \tag{2.9}$$

따라서 변분 오토인코더의 목적인 쿨백-라이블러 발산을 최소화하는 문제는 결국 $\log p_\theta(\mathbf{x}_i)$를 최대화하는 문제로 바뀐다. 부등식을 기준으로 우변의 첫 번째 항은 디코더에 의해 재구성된 출력값의 우도(likelihood)의 측도를 나타내는 것이기 때문에 재구성 항(reconstruction term)이라고 한다. 또한, 부등식을 기준으로 우변의 두 번째 항인 쿨백-라이블러 발산항에 해당하는 부분은 근사시킨 사후 확률 분포에 규제를 주는 효과가 있기 때문에 정칙자(regularizer)라고 한다 (Odaibo, 2019).

2.4. AutoSVD

하이브리드 방법은 콘텐츠 기반 필터링 방법과 협업 필터링 방법을 결합시킨 것이다. 협업 필터링 방법은 사용자의 선호를 파악하기에 효과적이지만 사용자와 아이템의 평점 행렬이 희박 행렬이라는 단점이 있다. 이를 보완하기 위하여 아이템에 대한 정보를 결합함으로써 추천 시스템의 성능을 향상시키고자 한다 (Zhang 등, 2017). 그 중 본 논문에서 비교할 하이브리드 방법은 AutoSVD 방법과 AutoSVD++ 방법이다. 두 방법은 공통적으로 축약 오토인코더(contractive autoencoder)와 SVD 방법을 결합하여 아이템의 보조 정보(side information)를 활용한다.

축약 오토인코더는 잡음이 추가된 입력값을 이용하여 입력값으로부터 더욱 대표성이 높은 핵심 요소를 추출할 것을 기대한다는 점에서 잡음 제거 오토인코더와 유사한 목적이 있다고 할 수 있다. 그러나 구체적으로 살펴보자면, 잡음 제거 오토인코더는 재구성의 강건성(robustness of reconstruction)을 확보하는 것이 목적이지만 축약 오토인코더는 대표성의 강건성(robustness of representation)을 확보하는 것이 목적이라는 점에서 차이가 있다 (Rifai 등, 2011). 다시 말해 잡음 제거 오토인코더는 잡음이 추가된 입력값으로부터 잡음이 제거된 출력값을 산출하도록 디코더 부분을 학습시키는 반면, 축약 오토인코더는 입력값에 약간의 변화가 있더라도 원래의 입력값을 대표할 수 있는 핵심 요소를 추출하는 인코더 부분을 학습한다. 축약 오토인코더로 아이템에 대한 핵심 요소를 추출하는 인코더를 학습하고자 하기 때문에 표기를 새롭게 정의하도록 한다. 전체 평점 행렬 $\mathbf{X} \in \mathbb{R}^{N\times M}$에 대하여 j번째 열은 $\mathbf{x}_j = (x_{1j}, \ldots, x_{Nj})^T$로 정의하면 인코더는 이를 입력값으로 받아 활성화 함수 $f(\cdot)$를 거쳐 $k \ll M$으로 차원이 축소된 잠재 변수를 $\mathrm{cae}(\mathbf{x}_j) \in \mathbb{R}^k$라고 지칭한다. 이때 활성화 함수를 구성하는 가중치 행렬은 $\mathbf{W}_1 \in \mathbb{R}^{k\times N}$, 편향 벡터는 $\mathbf{b}_1 \in \mathbb{R}^k$로 정의한다. 즉, 축약 오토인코더를 통해 추출한 아이템에 대한 잠재 요소를 의미하는 $\mathrm{cae}(\mathbf{x}_j)$는 다음과 같이 표기할 수 있고 이를 디코더의 입력값으로 받아 최종적으로 출력하는 값인 $\hat{\mathbf{x}}_j$은 식 (2.10)과 같이 계산된다.

$$\begin{aligned}\mathrm{cae}(\mathbf{x}_j) &= f(\mathbf{W}_1\mathbf{x}_j + \mathbf{b}_1), \quad j = 1, \ldots, M,\\ \hat{\mathbf{x}}_j &= g(\mathbf{W}_2\mathrm{cae}(\mathbf{x}_j) + \mathbf{b}_2) = g(\mathbf{W}_2 f(\mathbf{W}_1\mathbf{x}_j + \mathbf{b}_1) + \mathbf{b}_2), \quad j = 1, \ldots, M.\end{aligned} \tag{2.10}$$

축약 오토인코더의 목적함수는 식 (2.11)과 같다. 여기서 가중치 행렬 $\mathbf{W}_1, \mathbf{W}_2$에 대하여 $\mathbf{W}_2 = \mathbf{W}_1{}^T$라는 제약을 주고 $\mathbf{W}_1$을 $\mathbf{W}$로 정의한다. 따라서 $\hat{\mathbf{X}}$의 j번째 열은 $\hat{\mathbf{x}}_j = g(\mathbf{W}^T f(\mathbf{W}\mathbf{x}_j + \mathbf{b}_1) + \mathbf{b}_2)$, $j = 1, \ldots, M$으로 계산한다 (Zhang 등, 2017).

$$\begin{aligned}&\underset{\mathbf{W},\mathbf{b}_1,\mathbf{b}_2}{\operatorname{argmin}}\left(L\left(\mathbf{X}, \hat{\mathbf{X}}\right) + \lambda\|\mathcal{J}_f(\mathbf{X})\|_F^2\right)\\ &\text{where} \quad \|\mathcal{J}_f(\mathbf{X})\|_F^2 = \sum_{j=1}^{M}\left(\frac{\partial\,\mathrm{cae}(\mathbf{x}_j)}{\partial\,\mathbf{x}_j}\right)^2.\end{aligned} \tag{2.11}$$

식 (2.11)에서 $\|\mathcal{J}_f(\mathbf{X})\|_F^2$은 인코더에 해당하는 활성화 함수 $f(\cdot)$의 야코비안(Jacobian)에 대하여 Frobenius norm을 계산한 것으로, 인코더가 입력값으로부터 추출한 핵심 요소의 대표성에 대한 야코비안(Jacobian of representation)을 의미한다고 할 수 있다 (Rifai 등, 2011). 입력값을 인코더 부분에 통과시킬 때 입력값으로부터 추출된 핵심 요소의 편미분 값을 최소화시킴으로서 입력값에 작은 차이가 있더라도 핵심 요소는 민감하게 반응하지 않는 효과를 얻을 수 있다 (Aggarval, 2018). 따라서 SVD 방법 또는 SVD++ 방법만 사용한 경우는 잠재 요소의 단순 선형 관계에 초점을 두는 것에 비해 축약 오토인코더를 이용함으로써 아이템에 대해 추출된 비선형 잠재 요소를 고려할 수 있다. AutoSVD 방법은 SVD 방법에서 j 번째 아이템에 대한 잠재 요소를 나타내는 벡터 $\mathbf{q}_j$를 대신하여 $(\beta \cdot \text{cae}(\mathbf{x}_j) + \boldsymbol{\epsilon}_j)$를 대입한다. 여기서 β의 역할은 $\text{cae}(\mathbf{x}_j)$를 정규화(normalization) 시켜주는 초모수(hyperparameter)에 해당하며, $\boldsymbol{\epsilon}_j \in \mathbb{R}^k$는 축약 오토인코더에서 발생할 수 있는 오차에 해당하는 벡터이다. 따라서 AutoSVD 방법을 이용하여 $\hat{x}_{ij}$을 계산하면 식 (2.12)로 나타낼 수 있다

$$\hat{x}_{ij} = \mu + b_i + b_j + \Big(\beta \cdot \text{cae}(\mathbf{x}_j) + \boldsymbol{\epsilon}_j\Big)^T \mathbf{p}_i. \tag{2.12}$$

축약 오토인코더를 이용하여 아이템에 대한 잠재 요소를 추출하는 방법과 마찬가지로 사용자에 대한 잠재 요소를 추출하여 $\mathbf{p}_i$를 대신할 수 있으나 사실상 사용자의 고유 정보를 얻는 것에 많은 어려움이 있기 때문에 위와 같이 아이템에 대한 보조 정보를 활용하는 것이 합리적이다. 이렇게 계산된 값을 통해 AutoSVD 방법의 목적함수를 정의하면 식 (2.13)과 같다

$$\underset{\mathbf{p}^*, b^*, \boldsymbol{\epsilon}^*}{\text{argmin}} \sum_{(i,j)\in K(R)} \Big(\big(x_{ij} - \hat{x}_{ij}\big)^2 + \lambda\big(\|\mathbf{p}_i\|^2 + \|\boldsymbol{\epsilon}_j\|^2 + {b_i}^2 + {b_j}^2\big)\Big). \tag{2.13}$$

SGD 방법을 이용하여 모수 업데이트를 진행하는 AutoSVD 방법의 알고리즘은 다음과 같다.

Algorithm 1 Training algorithm for AutoSVD

procedure UPDATE PARAMETERS
  $e_{ij} \overset{\text{def}}{=\!=} x_{ij} - \hat{x}_{ij}$
  for each i, j **do**
    $b_i \leftarrow b_i + \gamma_1(e_{ij} - \lambda_1 \cdot b_i)$
    $b_j \leftarrow b_j + \gamma_1(e_{ij} - \lambda_1 \cdot b_j)$
    $\boldsymbol{\epsilon}_j \leftarrow \boldsymbol{\epsilon}_j + \gamma_2(e_{ij} \cdot \mathbf{p}_i - \lambda_2 \cdot \boldsymbol{\epsilon}_j)$
    $\mathbf{p}_i \leftarrow \mathbf{p}_i + \gamma_2(e_{ij}(\beta \cdot \text{cae}(\mathbf{x}_j) + \boldsymbol{\epsilon}_j) - \lambda_2 \cdot \mathbf{p}_i)$
    $\hat{x}_{ij} \leftarrow$ update by equation (2.12)
    $e_{ij} \leftarrow x_{ij} - \hat{x}_{ij}$
  end for
end procedure

2.5. AutoSVD++

SVD 방법과 축약 오토인코더를 결합한 AutoSVD 방법은 잠재 요소를 추출할 때 비선형 관계를 고려한다는 점에서 높은 정확도를 기대할 수 있으나 이는 명시적 피드백을 이용한다는 점에서 높은 희박성을 갖는다는 문제에 당면한다. 이를 보완하고자 내재적 피드백을 함께 고려한 방법인 SVD++ 방법에 축약 오토인코더를 결합한 것이 AutoSVD++ 방법이다. AutoSVD 방법에서 아이템에 대한 잠재 요소 벡터 $\mathbf{q}_j$를 대신한 것과 같은

방식으로 사용자에 대한 잠재 요소 벡터 $\mathbf{p}_i$를 SVD++ 방법에 의해 두 부분으로 나눌 수 있다. AutoSVD++ 방법을 이용하면 $\hat{x}_{ij}$을 식 (2.14)와 같이 계산한다.

$$\hat{x}_{ij} = \mu + b_i + b_j + \left(\beta \cdot \text{cae}(\mathbf{x}_j) + \boldsymbol{\epsilon}_j\right)^T \left(\mathbf{p}_i + |N(i)|^{-\frac{1}{2}} \sum_{j \in N(i)} \mathbf{y}_j\right). \tag{2.14}$$

즉, $(\beta \cdot \text{cae}(\mathbf{x}_j) + \boldsymbol{\epsilon}_j)$는 축약 오토인코더에 해당하는 부분이고, $(\mathbf{p}_i + |N(i)|^{-1/2} \sum_{j \in N(i)} \mathbf{y}_j)$는 SVD++ 방법에 해당하는 부분이다. AutoSVD++ 방법의 목적함수는 식 (2.15)로 나타낼 수 있다.

$$\underset{\mathbf{p}^*, b^*, \boldsymbol{\epsilon}^*, \mathbf{y}^*}{\text{argmin}} \sum_{(i,j) \in K(R)} \left(\left(x_{ij} - \hat{x}_{ij}\right)^2 + \lambda \left(\|\mathbf{p}_i\|^2 + \|\boldsymbol{\epsilon}_j\|^2 + {b_i}^2 + {b_j}^2 + \sum_{j \in N(i)} \|\mathbf{y}_j\|^2 \right) \right), \tag{2.15}$$

여기서 $\boldsymbol{\epsilon}^*, \mathbf{y}^*$는 $K(R)$에 속하는 i, j에 대하여 모든 $\boldsymbol{\epsilon}_j, \mathbf{y}_j$를 표현한 것이다. AutoSVD++ 방법을 이용하면 AutoSVD 방법과는 다르게 $\mathbf{y}_j$를 업데이트하는 과정이 필요하기 때문에 시간이 많이 소요된다는 단점이 있다. 그럼에도 불구하고 내재적 피드백을 행렬 인수분해를 통해 용이하게 결합할 수 있다는 장점과 높은 성능을 갖기 때문에 AutoSVD++ 방법은 합리적이라고 할 수 있다. SGD 방법을 이용하여 모수 업데이트를 진행하는 AutoSVD++ 방법의 알고리즘은 다음과 같다.

Algorithm 2 Training algorithm for AutoSVD++

procedure UPDATE PARAMETERS
  $e_{ij} \overset{\text{def}}{=\!=} x_{ij} - \hat{x}_{ij}$
  for each i, j **do**
    $b_i \leftarrow b_i + \gamma_1(e_{ij} - \lambda_1 \cdot b_i)$
    $b_j \leftarrow b_j + \gamma_1(e_{ij} - \lambda_1 \cdot b_j)$
    $\boldsymbol{\epsilon}_j \leftarrow \boldsymbol{\epsilon}_j + \gamma_2(e_{ij}(\mathbf{p}_i + |N(i)|^{-\frac{1}{2}} \sum_{j \in N(i)} \mathbf{y}_j) - \lambda_2 \cdot \boldsymbol{\epsilon}_j)$
    $\mathbf{p}_i \leftarrow \mathbf{p}_i + \gamma_2(e_{ij} \cdot (\beta \cdot \text{cae}(\mathbf{x}_j) + \boldsymbol{\epsilon}_j) - \lambda_2 \cdot \mathbf{p}_i)$
    $\forall j \in N(i) : \mathbf{y}_j \leftarrow \mathbf{y}_j + \gamma_2(e_{ij} \cdot |N(i)|^{-\frac{1}{2}}(\beta \cdot \text{cae}(\mathbf{x}_j) + \boldsymbol{\epsilon}_j) - \lambda_2 \cdot \mathbf{y}_j)$
    $\hat{x}_{ij} \leftarrow$ update by equation (2.14)
    $e_{ij} \leftarrow x_{ij} - \hat{x}_{ij}$
  end for
end procedure

3. 실제 데이터 분석

본 논문에서는 Python으로 실제 데이터의 모형 적합을 진행하였으며 개발 환경으로는 Google Colab에서 제공하는 GPU를 사용하였다. 딥러닝 프레임워크 중에서는 Tensorflow와 Keras를 이용하였고 구체적인 사양은 다음과 같다.

- Python: 3.6.9
- Tensorflow: 2.4.0
- Keras: 2.4.3
- CPU: Intel Xeon 2.20GHz

- GPU: Tesla P100-PCIE-16GB and Tesla T4
- RAM: 12.72GB

앞에서 언급한 모형들을 실제 데이터에 적용하고 각 모형을 비교하기 위하여 세 가지의 동일한 절차를 수행한다. 첫 번째는 5겹 교차 검증(5-fold cross-validation)을 반복 시행한다. 먼저 전체 데이터의 20%를 시험 데이터로, 80%를 훈련 데이터로 나눈다. 이 훈련 데이터를 다시 20%를 훈련 데이터로, 80%를 검증 데이터로 나눈다. 이때의 훈련 데이터를 앞서 나눈 훈련 데이터와 명칭을 구분하기 위하여 보조 훈련 데이터라고 지칭하도록 한다. 일반적으로 전체 데이터를 시험, 훈련, 검증 데이터로 분할할 때 행 또는 열을 기준으로 한다. 그러나 추천 시스템에서는 사용자가 행으로, 아이템이 열로 이루어진 평점 행렬을 만든 후 사용자를 기준으로 분할할 경우, 한 사용자의 정보가 시험 데이터에는 존재하지만 훈련, 검증 데이터에는 존재하지 않을 수 있다. 아이템을 기준으로 분할할 경우도 마찬가지이다. 즉, 한 가지만을 기준으로 데이터를 분할할 경우 시험, 훈련, 검증 데이터 각각이 보유하는 정보가 편향될 수 있다. 이러한 문제의 발생을 방지하고자 본 논문을 비롯하여 추천 시스템에서는 실제 평가가 이루어진 값을 기준으로 특정 사용자의 평점 정보를 시험, 훈련, 검증 데이터에 모두 보유할 수 있도록 분할하였다. 따라서 특정 사용자가 단 한번도 시험 데이터에 등장하지 않을 수 있는 가능성을 배제하였다.

두 번째는 각 모형에서 손실함수로서 식 (3.1)에 의해 계산된 마스크 평균 제곱 오차(masked mean squared error; masked MSE)에 제곱근을 취한 형태로 정의한다. 추천 시스템에서 다루는 데이터는 대개 희박성을 갖는다는 특징으로 인하여 손실함수로서 일반적인 평균 제곱근 오차를 정의할 경우 결측값을 대체한 값에 크게 좌우된다. 따라서 본 논문에서는 손실함수로서 실제 평가가 이루어진 점수에 대한 평균 제곱근 오차(이를 마스크 평균 제곱 오차라 한다)를 정의한다 (Sedhain 등, 2015; Kuchaiev와 Ginsburg, 2017).

$$\text{masked RMSE} = \sqrt{\sum_{(i,j)\in K(R)} \frac{\left(x_{ij} - \hat{x}_{ij}\right)^2}{n}}, \tag{3.1}$$

$K(R) = \{(i,j)|x_{ij} \neq 0,\ i = 1,\ldots,N,\ j = 1,\ldots,M\}$, $n = \sum_{i=1}^{N}\sum_{j=1}^{M} I(x_{ij} \neq 0)$ 여기서 n은 실제 평가가 이루어진 개수를 의미한다.

세 번째는 변분 오토인코더의 경우 결측값을 식 (3.2)에 의해 계산한 값으로 대체한 후 모형을 적합시켰다는 점이다. 입력값의 분포를 학습하는 변분 오토인코더는 결측값을 대체하지 않을 경우 모형 학습이 제대로 진행되지 않았기 때문에 결측값을 대체하는 작업을 수행하였다. 데이터에서 결측값의 경우 1 점으로 대체하거나 (Wu 등, 2016) 3점으로 대체하는 등 (Sedhain 등, 2015) 임의의 점수로 대체하는 것과 다르게, 본 논문에서는 평점 행렬의 행 평균과 열 평균을 이용하여 계산한 값으로 결측값을 대체한다.

$$x^*_{ij} = \frac{1}{2}\left(\sum_{j=1}^{M}\frac{x_{ij}}{n_j} + \sum_{i=1}^{N}\frac{x_{ij}}{n_i}\right). \tag{3.2}$$

이때 n_i는 i번째 사용자가 실제 평가를 한 아이템의 개수를, n_j는 j번째 아이템의 평점을 매긴 사용자의 수라고 정의한다. 만약 사용자가 평가를 하지 않은 결측값일 경우 i 번째 사용자가 전체 아이템 중 평가를 한 점수들의 평균 ($\sum_{j=1}^{M} x_{ij}/n_j$)과 j번째 아이템이 전체 사용자로부터 받은 점수들의 평균 ($\sum_{i=1}^{N} x_{ij}/n_i$)을 합하여 2로 나눈 값으로 대체한다. 식 (3.2)를 이용하여 결측값을 대체한 평점 행렬을 $\mathbf{X}^* \in \mathbb{R}^{N\times M}$으로 정의하면 i 번째 행은 $\mathbf{x}_i^* = (x_{i1}^*,\ldots,x_{iM}^*)^T$로 정의할 수 있다. 따라서 변분 오토인코더의 목적함수 중 $D_{\mathrm{KL}}(q_\phi(\mathbf{z}_i|\mathbf{x}_i)\|\mathbf{z}_i))$를 $D_{\mathrm{KL}}(q_\phi(\mathbf{z}_i|\mathbf{x}_i^*)\|\mathbf{z}_i))$로 계산한다.

이러한 동일 조건하에서 5겹 반복 교차 검증을 진행하여 초모수의 값들을 선택한다. 그 결과, 각 모형별로 평균 오차가 가장 작은 초모수 조합으로 이루어진 모형을 최종 모형으로 정의한다. 이를 각 초모수 조합별로

Table 1: Mean of masked RMSE of autoencoder and denoising autoencoder and Kullback-Leibler divergence of variational autoencoder on MovieLens 1M data set. For each model, two activation functions are used. The first one is the activation function for each layer, and the second one is for the last layer before an output layer

Models	Nodes	Dropout	Tanh, ReLU	ReLU, ReLU	SeLU, ReLU
Autoencoder	(165, 54, 18)	0.4	0.9562	0.9563	0.9521
		0.6	0.9555	0.9561	0.9564
	(350, 245, 171)	0.4	**0.9460**	1.0543	0.9724
		0.6	0.9553	0.9553	0.9553
	(165, 54, 18, 6)	0.4	0.9577	0.9594	0.9556
		0.6	0.9556	0.9561	0.9565
	(350, 245, 171, 120)	0.4	0.9593	1.1824	1.0456
		0.6	0.9480	0.9971	0.9751
Denoising AE	(165, 54, 18)	0.4	0.9622	0.9515	0.9516
		0.6	0.9524	0.9521	0.9516
	(350, 245, 171)	0.4	0.9544	0.9558	0.9841
		0.6	0.9515	0.9515	**0.9514**
	(165, 54, 18, 6)	0.4	0.9643	0.9528	0.9517
		0.6	0.9525	0.9520	0.9519
	(350, 245, 171, 120)	0.4	0.9577	0.9556	0.9634
		0.6	0.9593	0.9530	0.9551
Variational AE	(165, 54, 18)	0.4	106.1066	-	-
		0.6	105.8023	-	-
	(350, 245, 171)	0.4	110.7079	-	-
		0.6	127.3426	-	-
	(165, 54, 18, 6)	0.4	99.3612	-	-
		0.6	**96.4264**	-	-
	(350, 245, 171, 120)	0.4	99.1027	-	-
		0.6	101.0455	-	-

시행하여 가장 작은 값에 해당하는 초모수 조합을 모형의 최적의 초모수 조합으로 결정한다. 이렇게 결정된 최적의 모형을 시험 데이터를 이용하여 평균 제곱 오차를 얻는다. 이 역시 시험 데이터를 이용하여 예측 오차에 대한 평균을 계산함으로써 최종 모형의 결과로 정의하였다. 이러한 방법을 동일하게 적용하여 초모수의 값들을 선택하였다. 이제 본 논문에서 분석한 세 개의 데이터를 설명하도록 한다.

1. **MovieLens 1M data** 미네소타 대학교의 GroupLens 연구조직에 의해 수집된 MovieLens는 영화에 대한 사용자들의 평점을 담고 있는 데이터이다. 데이터의 크기에 따라 1B, 1M, 10M, 20M 그리고 100K 데이터셋이 제공되고 있으며 2018년 9월까지 꾸준히 업데이트되고 있다. 본 논문에서는 2003년 2월에 발표된 MovieLens 1M (https://grouplens.org/datasets/movielens/1m/) 데이터를 사용하였다 (Harper와 Konstan, 2015). 데이터는 6,040명의 사용자가 3,900개의 영화에 대하여 1점부터 5점까지 평점을 매긴 것으로, 총 1,000,209개의 평점을 담고 있다. 본 논문에서는 3,900개의 영화 중에서 사용자로부터 평가를 가장 많이 받은 상위 500개의 영화만을 고려하였다. 따라서 약 82%의 희박성 비율 (sparsity rate)을 갖는다.
2. **Amazon Review Data** Amazon에서 판매하는 모든 제품에 대해서 평점이나 후기와 같은 리뷰와 제품의 종류, 가격, 이미지와 같은 메타데이터를 담고 있으며 1996년 5월부터 축적된 데이터이다. 제품의 종류에 따라 패션과 뷰티부터 장난감과 비디오 게임까지 다양한 제품군에 대하여 제공하고 있다. 그중 본 논문에서는 2018년 10월에 발표된 책에 대한 평점 데이터 (https://nijianmo.github.io/amazon/index.html)를 사용

Table 2: Mean of masked RMSE of autoencoder and denoising autoencoder and Kullback-Leibler divergence of variational autoencoder on Amazon Review data set. For each model, two activation functions are used. The first one is the activation function for each layer, and the second one is for the last layer before an output layer

Models	Nodes	Dropout	Tanh, ReLU	ReLU, ReLU	SeLU, ReLU
Autoencoder	(165, 54, 18)	0.4	0.9184	0.9291	**0.8990**
		0.6	0.9088	0.9040	0.9016
	(350, 245, 171)	0.4	0.9880	0.9528	0.9334
		0.6	0.9538	0.9017	0.9070
	(165, 54, 18, 6)	0.4	1.0823	0.9252	0.8992
		0.6	0.9525	0.9520	0.9519
	(350, 245, 171, 120)	0.4	0.9949	1.0264	0.9287
		0.6	0.9483	0.9170	0.9609
Denoising AE	(165, 54, 18)	0.4	1.0524	1.0647	0.9573
		0.6	1.0275	1.0155	0.9587
	(350, 245, 171)	0.4	0.9978	1.0460	1.0519
		0.6	0.9562	1.0369	1.0323
	(165, 54, 18, 6)	0.4	1.0581	0.9561	0.9241
		0.6	0.9288	**0.9048**	0.9056
	(350, 245, 171, 120)	0.4	1.0262	0.9692	1.2873
		0.6	1.0395	0.9228	1.6270
Variational AE	(165, 54, 18)	0.4	50.4234	-	-
		0.6	82.5037	-	-
	(350, 245, 171)	0.4	80.5075	-	-
		0.6	87.2212	-	-
	(165, 54, 18, 6)	0.4	**44.9514**	-	-
		0.6	50.6193	-	-
	(350, 245, 171, 120)	0.4	61.5598	-	-
		0.6	61.9918	-	-

하였다 (Ni와 McAuley, 2019). 데이터는 15,362,619명의 사용자가 2,908,451 개의 책에 대하여 1점부터 5점까지 평점을 매긴 것으로, 총 51,311,620개의 평점을 담고 있다. 그러나 이는 희박성 비율이 99.99%로 과도하게 높은 경향이 있었다. 본 논문에서는 희박성 비율이 과도하게 높아지는 것을 방지하기 위해 평가를 많이 한 상위 461 명의 사용자와 그에 대한 495개의 책만을 고려하였다. 그 결과로 생성된 데이터는 약 94%의 희박성 비율을 갖는다.

3. **Yelp Review Data** 2004년부터 시작된 Yelp는 레스토랑이나 여러 서비스에 대한 소비자들의 리뷰를 공유할 수 있는 플랫폼으로, 현재 대표적인 온라인 관광 소셜 커머스 포털 중 하나로 자리잡았다. 특히 사용자들이 레스토랑에 대한 후기를 공유하고 서로 친목을 형성할 수 있도록 모바일 플랫폼을 제공하여 더욱 많은 양의 데이터를 보유하고 있다. 본 논문에서는 2018년에 발표된 Yelp Review (https://www.yelp.com/dataset) 데이터를 사용하였다. 데이터는 1,326,101명의 사용자가 174,567개의 레스토랑에 대하여 1점부터 5점까지 평점을 매긴 것으로, 총 5,261,668개의 평점을 담고 있다. 그러나 이 경우에도 마찬가지로 희박성 비율이 99.998%로 과도하게 높기 때문에 본 논문에서는 평가를 가장 많이 한 상위 8,748명의 사용자와 그에 해당하는 500개의 레스토랑만 고려하였다. 그 결과 약 98%의 희박성 비율을 갖는 데이터를 이용하였다.

우선 모형의 학습 과정에 대해 기술하고 그 결과를 요약하면 다음과 같다. 데이터에 오토인코더, 잡음제거 오토인코더 그리고 변분 오토인코더를 학습시킬 때 각 모형에서 은닉층(layers)의 수와 층을 거치면서

Table 3: Mean of masked RMSE of autoencoder and denoising autoencoder, and Kullback-Leibler divergence of variational autoencoder on Yelp Review data set. For each model, two activation functions are used. The first one is the activation function for each layer, and the second one is for the last layer before an output layer

Models	Nodes	Dropout	Tanh, ReLU	ReLU, ReLU	SeLU, ReLU
Autoencoder	(165, 54, 18)	0.4	0.9933	0.9870	0.9793
		0.6	0.9882	0.9832	0.9828
	(350, 245, 171)	0.4	0.9531	1.3139	**0.9273**
		0.6	0.9833	0.9806	0.9842
	(165, 54, 18, 6)	0.4	1.1148	1.0053	0.9825
		0.6	1.0225	0.9835	0.9837
	(350, 245, 171, 120)	0.4	1.0823	1.4550	1.2050
		0.6	0.9839	1.1384	1.4583
Denoising AE	(165, 54, 18)	0.4	1.0153	1.1909	1.0363
		0.6	1.0052	1.2017	1.0653
	(350, 245, 171)	0.4	1.0153	1.1909	1.0068
		0.6	0.9860	1.1259	1.0274
	(165, 54, 18, 6)	0.4	1.0709	1.0621	1.0126
		0.6	0.9986	**0.9831**	0.9853
	(350, 245, 171, 120)	0.4	0.9964	1.4327	1.0302
		0.6	0.9981	3.3360	1.0498
Variational AE	(165, 54, 18)	0.4	42.6365	-	-
		0.6	43.2350	-	-
	(350, 245, 171)	0.4	60.2496	-	-
		0.6	77.6588	-	-
	(165, 54, 18, 6)	0.4	**35.9175**	-	-
		0.6	36.6993	-	-
	(350, 245, 171, 120)	0.4	39.9970	-	-
		0.6	42.0888	-	-

추출할 핵심 요소(nodes)의 수, 그리고 과적합을 방지하기 위해 각 층에서 사용하지 않을 핵심 요소의 비율(dropout) 그리고 층을 연결하는 활성화 함수를 초모수로 지정하여 조절하였다. Table 1, Table 2, Table 3은 오토인코더, 잡음 제거 오토인코더, 변분 오토인코더를 세 데이터에 학습시킨 결과이다. 이 세 개의 표에서 나타나는 활성화 함수의 경우 첫 번째는 은닉층 사이에 해당하는 활성화 함수를, 두 번째는 최종 출력 직전의 활성화 함수를 의미한다. 오토인코더와 잡음 제거 오토인코더는 모형의 평가를 masked RMSE의 값으로 한다. 한편, 변분 오토인코더는 모형의 정의 자체가 실제 분포함수와 가정한 분포함수의 거리를 쿨백-라이블러 발산을 측정하여 최소화하는 방법론이기 때문에 이 측도를 성능의 기준으로 고려하여 모형을 평가한다.

모든 데이터 분석에서 교차 검증 과정에서 발생하는 임의성으로 인한 변동을 줄이기 위하여 위에 서술한 전체 과정을 5번 반복 실험한 후의 결과값을 평균내어 요약한다. 평균값의 표준오차(standard error)들은 매우 작은 값이어서 테이블에 따로 표기되어 있지 않다. 각 테이블에서 최적의 학습 결과를 보여주는 초모수 결합의 결과는 굵은 숫자로 표기되어 있다.

Table 4는 축약 오토인코더를 세 데이터에 학습시켜 얻어낸 masked RMSE를 보여준다. 축약 오토인코더로 학습한 cae($\mathbf{x}_j$)을 이용하여 AutoSVD와 AutoSVD++를 학습하여 얻은 masked RMSE은 Table 5에 나타나 있다.

AutoSVD 방법과 AutoSVD++ 방법을 이용할 때는 축약 오토인코더의 결과에서 식 (3.1)에서 정의한

Table 4: Mean of masked RMSE of contractive autoencoder on the three data sets at various hyperparameter settings. For each model, two activation functions are used. The first one is the activation function for each layer, and the second one is for the last layer before an output layer

	Models	Nodes	Dropout	Tanh, ReLU	ReLU, ReLU	SeLU, ReLU
MovieLens 1M data	Contractive AE	(165, 54)	0.4	25.2896	7.2835	17.1496
			0.6	29.7528	6.4412	11.0278
		(350, 245)	0.4	20.9779	6.7658	11.8221
			0.6	30.7249	5.6920	**5.5486**
Amazon Review data	Contractive AE	(165, 54)	0.4	7.2179	7.5730	7.7275
			0.6	7.5779	7.7128	7.8975
		(350, 245)	0.4	**6.9511**	7.4287	7.7599
			0.6	7.4132	7.7153	8.2678
Yelp Review data	Contractive AE	(165, 54)	0.4	0.2059	0.0350	0.3279
			0.6	0.1994	**0.0101**	0.5782
		(350, 245)	0.4	0.1744	0.0457	0.3043
			0.6	0.2491	0.0101	0.5920

Table 5: Mean of masked RMSE of AutoSVD and AutoSVD++ on the three data sets at various hyperparameter values

	Model	$\lambda = 0.1$, $\gamma = 0.01$	$\lambda = 0.1$, $\gamma = 0.001$	$\lambda = 0.1$, $\gamma = 0.0001$	$\lambda = 0.01$, $\gamma = 0.001$	$\lambda = 0.001$, $\gamma = 0.0001$
MovieLens 1M data	AutoSVD	**0.6820**	0.7029	0.7378	0.6954	-
	AutoSVD++	**0.6859**	-	0.7216	0.7038	-
Amazon Review data	AutoSVD	**0.7057**	0.7922	-	-	-
	AutoSVD++	0.8056	0.7339	0.7232	0.7339	**0.7231**
Yelp Review data	AutoSVD	0.7423	0.7413	**0.7411**	0.7411	-
	AutoSVD++	0.7434	0.7429	0.7428	**0.7428**	-

masked RMSE 값이 가장 작은 조합의 초모수로 학습한 모형으로부터 추출한 cae($\mathbf{x}_j$)를 이용하였으며 모형의 알고리즘 설명에서 언급했던 λ 와 γ 를 초모수로 지정하여 조절한다. 결과표 내의 비어있는 칸들은 그에 상응하는 초모수나 활성화 함수의 사용이 적절하지 않아 과적합이 발생한 경우를 의미한다.

마지막으로 위와 같은 학습 과정을 통하여 최종적으로 얻어진 최적의 모수 조합을 이용하여 MovieLens 1M, Amazon Review, 그리고 Yelp Review 데이터에 위에서 기술한 다섯 개의 모형들을 적합시켜 구한 masked RMSE와 mean absolute error (MAE)값들이 Table 6에 나타나 있다.

데이터마다 최적의 초모수 조합이 달라 일정한 패턴을 발견하기는 어렵지만, 고객 평가 데이터이고 0이 많이 있는 데이터라면 이 테이블에 나와 있는 셋팅들을 참고한다면 도움이 될 수도 있을 것이라 생각한다.

4. 결론 및 논의

희박성 비율이 서로 다른 세 개의 데이터에 다섯 가지 모형을 적합시킨 결과, 희박성 비율이 높을수록 은닉층이 많고 각 은닉층에서 사용하지 않을 핵심 요소의 비율이 낮은 모형의 성능이 우수하였다. 이는 이미 희박성 비율이 높기 때문에 보유한 정보를 최대한 이용하는 것이 유리하기 때문이라고 볼 수 있다. 반대로 희박성 비율이 상대적으로 낮은 데이터인 MovieLens 1M의 경우 각 은닉층에서 사용하지 않을 핵심 요소의 비율이

Table 6: Masked RMSE and MAE of the five models with 'optimally' tuned parameters on the three data sets. For each model, two activation functions are used. The first one is the activation function for each layer, and the second one is for the last layer before an output layer

	Hyperparameters	AE	DAE	VAE	AutoSVD	AutoSVD++
MovieLens 1M data	Nodes	(350, 245, 171)	(359, 245, 171)	(165, 54, 18, 6)	(165, 54)	(165, 54)
	Dropout	0.4	0.6	0.6	0.6	0.6
	Activation function	Tanh, ReLU	SeLU, ReLU	Tanh, ReLU	SeLU, ReLU	SeLU, ReLU
	λ ,γ	-	-	-	0.1, 0.01	0.1, 0.01
	masked RMSE	0.8898	0.9551	0.9723	**0.8600**	0.8651
	MAE	0.7012	0.7814	0.7611	**0.6768**	0.6811
Amazon Review data	Nodes	(165, 54, 18)	(165, 54, 18, 6)	(165, 54, 18, 6)	(350, 245)	(350, 245)
	Dropout	0.4	0.6	0.4	0.4	0.4
	Activation function	SeLU, ReLU	ReLU, ReLU	Tanh, ReLU	Tanh, ReLU	Tanh, ReLU
	λ ,γ	-	-	-	0.1, 0.001	0.001, 0.0001
	masked RMSE	**0.8493**	0.8505	0.8981	0.9355	0.8763
	MAE	**0.6403**	0.6665	0.6905	0.7097	0.7242
Yelp Review data	Nodes	(350, 245, 171)	(165, 54, 18, 6)	(165, 54, 18, 6)	(165, 54)	(165, 54)
	Dropout	0.4	0.6	0.4	0.4	0.4
	Activation function	SeLU, ReLU	ReLU, ReLU	Tanh, ReLU	ReLU, ReLU	ReLU, ReLU
	λ ,γ	-	-	-	0.001, 0.001	0.0001, 0.001
	masked RMSE	**0.9307**	0.9834	1.0577	0.9512	0.9524
	MAE	**0.7368**	0.7676	0.7920	0.7392	0.7406

높은 경우 모형 성능이 향상되는 것을 볼 수 있었다.

세 데이터 모두 공통적으로 오토인코더 방법이 잡음 제거 오토인코더와 변분 오토인코더에 비해 성능이 우수하였다. 그러나 희박성 비율이 높은 Amazon Review와 Yelp Review 데이터의 경우 AutoSVD 방법과 AutoSVD++ 방법에 비해 잡음 제거 오토인코더와 변분 오토인코더의 성능이 더욱 우수하였다. 이를 통해 잡음 제거 오토인코더와 변분 오토인코더는 입력값의 대표성이 높은 잠재 요인을 추출하는 것에 적합하다는 것을 알 수 있다. 그럼에도 불구하고 MovieLens 1M 데이터의 경우 AutoSVD 방법과 AutoSVD++ 방법의 성능이 매우 우수한 것을 발견하였다. 따라서 두 방법론에서 사용하는 cae($\mathbf{x}_j$) 부분이 모형 성능을 결정하는 것에 매우 중요한 역할을 한다고 볼 수 있다. 즉, 축약 오토인코더의 초모수를 더욱 미세하게 조절한다면 AutoSVD 방법과 AutoSVD++ 방법의 성능이 향상될 것임을 기대할 수 있다.

본 논문은 다양한 초모수 조합으로 모형을 실제 데이터로 학습시켜 가장 성능이 좋은 딥러닝 기반의 추천 시스템 모형을 찾고자 다양한 시도를 하였다는 점에서 의의가 있다. 추천 시스템 종류 중에서 협업 필터링 방법과 하이브리드 방법을 이용한 오토인코더 기반의 딥러닝 모형들을 비교하였으며 영화, 책, 레스토랑에 대하여 사용자가 1점부터 5점까지 매긴 평점 데이터를 분석하였다. 각 데이터는 희박성 비율이 다르다는 차이점을 갖고 있지만 세 개의 데이터가 각각 82%, 94%, 98%의 높은 비율로 고객의 평점을 가지고 있지 않다는 점을 고려할 때, 이 논문에서 고려한 딥러닝 기반의 오토인코더를 이용한 추천 시스템의 성능은 평점에 대한 예측 오차가 대부분 1점 미만으로 만족할만한 성능을 보인다고 할 수 있겠다.

References

Aggarwal CC (2018). *Neural Networks and Deep Learning*, Springer, NewYork.

Ali SM, Nayak GK, Lenka RK, and Barik RK (2018). Movie recommendation system using genome tags and content-based filtering, *Advances in Data and Information Sciences*(pp.85–94), Springer, NewYork.

Harper FM and Konstan JA (2015). The movielens datasets: History and context, *ACM Transactions on Interactive Intelligent Systems*, **5**, 1–19.

Kingma DP and Welling M (2013). *Auto-Encoding Variational Bayes*, arXiv preprint arXIV:1312.6114.

Koren Y (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 426–434.

Koren Y, Bell R, and Volinsky C (2009). Matrix factorization techniques for recommender systems, *Computer*, **42**, 30–37.

Kuchaiev O and Ginsburg B (2017). *Training Deep Autoencoders for Collaborative Filtering*, arXiv preprint arXiv:1708.01715.

Ni J, Li J, and McAuley J (2019). Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 188–197.

Odaibo S (2019). *Tutorial: Deriving the Standard Variational Autoencoder (vae) Loss Function*, arXiv preprint arXiv:1907.08956.

Rifai S, Vincent P, Muller X, Glorot X, and Bengio Y (2011). Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning June 2011*, 833—840

Schafer JB, Frankowski D, Herlocker J, and Sen S (2007). Collaborative filtering recommender systems, *The Adaptive Web*(pp. 291–324), Springer, NewYork.

Sedhain S, Menon AK, Sanner S, and Xie L (2015). Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*, 111–112.

Vincent P, Larochelle H, Bengio Y, and Manzagol PA (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, 1096–1103.

Wu Y, DuBois C, Zheng AX, and Ester M (2016). Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 153–162.

Zhang S, Yao L, Sun A, and Tay Y (2019). Deep learning based recommender system: A survey and new perspectives, *ACM Computing Surveys (CSUR)*, **52**, 1–38.

Zhang S, Yao L, and Xu X (2017). Autosvd++ an efficient hybrid collaborative filtering model via contractive auto-encoders. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 957–960.

Zheng N and Xue J (2009). *Manifold Learning*(pp.87-119), Springer, London.

Received January 21, 2021; Revised February 15, 2021; Accepted February 15, 2021

오토인코더를 이용한 딥러닝 기반 추천시스템 모형의 비교 연구

이효진[a], 정윤서[1,a]

[a]고려대학교 통계학과

요 약

추천 시스템은 고객의 데이터를 이용하여 개인 맞춤화된 상품을 추천한다. 추천 시스템은 협업 필터링, 콘텐츠 기반 필터링 그리고 이 두 가지를 합친 하이브리드 방법의 세 가지로 크게 나누어진다. 이 연구에서는 딥러닝 방법론에 기초한 오토인코더를 이용한 추천 시스템에 대한 소개와 그 모형들의 비교 연구를 진행한다. 오토인코더는 데이터 행렬에 0이 많은 경우의 문제를 효과적으로 다룰 수 있는 딥러닝 기반의 비지도학습 모형이다. 이 연구에서는 세 개의 실제 데이터를 이용하여 다섯 가지 종류의 오토인코더 기반 모형들을 비교한다. 처음의 세 개 모형은 협업 필터링에 속한 모형이고 나머지 두 개의 모형은 하이브리드 모형이다. 실제 데이터는 고객의 평점 데이터이고, 대부분의 평점이 없어서 희박성 비율이 높다는 특징이 있다.

주요용어: 오토인코더, 딥러닝, 추천 시스템, 희박 행렬

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No.2019R1F1A1040515, No.2019R1A4A1028134).

[1]교신저자: (02841) 서울특별시 성북구 안암로 145, 고려대학교 통계학과. E-mail: yoons77@korea.ac.kr