이화여자대학교 대학원

2018학년도

석사학위 청구논문

딥 오토 인코더를 이용한 3차원 텐서에서의 추천 시스템

컴 퓨 터 공 학 과

박 진 아

2019

딥 오토 인코더를 이용한 3차원 텐서에서의 추천 시스템

이 논문을 석사학위 논문으로 제출함

2019년 1월

이화여자대학교 대학원

컴 퓨 터 공 학 과 박 진 아

박 진 아 의 석사학위 논문을 인준함

지 도 교 수 용 환 승 _____

심 사 위 원 이 민 수 _____

용 환 승 _____

박 상 수 _____

이화여자대학교 대학원

목 차

표 목 차

그 림 목 차

논 문 개 요

본 논문은 차원축소(dimensional reduction)에 사용되는 협업 필터링(collaborative filtering)분야의 딥러닝(deep learning) 모델인 오토인코더(Autoencoder)를 텐서플로우에서 사용하여 추천 시스템(recommendation system)을 제안한다.

본 연구에서는 입력 층(Input layer)과 하나의 은닉 층(one-hidden layer), 출력 층(output layer)으로 구성되어 총 3개의 층(layer)으로 이루어진 오토인코더와 은닉 층(hidden layer)을 더 추가해서 깊어진 모델인 딥 오토인코더(deep Autoencoder)를 사용한다. 오토인코더로 추천하였을 때의 결과와 딥 오토인코더를 사용했을 때의 결과를 비교하여 분석하고 이를 통해 추천 성능 정확성을 높이는 것을 목적으로 한다.

기존의 연구에서는 입력 데이터로 사용자와 아이템, 평점으로 이루어진 2차원 행렬(2-dimensional arrays) 데이터를 이용하였다. 영화나 식당 평점 데이터를 분석하여 사용자, 아이템 데이터 기반으로 평점 데이터를 분석하였지만 본 연구에서는 시간 축을 추가한 3차원 텐서(3-dimensional Tensor) 데이터를 통해 각 항목들 간의 연관성을 분석하여 사용자와 아이템 그리고 시간을 고려한 추천 시스템을 제안한다.

결과적으로 은닉 층이 추가된 딥 오토인코더 모델을 사용하였을 때 은닉 층이 1개인 오토인코더 모델보다 추천 성능이 높아진 것을 확인하였으며 3차원 텐서 데이터에서 평점 데이터에 영향을 미치는 각 데이터 항목들 간의 연관성을 분석하였다.

Ⅰ. 서론

A. 연구배경 및 내용

최근 몇 년간 추천 시스템(recommendation system)[1] 은 전자 상거래(e-commerce) 등을 포함한 온라인 네트워크 거래에서 중요한 역할을 해오고 있다. 추천 시스템을 잘 이용하면 소비자들의 만족도를 높일 수 있을 뿐만 아니라 상품이나 서비스를 제공하는 업체도 이익을 창출할 수 있기 때문에 더욱 주목받았고 최근까지 활발히 이용되고 있다.

영화 제작사이자 스트리밍 웹사이트인 넷플릭스(Netflix)의 영화중에서 시청 된 영화의 약 80%정도가 추천 시스템에 의해서 시청될 만큼 추천 시스템은 생활에 밀접하게 사용되고 있다.[2] 현재 추천 시스템 기술 중 협업 필터링 (collaborative filtering)은 다양한 분야에서 이용되고 있으며, 협업 필터링 기법을 통해 사용자들의 과거 활동들에 기반을 두어 상품에 대한 사용자의 선호도를 분석하고 예측한다.[3]

다양한 협업 필터링 기법 중에서 기존에 가장 많이 쓰였던 기법 중 하나인 행렬 분해(Matrix factorization)[4, 5] 방법은 사용자와 상품간의 관계를 파악하기 위해 사용되었다. 딥러닝(deep learning) 에 대한 관심이 증가하고 개념이 발전되며 협업 필터링 성능을 향상시키기 위한 딥러닝 기술과의 통합적인 연구가 많이 진행되었고 딥러닝 기술이 적용된 협업 필터링 기법이 발전되었다.

본 연구에서는 딥러닝 협업 필터링 기법인 오토인코더(Autoencoder)[6, 7]를 사용하여 영화 평가 데이터를 분석하는 모델을 구현하였다. 이전의 오토인코더는 크게 두 가지 방식으로 분류할 수 있는데, 사용자 기반의(user-based) 방법과 아이템 기반의(item-based) 방법으로 분류한다.[8] 기존의 오토인코더 연구에서 2차원 행렬(2-dimensional arrays) 데이터를 사용하여 사용자와 아이템 기반으로 평점을 분석한 것을 확장해 3차원 텐서(3-dimensional Tensor) 데이터를 분석해 사용자와 아이템 기반으로 평점을 분석할 뿐만 아니라 사용자와 시간, 아이템과 시간과의 관계를 고려하여 분석하였다. 분석한 결과는 오토인코더를 통해 예측한 벡터와 입력

데이터로 주어진 벡터의 오차율을 계산한 RMSE(Root Mean Squared Error)을 통해 확인한다.

또한 텐서플로우(Tensorflow)[9] 기반의 GPU환경에서 오토인코더를 통한 분석 결과와 오토인코더 모델에 3개 이상의 은닉 층(hidden layer)을 추가한 딥 오토인코더(deep Autoencoder)에서의 분석 결과를 비교하여 추천 성능을 비교 평가 한다.

B. 논문 진행 구성

본 논문은 다음과 같이 진행한다. 2장에서는 딥러닝(deep learning)에서의 추천 시스템(recommendation system) 방법과 데이터를 딥러닝(deep learning)을 통해 분석하는 관련 연구와 동향에 대하여 살펴본다. 3장에서는 본 연구에서 제안한 딥러닝 모델인 오토인코더와 딥 오토인코더의 구조에 대해 다룬다. 4장에서는 실험 환경과 실험에 사용된 2차원 데이터와 3차원 데이터에 대하여 설명하고 실험 방법 및 결과를 보여준다. 그리고 3차원 데이터를 활용해 실험한 오토인코더와 딥 오토인코더의 연구 결과를 확인 및 비교하고 데이터 항목 간 연관성을 분석한다. 그 후 분석한 결과에 대하여 논의한다.

마지막으로 5장에서 본 연구의 결론을 짓고 향후 연구 방향과 계획에 대해 다루고 논문을 마무리 짓는다.

Ⅱ. 관련 연구 및 동향

본 장에서는 추천 시스템에 사용되는 협업 필터링 방법에 대해 서술하고 추천 시스템에 사용되는 방법과 딥러닝 기술이 적용된 관련 연구들에 대하여 살펴본다. 그리고 빅 데이터를 딥러닝에 적용해 연구한 사례들에 대하여 다룬다.

A. 딥러닝(Deep learning)에서의 추천 시스템

아마존(Amazon)이나 넷플릭스(Netflix)같은 전자 상거래(e-commerce) 웹 사이트들은 추천 시스템을 활용하여 사용자들에게 그들의 상품들을 추천한다. 추천 시스템 방법은 크게 두 가지로 나눌 수 있는데 내용 기반 추천 기법(context-based)과 개인화(personalized) 추천 기법으로 나눌 수 있다.

내용 기반 추천 기법(context-based)추천 기법은 장소나 날짜, 시간과 같은 요소들을 취해 추천에 적용하는 기법이다.[10] 사용자 개인화(personalized) 추천 기법은 협업 필터링(collaborative filtering)방법으로 접근하여 사용자들에게 상품을 추천한다. 이 방법을 통해 특정 사용자의 취향에 따른 관심사를 유추하고 예측할 수 있는데, 비슷한 관심사를 가진 다른 사용자들의 선호도와 특정 사용자의 취향에 기반을 두어 사용자에게 상품을 추천한다. 이 기법에서는 무작위로 선별된 사용자 그룹과 관심사와 취향이 비슷한 사용자 그룹이 있을 때 관심사와 취향이 같은 사용자 그룹이 특정 상품에 관한 의견이 비슷할 것이라고 가정한다.

추천 시스템을 설계할 때는 이 예측의 정확성을 높이는 것을 목표로 한다. 영화 제작사이자 스트리밍 웹 사이트인 넷플릭스는 영화에 대한 사용자들의 평점을 예측하는 알고리즘의 정확성을 높이기 위해 넷플릭스 주관 영화 추천 알고리즘 개선 대회(Netflix Prize Contest)를 개최할 만큼 추천 시스템은 산업에서 밀접하게 사용되고 있다.[11] 추천 시스템 방법의 성능을 평가할 때는 주로 예측 행렬과 입력 행렬의 오차율을 계산하여 평가하는데, 사용자가 아이템에 대하여 평가한 행렬인 M(User) x N(Item) Matrix를 A라고 했을 때 협업 필터링을 이용해 예측한 행렬 A’ 와의 오차를 RMSE(Root Mean Squared Error)로 계산하여 오차율을 측정

해 성능을 평가한다.

딥러닝 기술은 이미지 인식(image recognition), 자연어 이해(natural language understanding) 및 강화 학습(reinforcement learning) 등에 적용되어 사용되고 있다.[12] 처음으로 딥러닝 기술이 적용된 추천 시스템 기법 중에는 RBM(Restricted Boltzmann Machines)[13] 으로, RBM 구조를 이용한 연구가 시행되었다. 추천 시스템에 활용되는 행렬 분해(Matrix factorization) 기술은 차원축소의 형태로 이해할 수 있는데, 그렇기 때문에 자연스럽게 차원축소에 사용되는 최신 딥 러닝 신경망 구조(deep learning neural network architecture)인 오토인코더가 추천 시스템 방법으로 적용된 연구가 진행되었다. 이 연구에서는 영화 평점 데이터를 대상으로 협업 필터링 기법과 오토 인코더를 결합한 추천 시스템을 제안하였다. [14]

구글(Google)의 동영상 스트리밍 웹 사이트인 유튜브(Youtube)는 방대한 양의 데이터를 정교하게 추천 시스템에 활용하기 위해 심층신경망(Deep Neural Network, DNN)을 사용하여 비디오 추천 시스템에 적용하였다.[15] 그동안 수행되었던 행렬분해 기법과 달리 상대적으로 적은 양의 연구를 수행하였으며 거대한 양의 데이터를 수행하는데 DNN을 효과적으로 사용하였다고 설명한다. DNN은 비디오 추천 시스템뿐만 아니라 사용자 개인의 뉴스(News)추천 시스템 연구[16]에도 사용되었으며 다양한 분야의 추천 시스템 연구에 활용되고 있다.

B. 데이터 분석과 딥러닝(Deep learning)

딥러닝은 현재 기계 학습(machine learning)과 패턴 인식(pattern recognition) 등의 영역에서 연구가 활발히 진행되고 있다. 특히 음성 인식(speech recognition)과 컴퓨터 비전(computer vision), 자연어 처리(natural language processing) 등의 영역에서 성공적인 결과를 보여주었다.[17] 연구에서 쓰이는 데이터의 크기가 점점 커지고 양이 방대해지는 만큼 이 데이터를 다루는 데 딥러닝의 역할이 중요하게 대두되고 있으며 최근 주로 사용되는 거대하고 큰 데이터를 다루는 방법에 대한 연구가 수행되고 있다.

오늘 날 기계 학습(machine learning) 기술과 컴퓨터 시스템 환경이 발전하며 데이터를 분석하고 이를 통해 의미를 추출하는 것이 중요한 역할이 되었다.[18] 비교적 얕은 구조를 지닌 기존의 데이터 학습 방법들과 달리 딥러닝을 통한 학습은 정답 데이터가 주어지는 지도학습(supervised learning)과 정답 데이터 없이 학습 하는 비지도 학습(unsupervised learning)이 가능하여 깊어진 모델에서의 분류(classification)가 가능하다.

매일 대량의 데이터를 수집하여 분석하는 구글(Google), 페이스북(Facebook)과 같은 회사들은 딥러닝 기반 연구에 몰두하고 있다. 구글(Google)은 웹 사이트 내 번역기, 제품의 음성 인식, 거리 뷰(street view), 이미지 검색 엔진 등에서 모아진 크고 복잡한 데이터들을 딥러닝 알고리즘에 적용한다.[19] 마이크로소프트(Microsoft)도 마찬가지로 제품 내 실시간 번역기의 데이터를 딥러닝 기술에 적용하여 분석한다.[20]

본 연구에서는 기존에 딥러닝 추천 시스템에 사용되었던 2차원 행렬 데이터(2-dimensional arrays)에서 확장하여 고차원 데이터인 3차원 텐서 데이터(3-dimensional Tensor)를 통해 추천 시스템에 적용하여 데이터 항목과 추천 성능과의 관계를 분석하고자 한다.

Ⅲ. 딥러닝 모델을 이용한 추천 시스템

본 논문에서는 딥러닝 모델인 오토인코더와 딥 오토인코더를 사용하여 추천 시스템을 구현한다. 이를 위해 3장에서는 먼저 오토인코더 모델의 구조에 대하여 설명하고 딥 오토인코더와의 차이점을 살펴본다. 그리고 이를 구현하는 과정에 대하여 서술한다. 그 후에 빅 데이터 실험에서 오토인코더와 딥 오토인코더의 필요성에 대하여 분석한다.

A. 오토인코더

오토인코더는 협업 필터링(collaborative filtering)분야에서의 딥러닝 신경망(deep learning neural network architecture) 중 하나이다. 협업 필터링을 위한 오토인코더의 구조는 그림 1을 통해 확인할 수 있다.

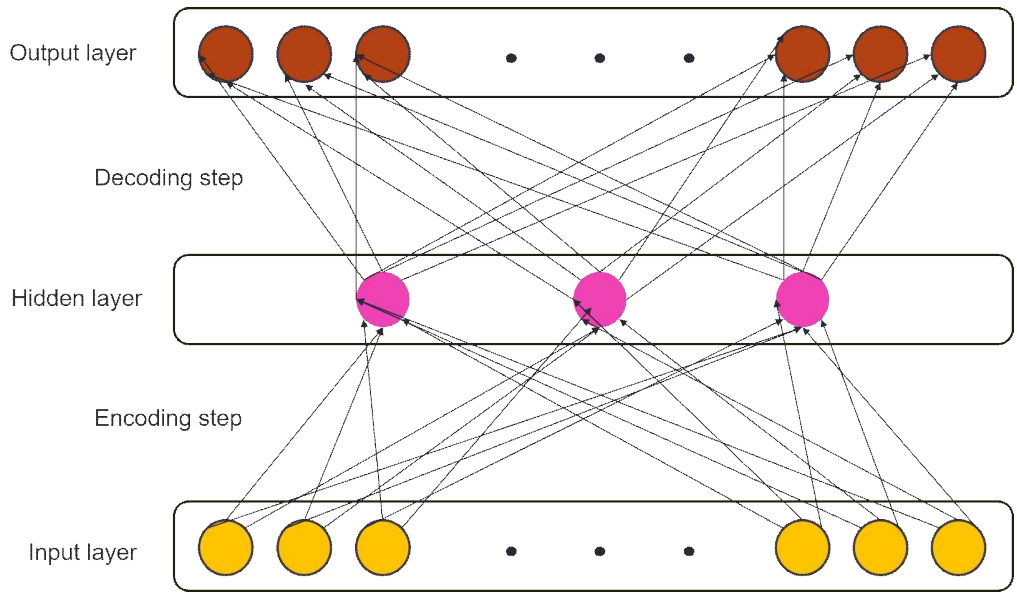

그림 1. 오토인코더(Autoencoder) 모델

그림 1은 본 연구에서 제안하는 오토인코더의 기본 모델을 표현한 것이다. 그림 1을 통해 보이듯 구조는 입력 층(input layer)과 출력 층(output layer), 그리고 그 사이에 하나의 은닉 층(one-hidden layer)으로 구성된 순방향 신경망(feedforward neural network)이다.[21] 입력 층 속의 유닛(unit)의 개수와 출력 층 속의 유닛(unit)의 수는 동일하여야 한다. 이러한 오토인코더의 특수한 구조로 인해 라벨(label)이 있는 데이터가 필요 없으며 오로지 입력 데이터만 주어지면 비지도 학습(unsupervised learning)이 가능하다. 또한 입력 층의 크기보다 은닉 층의 크기를 작게 설정한다. 오토인코더 모델이 은닉 층에서 입력 층의 데이터를 압축하여 데이터의 특징(feature)을 분석한다.

오토인코더는 데이터의 특징을 분석하여 학습한다. 입력 층에서 출력 층으로 변환하는 과정을 두 단계로 나누는데, 인코딩 단계(encoding step)와 디코딩 단계(decoding step)로 나눌 수 있다. 입력 층에서 은닉 층으로 가는 단계를 인코딩 단계(encoding step)라고 하며, 은닉 층에서 출력 층으로 가는 과정을 디코딩 단계(decoding step)라고 칭한다.

학습은 다음과 같은 계산을 통해 진행된다. 학습 단계에서 인코더는 입력 데이터의 특징을 매핑(mapping)하게 된다. 이 과정에서는 데이터에 따라 과적합(overfitting)을 줄이기 위해 가중치(weight)와 바이어스(bias)를 계산에 이용하고 필요에 따라 선형 함수(linear function)나 비선형함수(non-linear function)와 같은 활성화 함수(activation function)를 취한다.

만약 입력 데이터로 2차원 행렬 데이터인 U(사용자) x I(아이템) 행렬(Matrix)을 입력 받게 된다면 인코딩 단계와 디코딩 단계에서 입력 데이터에 가중치(weight)를 곱하고 바이어스(bias)를 더한다. 입력 층에서 은닉 층으로 인코딩된 후 은닉 층에서 출력 층으로 가는 각 단계를 거쳐 입력 데이터와 동일한 유닛(unit)을 가진 출력 데이터가 생성된다.

최근 연구에서는 시그모이드 함수(Sigmoid function), 탄젠트 함수, ReLU(Rectified Linear Unit) 같은 활성화 함수를 이용한다. 본 연구에서는 인코딩 단계에서 계산을 마친 후 시그모이드 함수(Sigmoid function)를 활성화 함수로 이용하였다.

B. 딥 오토인코더의 필요성

한 개의 은닉 층(one-hidden layer)만을 가진 오토인코더의 구조와 달리 딥 오토인코더[22, 23]는 여러 개의 은닉 층을 가질 수 있다. 층(layer)이 추가될수록 더 깊어진다고 표현하지만 은닉 층이 많아진다고 해서 더 효과적이라고는 말할 수 없기에 적절한 층의 개수와 층을 구성하는 유닛의 개수를 실험을 통해 정해야 한다.

그림 2를 통해 딥 오토인코더의 구조에 대하여 살펴본다.

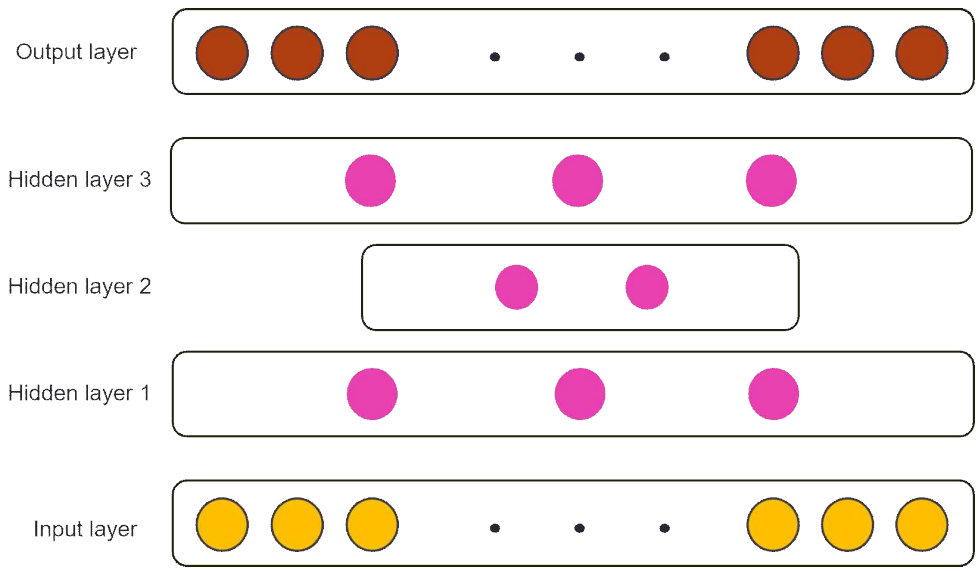

그림 2. 딥 오토인코더 모델의 구조

그림 2는 딥 오토인코더 모델의 구조를 그림으로 나타낸 것이다. 이 실험에서는 오토인코더에 은닉 층을 더 추가하여 총 3개에서 5개의 은닉 층과 입력 층, 출력 층으로 구성된 딥 오토인코더 모델을 사용하였다. 은닉 층을 구상할 때는 과적합(overfittiing)을 피하기 위하여 입력 데이터에 맞게 은닉 층의 개수와 은닉 층의 유닛(unit)수를 고려하여야 한다.

본 실험에서는 그림 2에서 나타내는 첫 번째 은닉 층인 은닉 층 1(hidden layer 1)과 마지막 은닉 층인 은닉 층 3(hidden layer 3)의 유닛 수를 같게 만들어 딥 오토인코더의 구조를 대칭적으로 구성한 실험과, 유닛 수를 다르게 하여 비대칭적으로 구성한 실험 두 가지 모두 수행하여 보았다. 은닉 층이 5개로 구성됐을 때도 마찬가지로 진행하였다. 학습 훈련 횟수(train epoch)는 500번에서 1000번 수행 하였으며, GPU환경에서 수행하여 학습 시간을 줄였다.

또한 오토인코더와 딥 오토인코더 실험에서 각 인코딩 단계(encoding step)를 거칠 때마다 가중치(weight)를 곱하고 바이어스(bias)를 더해 계산했으며, 인코딩 단계에서 시그모이드(Sigmoid) 함수를 활성화 함수(activation function)로 사용하였다. 그리고 입력 데이터를 통해 재구성(recreation)한 벡터인 예측 벡터(prediction vector)와 실제 입력 벡터의 오차율인 RMSE(Root Mean Squared Error)를 통해 손실(loss)을 계산하여 실험 결과를 비교하였다.

표 1. 2차원 행렬 데이터에서 오토인코더와 딥 오토인코더 결과 비교

Model	Layer(Unit)	Test loss(RMSE)
Autoencoder	1(500)	0.97
Deep Autoencoder	3(500,800,1000)	0.92
	5(1000,500,250,500,1000)	0.99

표 1은 2차원 영화 평점 데이터를 통해 오토인코더와 딥 오토인코더로 구현한 실험 결과를 비교한 표이다. 2차원 행렬 데이터인 무비렌즈(MovieLens)[24] 데이터를 통해 영화 추천 시스템을 구현하였을 때, 은닉 층이 1개인 오토인코더의 오차율인 RMSE가 0.974인 것에 반해 은닉 층이 3개이고 각 층이 500, 800, 1000개의 유닛으로 구성된 딥 오토인코더의 오차율인 RMSE가 0.924로, 동일한 데이터에 대해 낮아진 오차율을 확인하였다. 또한 은닉 층이 5개인 구조에서보다 은닉 층이 3개인 구조에서 실험의 오차율이 낮아 은닉 층이 3개일 때가 은닉 층이 5개일 때보다 효율적이라고 분석하였다. 딥 오토인코더는 표 1에 나타난 구조 외에도 여러 가지 방법을 통해 수행하였으며 2차원 행렬 데이터 실험에 사용한 대표적인 구조를 표 1에 기재하였다.

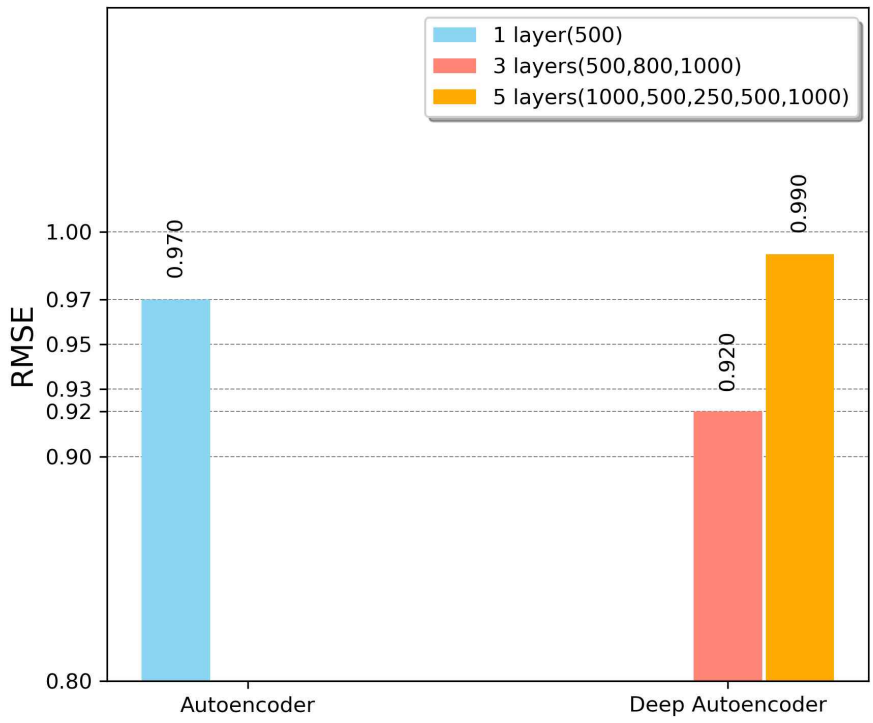

그림 3. 2차원 행렬 데이터에서 오토인코더와 딥 오토인코더 비교 그래프

그림 3은 표 1의 실험 결과를 그래프로 나타낸 것이다. 오차율이 낮을수록 실험을 통해 재구성(recreation)한 예측 벡터와 실제 입력 데이터와의 차이가 낮은 것으로 판단한다. 그림 3을 통해서도 은닉 층이 5개인 딥 오토인코더보다 3개의 은닉 층으로 구성된 딥 오토인코더가 실험에서 효과적이었음을 알 수 있다. 따라서 본 실험에서는 2차원 영화 평점 데이터에서 은닉 층이 3개이고 유닛이 500, 800, 1000으로 구성된 구조를 통해 딥 오토인코더에서 오차율이 가장 낮은 것을 확인하였다.

Ⅳ. 구현 및 실험

4장에서는 실험을 수행한 시스템 환경에 대하여 자세히 서술한 후 실험에 사용한 데이터 및 실험 방법에 대하여 구체적으로 설명한다. 그리고서 실험 결과를 확인하고 5장으로 넘어가 연구 결과에 대하여 논의한다.

A. 시스템 환경

본 연구의 실험 시스템 구현 환경은 표 2와 같다. 표 2에서 보이듯 PC Intel core i7-6700k CPU와 그래픽 카드(Graphic card)는 Nvidia GeForce GTX 1070을 사용하였다. 실험은 GPU 환경에서 구글의 오픈소스 딥러닝 프레임워크(open source deep learning framework)인 텐서플로우(Tensorflow)를 사용하여 수행하였다. 운영체제는 Ubuntu 16.04 64bit이며, 텐서플로우 Tensorflow 1.8.0 환경에서, 그리고 개발언어는 파이썬(Python3)을 사용하여 실험하였다.

표 2. 실험 시스템 구현 환경

운영체제(OS)	Ubuntu 16.04 64bit
CPU	Intel core i7-6700k CPU
Graphic Card	Nvidia GeForce GTX 1070
개발 언어 및 환경	Python3, Tensorflow 1.8.0

B. 실험 데이터

본 연구에서는 실험 데이터로 2차원 행렬 데이터와 3차원 텐서(Tensor) 데이터를 사용하였다. [25,26] 먼저 2차원 행렬 데이터는 무비렌즈(MovieLens)[24] 데이터로 영화 평점 데이터이다. 6,040명의 사용자들이 약 3,900가지의 영화들을 점수로 측정한 데이터인데 중요한 점은 사용자들이 모든 영화들을 다 보고 평가할 수 없기 때문에 값을 매기지 않은 값이 존재한다. 이를 0점, 즉 Zero data라고 한다.

그리고 사용자가 시청한 영화에 점수를 준 데이터들은 0이 아닌 1점에서 5점까지의 숫자로 Non-Zero data라고 한다. 행렬 내에서 평점이 0인 제로 데이터(zero data)들은 사용자가 값을 매기지 않은, 아직 시청하지 않은 영화이므로 사용자가 어떤 점수를 줄지 모르기 때문에 입력 데이터로 사용하지 않고 0이 아닌 값인 Non-zero 값만을 입력받는다. 즉 학습에 사용하는 데이터들은 모두 사용자가 점수를 준 값인 Non-zero 값이다.

2차원 행렬 데이터 실험에서는 영화에 대한 사용자들의 점수인 평점데이터 집합(rating dataset)을 취하여 입력 데이터로 사용한다. 이를 직관적으로 나타내면 표 3으로 나타낼 수 있다. 표 3에서 알 수 있듯이 사용자 1(User 1)은 영화 1(Movie 1)과 영화 4(Movie 4)는 시청 후 각각 1점과 5점을 주었지만 영화 3, 4, 5에 대해서는 평점을 주지 않았다는 것을 의미한다.

오토인코더는 비지도 학습 방법이기 때문에 입력 데이터를 학습 데이터(training data)와 학습 성능의 검증을 위한 테스트 데이터(test data)로 나눈다. 전체의 90%를 학습 데이터(training data)로 두었으며 나머지 10%를 테스트 데이터(test data)로 검증하였다. 실제로 파이썬(Python)내에서 실험에 입력 데이터로 사용한 2차원 행렬 데이터는 그림 4를 통해 확인할 수 있다.

표 3. 2차원 영화 평점 행렬 데이터의 예

User/ Movie	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	…	Movie 3952
User 1	1	0	0	5	0	…	2
User 2	0	2	1	3	4	…	1
User 3	5	0	2	0	2	…	0
User 4	4	1	1	4	3	…	4
User 5	1	3	2	5	0	…	5
…	…	…	…	…	…	…	…
User 6040	4	3	0	0	1	…	2

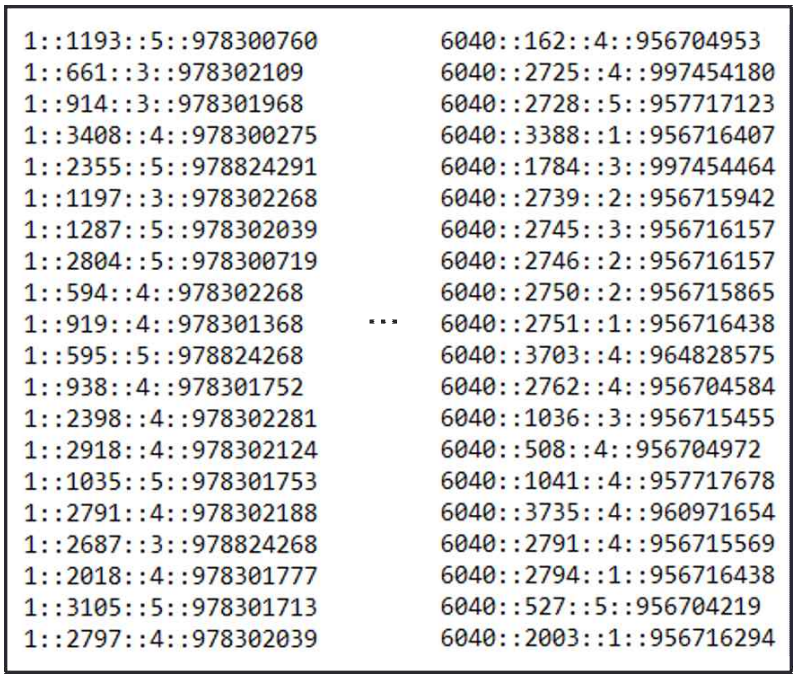

그림 4. 2차원 영화 평점 행렬 입력 데이터

그림 4를 분석하면, 한 행(row)내에서 첫 번째 항목(column)은 사용자를 나타낸다. 1은 사용자 6,040명 중 1번을 나타내며, 두 번째 항목은 영화를 나타낸다. 그리고 세 번째 항목은 사용자가 해당 영화에 대해 점수를 준 값으로 평점을 의미한다. 마지막 항목은 Time_stamp 값으로 실험에서는 사용하지 않기에 처리하지 않는다.

딥 오토인코더 추천 시스템 실험에서 사용된 3차원 데이터는 텐서 (BigTensor) 데이터 [25,26]로, 영화 평점 데이터인 무비렌즈(MovieLens)데이터[24]와 레스토랑 추천 데이터인 Yelp(YELP)데이터[27]를 사용하였다. 그림 5, 그림 6을 통해 실제 딥 오토인코더 모델에 입력데이터로서 사용한 3차원 텐서 데이터를 확인할 수 있다.

1	121	1	5.0	71567	1753	22	3.0
1	184	1	5.0	71567	1793	22	3.0
1	229	1	5.0	71567	1826	22	2.0
1	290	1	5.0	71567	1834	22	4.0
1	314	1	5.0	71567	1837	22	4.0
1	326	1	5.0	71567	1899	22	1.0
1	352	1	5.0	71567	1900	22	1.0
1	353	1	5.0	... 71567	1901	22	1.0
1	359	1	5.0	71567	1902	22	1.0
1	361	1	5.0	71567	1903	22	1.0
1	367	1	5.0	71567	1929	22	3.0
1	374	1	5.0	71567	1945	22	5.0
1	417	1	5.0	71567	2024	22	1.0
1	463	1	5.0	71567	2043	22	2.0
1	477	1	5.0	71567	2211	22	5.0
1	517	1	5.0	71567	2255	22	2.0
1	536	1	5.0	71567	2301	22	2.0

그림 5. 3차원 영화 평점 텐서 입력 데이터

1	1	1	3	500	39	31	4
2	2	2	5	3164	2269	9	4
3	3	3	3	18468	2967	56	5
4	4	4	4	1301	2388	54	3
5	5	5	1	20790	2942	47	5
6	6	6	4	76	1289	37	5
7	7	7	2	208	283	29	3
8	8	8	3	70813	1338	63	5
9	9	9	5	5319	7502	9	2
10	10	10	5	10502	1018	28	5
11	11	11	3	5893	1193	35	5
12	12	12	5	41056	401	60	3
13	13	11	1	70814	2402	70	5
14	14	13	2	70815	5578	13	1
15	15	14	4	14	1489	29	2
16	16	15	5	7695	10507	16	4
17	17	16	5	806	316	67	4
18	18	17	4	70816	42	60	1
19	19	1	5	70817	10644	36	1

그림 6. 3차원 레스토랑 평점 텐서 입력 데이터

그림 5과 그림 6에서 볼 수 있듯이 3차원 텐서 데이터에는 2차원 행렬 데이터에 없는 열(column)이 추가되어있다. 3차원 텐서 데이터는 무비렌즈(MovieLens) 데이터와 Yelp(YELP) 데이터 모두 사용자(User), 아이템 (Item), 시간(Year-Month), 그리고 평점으로 구성되어 있다. 우리는 이 행렬 데이터와 텐서 데이터 모두에서 0이 아닌 평점 값들인 Non-Zero data만을 학습하여 예측 하고자 한다.

본 실험에서는 3차원 텐서 데이터를 이용하여 시간과 사용자가 평점에 미치는 영향, 그리고 시간과 아이템이 평점에 미치는 영향을 알아보고 2차원 행렬 데이터를 분석하는 것에서 연장하는 것을 목적으로 한다.

C. 실험 방법 및 결과

다음으로는 실험 방법에 대하여 설명한다. 정답 라벨(label)이 필요하지 않은 비지도 학습(unsupervised learning)인 만큼, 입력 데이터들의 90%는 학습 데이터(training data) 로 학습시키고, 나머지 10%를 실험을 검증하기 위한 테스트 데이터(test data)로 두었다.

먼저, 2차원 영화 평점 데이터로 오토인코더 실험을 수행하였다. 아이템 기반(item-based) [14] 추천 방법을 사용하였으며 아이템(item)은 데이터에서 영화를 의미한다. 데이터 내 영화의 수는 3,952개이기에 가시적인 뉴런(neuron), 즉 유닛(unit)의 수의 값은 3,952 이다. 입력 뉴런(neuron)의 값과 출력 뉴런(neuron)의 값은 동일하다. 은닉 층(hidden layer)은 1개이고 은닉 층 속 뉴런(neuron) 값은 500으로 실험하였을 때 오차율이 가장 낮았다. 또한 Learning late는 0.005로 실행하였다.

모델의 구조에서 가중치(weight)와 바이어스(bias)에 대한 커널 초기화(kernel initialization)가 설정된다. 인코딩 단계(encoding step)단계에서 가중치(weight)와 바이어스(bias)의 초기 값은 0.0으로 두고 가중치(weight)의 평균(mean)은 0.0, 표준편차(stddev)는 0.05로 정규분포(normal distribution)하였고, 활성화 함수(activation function)는 시그모이드 함수(Sigmoid function)로 실험하였을 때 결과가 가장 좋았다. 인코딩 단계에서 가중치와 바이어스의 계산 후에 시그모이드 함수(Sigmoid function)(1) 를 취하여 디코딩 단계로 가게 된다. 식 (1)은 시그모이드 함수를 나타낸다.

$$f(x) = \frac{1}{1+e^{-x}} \quad \cdots \quad (1)$$

그림 7과 그림 8은 각각 2차원과 3차원 영화 평점 데이터 실험에 사용된 오토인코더 구조이다. 입력 층에서 은닉 층을 거쳐 출력 층으로 가며 입력 데이터 행렬이 예측 데이터 행렬로 재구성(reconstruction)된다. 이 예측 데이터 행렬과 입력 데이터 행렬 사이의 오차를 구하여 성능을 평가한다.

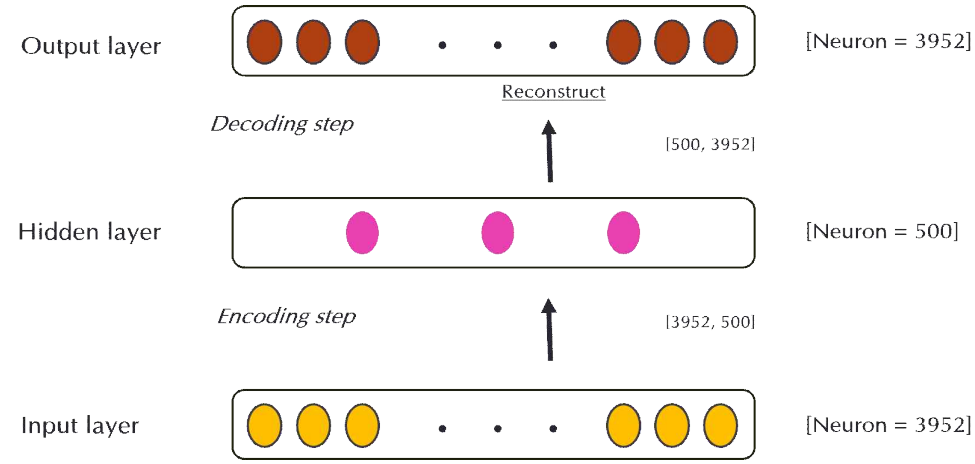

그림 7. 2차원 데이터 실험에 사용된 오토인코더

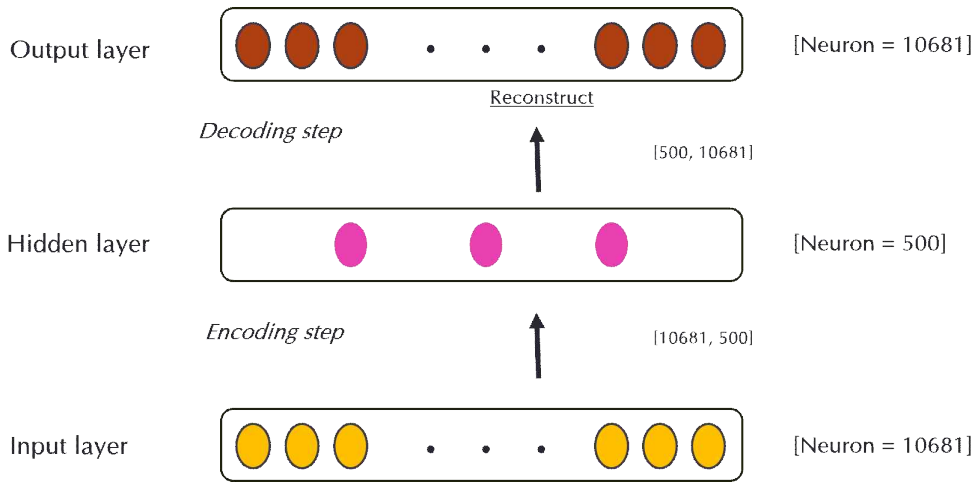

그림 8. 3차원 데이터 실험에 사용된 오토인코더

2차원 행렬 영화 평점 데이터에서 오토인코더를 통해 아이템 기반(item-based) 추천 시스템을 구현하였다. 입력 데이터를 재구성(reconstruction)하여 사용자(User)와 영화(Item)관계에 따른 User, Item, Rating으로 이루어진 예측 벡터를 생성하였고 예측 벡터와 실제 영화 평점 데이터와의 오차율(RMSE)을 확인하였다.

결과적으로 2차원 영화 평점 데이터를 사용한 오토인코더에서의 최적의 결과는 뉴런이 500일 때의 오차율인 0.974로 분석하였다. 은닉 층 속의 뉴런(neuron)이 128일 때의 테스트 손실(test loss), 즉 오차율(RMSE)은 0.992로 뉴런이 500일 때의 오차율인 0.974보다 높았다.

다음으로 3차원 텐서 영화 평점 데이터로 오토인코더를 구현한 실험에 대하여 살펴본다. 3차원 텐서 데이터는 기존의 데이터에서 추가된 열(column)인 시간(Year-Month)축을 통해 두 가지 실험을 하였다. 첫 번째로, 아이템 기반(Item-based)의 시간(Year-Month)과 영화(Item), 평점(Rating) 벡터를 예측하였다. 두 번째로는 사용자 기반(user-based)의 시간(Year-Month)과 사용자(User), 평점(Rating) 예측 벡터를 구현하였다. 그림 9는 아이템 기반(item-based)의 시간(Year-Month)과 영화(Item)의 관계를 통해 평점을 예측하는 오토인코더 모델이다. 따라서 그림 9의 입력 층과 출력 층의 뉴런의 수는 10,681로 3차원 영화 평점 데이터 내 아이템(Item)의 수와 같다.

아이템 기반(item-based) 방법으로 시간(Year-Month)과 영화(Item) 벡터를 가지고 실험하였을 때 오차율은 0.90이다. 반면 사용자 기반(user-based)방법으로 시간(Year-Month)과 사용자(User) 벡터를 이용해 실험하였을 때의 오차율은 0.44로, 더 낮은 것으로 확인하였다. 표 4를 통해 실험 결과를 비교하였다.

표 4. 오토인코더 구현 실험 결과

Data	Model	Method	Test loss (RMSE)
2-dimensional MovieLens	Autoencoder	Item-based(User-Item)	0.97
3-dimensional MovieLens	Autoencoder	Item-based(YM-Item)	0.9
		User-based(YM-User)	0.44

딥 오토인코더 실험에서는 먼저 2차원 행렬 데이터를 이용한 딥 오토인코더 실험 결과에 대하여 살펴본다. 오토인코더에서와 동일한 영화 평점 데이터를 사용하였으며, [24] 마찬가지로 아이템 기반(item-based)방법으로 사용자와 영화, 평점 벡터를 이용해 예측 벡터를 구하여 오차율을 계산하였다. 실험에서는 비대칭 구조의 딥 오토인코더와 대칭적 구조의 딥 오토인코더를 사용하였는데 그림 9와 그림 10을 통하여 이를 나타내었다. 대칭적 구조의 딥 오토인코더란 첫 번째 은닉 층과 마지막 은닉 층의 뉴런의 개수가 동일한 것은 물론 은닉 층의 구조가 대칭적인 것을 의미한다. 비대칭적 딥 오토인코더는 은닉 층의 뉴런의 개수가 대칭적이지 않아 은닉 층의 구조가 비대칭적으로 구성된 딥 오토인코더이다.

본 연구에서는 효과적인 구조를 찾기 위하여 은닉 층의 구조를 다르게 하여 연구를 진행하였다.

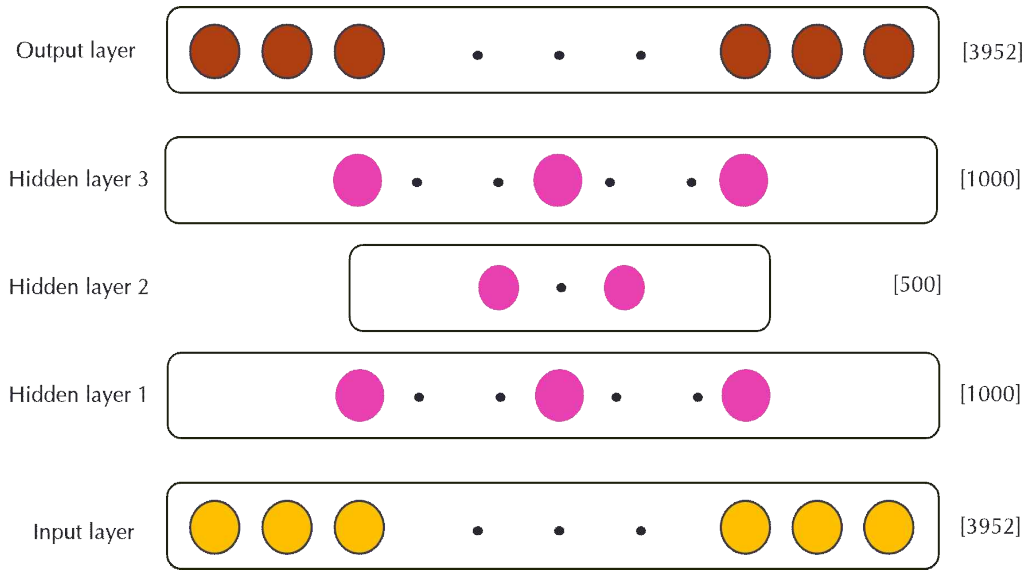

그림 9. 은닉 층이 3개인 대칭적 딥 오토인코더

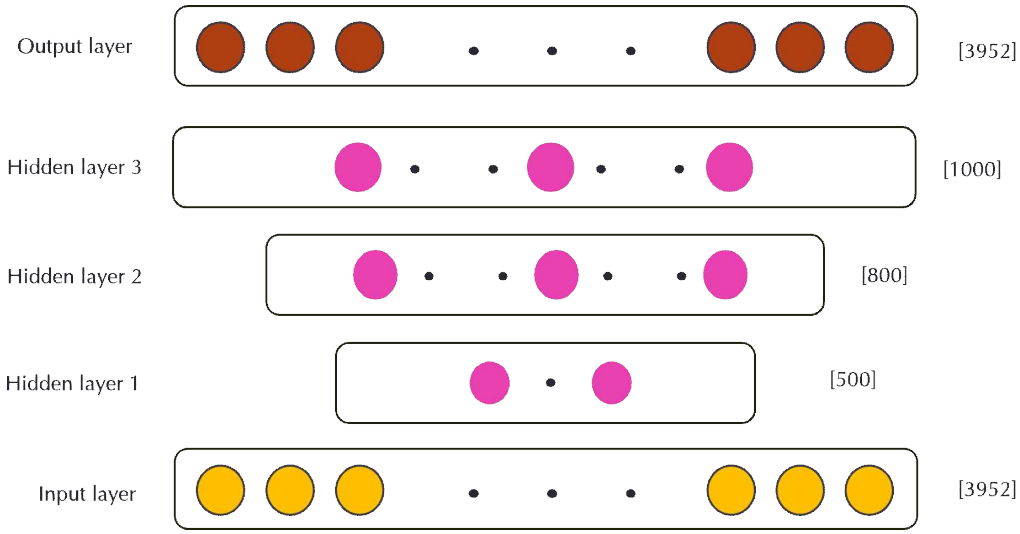

그림 10. 은닉 층이 3개인 비대칭적 딥 오토인코더

그림 9와 그림 10과 같이 은닉 층이 3개인 대칭적 딥 오토인코더 구조와 비대칭적 모델을 구현하여 실험하였다. 2차원 영화 평점 데이터를 이용한 아이템 기반(Item-based) 학습 방법으로 입력 층과 출력 층의 뉴런의 개수는 동일하게 3,952개이며 은닉 층의 개수를 조정하여 구조를 다르게 하여 실험하였다. 학습 횟수는 500번에서 1000번 수행하였으며 실험 결과, 영화 평점 데이터는 비대칭적 딥 오토인코더 모델에서 더 효과적으로 나타났다. 은닉 층 3개(뉴런 1000, 500, 1000)로 구성된 대칭적 딥 오토인코더에서의 RMSE는 0.96이었고, 비대칭적 딥 오토인코더(뉴런 500, 800, 1000)에서의 RMSE값은 0.924로 확인하였다. 세부적인 은닉 층의 구상과정은 데이터의 특성에 맞게 고려하여야 한다는 것을 알 수 있다.

딥 오토인코더 실험에서 3차원 텐서(Tensor) 데이터[26]은 영화 평점 데이터(MovieLens)[24]와 레스토랑 평점 데이터(YELP)[27]를 사용하였다. 영화 평점 데이터와 레스토랑 평점 데이터를 사용하여 사용자 기반의(user-based) 시간(Year-Month)과 사용자(User), 평점(Rating) 벡터를 이용한 실험과 아이템 기반의(item-based) 시간(Year-Month)과 아이템(Item), 평점(Rating) 벡터를 이용한 실험을 수행하였다.

3차원 텐서 데이터에 사용한 딥 오토인코더의 구조는 그림 11과 그림 12를 통해 확인한다.

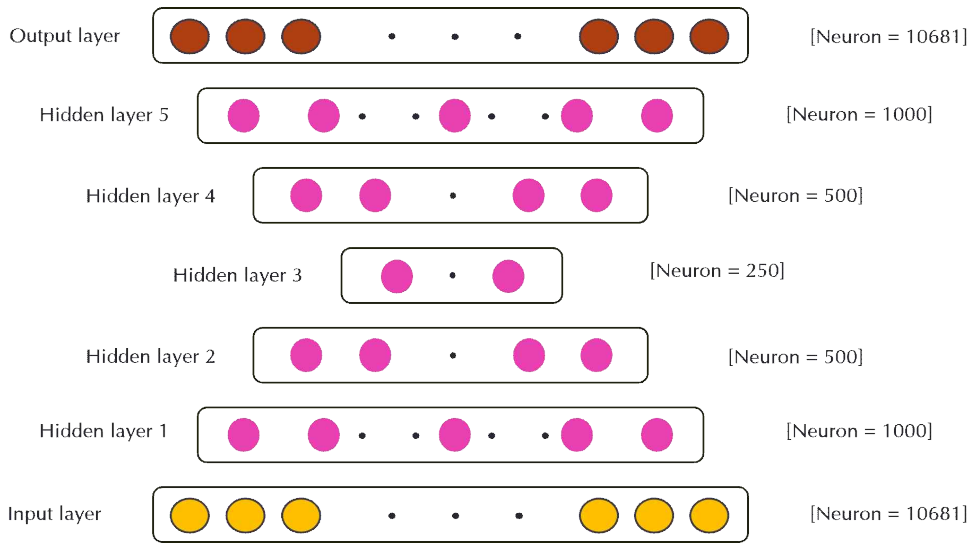

그림 11. 은닉 층이 5개인 대칭적 딥 오토인코더

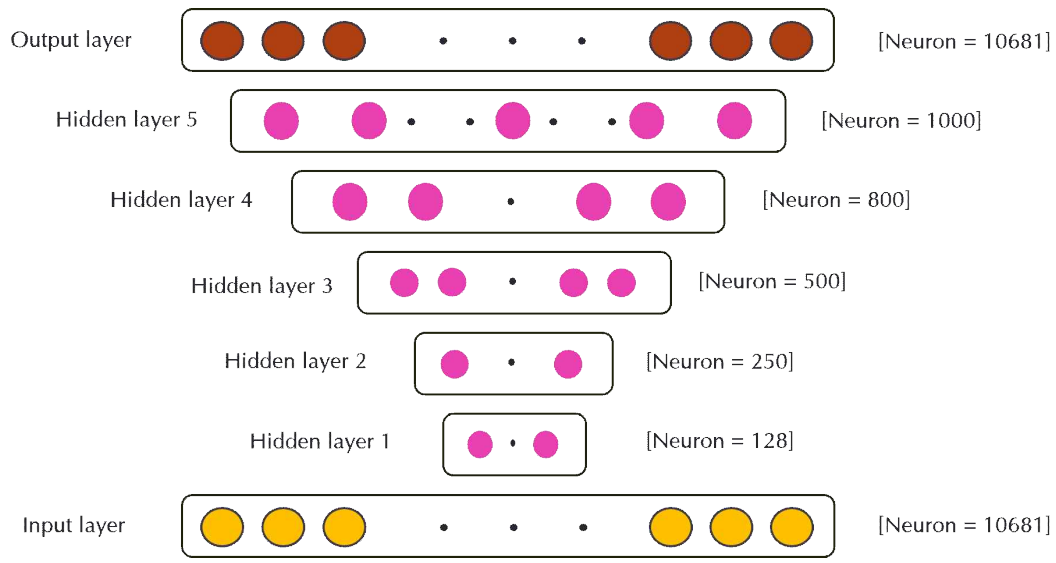

그림 12. 은닉 층이 5개인 비대칭적 딥 오토인코더

그림 11과 그림 12는 3차원 영화 평점 데이터에서 사용한 딥 오토인코더의 구조이다. 아이템 기반(item-based)방법으로 실행하였을 때의 구조이기에 입력 층(input layer)과 출력 층(output layer)의 뉴런의 개수는 동일하게 10,681로 구성된다. 딥 오토인코더를 이용한 최적 실험 결과는 표 5를 통해 나타내었다.

표 5. 딥 오토인코더 최적 실험 결과

Data	Model	Layer(Unit)	Method	Test loss(RMSE)
2-dimensional MovieLens	Deep Autoencoder	3(1000,500,1000)	Item-based (User-Item)	0.96
		3(500,500,1000)		0.94
		3(500,800,1000)		0.92
3-dimensional MovieLens	Deep Autoencoder	3(512,256,512)	Item-based (YM-Item)	0.73
		3(256,128,256)	User-based (YM-User)	0.41
3-dimensional YELP	Deep Autoencoder	3(500,800,1000)	Item-based (YM-Item)	1.17
		5(1000,500,250, 500,1000)	User-based (YM-User)	1.58

오토인코더와 딥 오토인코더 실험을 수행할 때 과적합이 일어나는 것을 주의해야 한다. 입력 데이터에 따른 모델 구조로 인해 과적합(overfitting)이 발생할 수 있다. 그림 13은 은닉 층(hidden layer)이 4개인 딥 오토인코더에서 2차원 행렬 데이터를 처리할 때 발생한 과적합의 예이다. 그림 13은 학습 반복 횟수(train epoch)당 학습 손실(train loss)과, 테스트 손실(test loss)을 비교한 그래프이다. 손실(loss)은 RMSE(Root Mean Squared Error)를 이용해 계산하였다.

이때의 실험 데이터는 2차원 행렬 영화 평점 데이터이고, 각 (500, 250, 128, 250)개의 유닛(unit)을 가진 4개의 은닉 층으로 구성 된 딥 오토인코더 구조이다. 가중치(weight)와 바이어스(bias)의 초기 값은 0.0으로 설정하였고 가중치(weight)의 평균(mean)은 0.0, 표준편차(stddev)는 0.07로 정규분포(normal distribution)하였다. 학습이 20번 진행될 때까지는 학습 손실(train loss)과 테스트 손실(test loss)이 함께 감소하며 학습이 진행되다가 25번 단계로 접어들었을 때는 학습 손실(train loss)은 줄어들지만 테스트 손실(test loss)은 오히려 증가하며 데이터에 대하여 과적합(overfitting)된 모습을 볼 수 있다. 이를 통해 과적합의 예시를 확인할 수 있다.

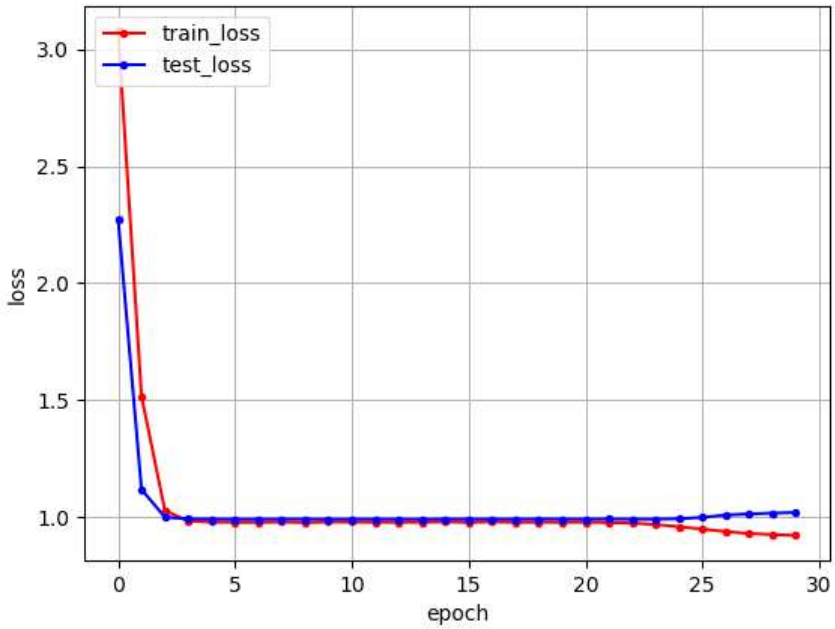

그림 13. 딥 오토인코더에서 발생한 과적합의 예

D. 오토인코더와 딥 오토인코더의 결과 비교

앞서 실험 결과를 통해 동일한 데이터에서 오토인코더와 딥 오토인코더의 오차율의 차이를 확인하였다. 실험 결과를 바탕으로 오토인코더와 딥 오토인코더의 결과에 대하여 분석하고 비교하고 실험을 통해 알게 된 데이터 항목 간의 연관성에 관하여 논의하고자 한다. 먼저 2차원 행렬 데이터인 영화 평점 데이터에 대하여 비교한다. 표 6은 오토인코더와 딥 오토인코더의 총 실험 결과를 비교한 표이다.

오토인코더와 딥 오토인코더 모두 동일한 데이터에서는 Weight, Bias, Learning rate, Batch size, Train epoch 등의 실험 환경을 동일하게 구성하였다. 딥 오토인코더는 은닉 층이 3개일 때와 5개일 때로 구현하여 실험 했으며 가장 실험 결과가 좋을 경우를 찾는 것을 목표로 하였다. 실험 결과를 비교함으로써, 결과적으로 은닉 층이 추가된 딥 오토인코더로 수행했을 때의 결과가 오토인코더 만으로 실험했을 때의 결과보다 성능이 높은 것으로 판단할 수 있으며 최적의 은닉 층의 구성 환경을 찾아 딥 오토인코더의 필요성을 확인하였다. 그리고 표 7을 통해 알 수 있듯이, 기존의 행렬 분해(Matrix factorization) 방법과 비교하였을 때 본 연구를 통한 2차원 데이터에서의 딥 오토인코더 실험에서 오차율(RMSE)이 낮아진 결과를 확인할 수 있었다.

또한 학습을 할 때는 Train epoch를 1,000회 이상 증가 시킨다고 해서 효율이 높아지지 않으며 과적합의 오류를 피하기 위해 500회에서 1,000회 학습을 진행하였다. 또한 3차원의 텐서 데이터는 2차원 행렬 데이터보다 크기가 크기 때문에 크기가 큰 데이터일수록 학습 시 Batch size를 줄여 메모리의 오류를 줄였다.

표 6. 오토인코더와 딥 오토인코더 총 실험 결과 비교

Data	Model	Layer(Unit)	Method	Test loss (RMSE)
2-dimensional MovieLens	Autoencoder	1(500)	Item-based (User-Item)	0.97
	Deep Autoencoder	3(1000,500,1000)		0.96
		3(500,500,1000)		0.94
		3(500,800,1000)		0.92
		5(1000,500,250, 500,1000)		0.99
		5(128,250,500, 800,1000)		1.16
3-dimensional MovieLens	Autoencoder	1(500)	Item-based (YM-Item)	0.99
			User-based (YM-User)	0.44
	Deep Autoencoder	3(512, 256, 512)	Item-based (YM-Item)	0.73
		3(256,128,256)	User-based (YM-User)	0.41
		5(1000,500,250, 500,1000)	Item-based (YM-Item)	0.75
		5(1000,500,250, 500,1000)	User-based (YM-User)	0.46
3-dimensional YELP	Deep Autoencoder	3(500,800,1000)	Item-based (YM-Item)	1.17
		3(1024,512,1024)	User-based (YM-User)	1.6
		5(1000,500,250, 500,1000)	Item-based (YM-Item)	1.21
		5(1000,500,250, 500,1000)	User-based (YM-User)	1.58

표 7. 2차원 데이터에서의 모델별 실험 결과 비교

Data	Model	Test loss(RMSE)
MovieLens	Matrix Factorization	0.93
	Autoencoder	0.97
	Deep Autoencoder	0.92

표 6을 통해서 확인할 수 있듯이, 본 연구를 통해 3차원 텐서 데이터인 영화 평점 데이터와 레스토랑 평점 데이터를 사용해 시간(Year-Month)을 포함한 각 데이터 항목들과 평점과의 연관성을 알아보았다.

표 7은 2차원 영화 평점 데이터인 무비렌즈(MovieLens)에서 행렬분해(matrix factorization) 방법과 오토인코더, 딥 오토인코더 방법을 비교한 표이다. 표 7에서 딥 오토인코더는 3개의 은닉 층을 가지며 각 유닛은 500,800,1000으로 구성된 모델이다. 2차원 영화 평점 데이터에서 실험 시 가장 오차율이 낮았던 딥 오토인코더 모델로 표 6에서도 구성을 확인 가능하다. 표 7을 통해 기존의 행렬분해 방법과 비교하여 딥 오토인코더를 통한 실험에서 오차율이 가장 낮아졌음을 확인하였다.

표 6의 결과를 바탕으로 3차원 데이터에서의 시간(Year-Month) 데이터와 다른 데이터 항목들의 관계를 비교하기 위해, 딥 오토인코더를 이용한 3차원 데이터의 실험 결과를 그림 14를 통해 그래프로 표현하였다. 그림 14에서 딥 오토인코더를 이용한 3차원 영화 평점 데이터(MovieLens)와 레스토랑 평점 데이터(YELP)에서의 시간(Year-Month)과 아이템(Item), 시간(Year-Month)과 사용자(User)와의 실험 결과를 비교할 수 있다.

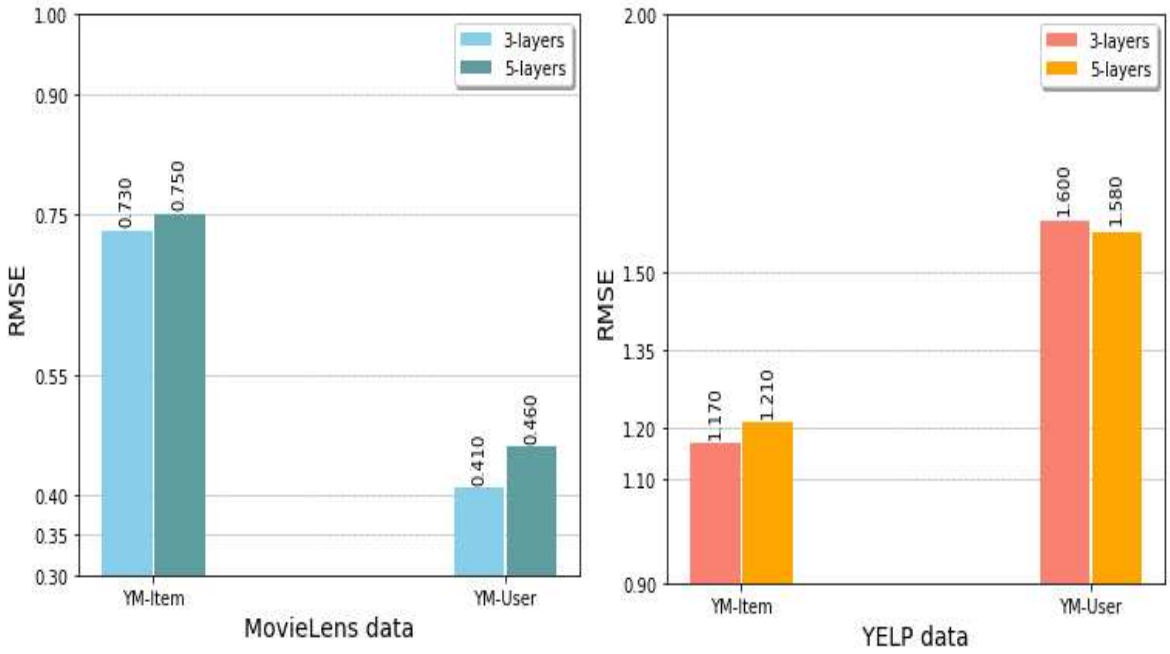

그림 14. 3차원 데이터에서의 시간 축을 고려한 실험 결과 비교

두 데이터에서 사용자 기반의(User-based) 시간(Year-Month)과 사용자(User), 평점(Rating) 벡터를 이용한 실험과 아이템 기반의(Item-based) 시간(Year-Month)과 아이템(Item), 평점(Rating) 벡터를 이용한 실험을 수행하였고 영화 평점 데이터에서는 시간(Year-Month)과 사용자(User) 벡터를 통해 실험한 결과 (0.41)가 시간(Year-Month)과 영화(Item)벡터를 통해 실험한 결과(0.73)보다 오차율이 낮게 측정되었다. 또한 영화 평점 데이터에서는 5개의 은닉 층을 사용하였을 때보다 3개의 은닉 층을 사용한 실험에서 더 낮은 오차율을 보여주었다.

반면, 레스토랑 평점 데이터에서는 시간(Year-Month)과 사용자(User) 벡터를 통해 사용자 기반(user-based)방법으로 실험한 결과(1.58)가 시간(Year-Month)과 식당

(Item)벡터를 통해 아이템 기반(item-based)방법으로 실험한 결과(1.17)보다 오차율이 높게 측정되었다. 레스토랑 평점 데이터에서 시간(Year-Month)과 아이템(Item)의 경우, 5개의 은닉 층을 사용하였을 때보다 3개의 은닉 층을 사용하였을 때의 실험 결과가 더 좋았지만 시간(Year-Month)과 사용자(User)의 경우, 3개의 은닉 층을 사용하였을 때보다 5개의 은닉 층을 사용한 실험에서 미세하게 오차율이 더 낮은 것을 보여주었다.

이를 통해 영화 추천 시스템에서는 시간과 사용자의 관계가 시간과 아이템의 관계보다 의미 있다고 분석하였고 레스토랑 추천 시스템을 구현할 때는 시간과 사용자의 관계보다 시간과 아이템 관계에서 유의미한 결과를 찾는다는 것으로 분석하였다. 그리고 은닉 층의 구성 또한 데이터에 따라 다른 구성에서 최적의 결과를 보여주었다. 따라서 추천 시스템 연구를 수행할 때는 데이터들의 특성에 맞게 각 항목들의 상관관계와 연관성을 고려하여 구성해야하며 최적의 학습 변수 및 은닉 층 구성변수 또한 실험을 통해 확인하여 데이터에 맞게 재구성해야 한다는 것을 확인할 수 있다.

Ⅴ. 결론 및 향후과제

본 장에서는 1장부터 4장까지 다룬 내용인 실험 데이터와 결과를 바탕으로 오토인코더와 딥 오토인코더의 실험 결과에 대하여 결론짓는다. 그리고 향후 연구 방향과 계획에 대하여 논의하고 논문을 마무리 한다.

추천 시스템이 전자 상거래(e-commerce) 뿐만 아니라 뉴스, 동영상 스트리밍 사이트 등 다양한 분야와 웹사이트에 사용되며 주목받았고 자연스럽게 협업 필터링(collaborative filtering) 방법과 딥러닝(deep learning)을 활용한 기술들이 연구를 통해 제안되었다. 기존 연구에서는 추천 시스템 연구에 가장 많이 쓰였던 방법인 행렬 분해(matrix factorization) 기법을 통해 상품과 사용자의 관계를 고려하여 추천하는 방법이 채택되어 사용되었다. 기존에 수행된 연구가 2차원 데이터를 가지고 데이터간의 의미를 추출하여 추천 시스템에 활용하는 것이라면, 본 연구에서는 이를 확장하여 3차원 텐서 데이터를 입력받아 더 추가된 항목이 평점에 미치는 영향에 대하여 분석하는 것을 목표로 하였다. 또한, 2차원 행렬 데이터와 3차원 행렬 데이터 모두 은닉 층의 구성에 따라 딥 오토인코더를 이용한 실험결과가 오토인코더만으로 실험한 결과보다 오차율이 낮아 추천 성능이 높다고 판단함으로써 딥 오토인코더의 필요성을 확인할 수 있었다.

본 실험은 GPU환경에서 실험이 진행된 만큼 학습 횟수(train epoch)를 1,000번 수행할 때 20분 내외로 수행되었으며 데이터의 크기가 커질수록 Batch size를 조절해 실험에서의 메모리 오류를 줄이도록 했다. 실험에서는 3차원 텐서 데이터를 입력받아 각 데이터 항목과의 연관성을 알아보기 위해 데이터를 나누는 방법으로 수행하였지만 향후 3차원 텐서 데이터를 입력받았을 때 데이터를 나누거나 쪼개지 않고 순수하게 그대로 학습시키는 방법을 고안할 필요가 있다. 2차원 행렬 데이터 이상의 고차원 데이터인 3차원 텐서 데이터를 딥러닝(deep learning)에 사용해 추천 시스템의 성능을 높이는 것을 연구 목적으로 하는 만큼 데이터 자체를 학습 전에 처리하는 과정 없이 어떠한 데이터든 간에 딥 오토인코더 모델에 사용할 수 있도록 하는 연구를 진행할 계획이다.

또한 3차원 이상의 고차원 데이터가 딥러닝 실험 과정 중에 메모리 오류 등으로 인해 학습이 중단되지 않고 수행되는 것이 중요하다고 판단하였으며 오토인코더 알고리즘 내에서 학습에 이용하는 계산을 수정 및 보완하여 오차율을 낮추는 것을 연구 목표로 진행할 계획이다. 따라서 현재 연구에서 수행한 부분을 더 심화하여 추천 시스템 연구에서 3차원 이상의 고차원 데이터 항목들 간의 연관관계를 분석하는 것을 향후 목표로 한다.

참 고 문 헌

[1] Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). “Collaborative filtering recommender systems.” In The adaptive web (pp. 291-324). Springer, Berlin, Heidelberg.

[2] Gomez-Uribe, Carlos A., and Neil Hunt. “The netflix recommender system: Algorithms, business value, and innovation.” ACM Transactions on Management Information Systems (TMIS) 6.4 (2016): 13.

[3] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295). ACM. 2001.

[4] Koren, Yehuda, Robert Bell, and Chris Volinsky. “Matrix factorization techniques for recommender systems.” Computer 8 (2009): 30-37.

[5] Lee, Joonseok, et al. “Local low-rank matrix approximation.” International Conference on Machine Learning. 2013.

[6] Shi, Yue, Martha Larson, and Alan Hanjalic. “Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges.” ACM Computing Surveys (CSUR) 47.1 (2014): 3.

[7] Ouyang, Y., Liu, W., Rong, W., & Xiong, Z. (2014, November). “Autoencoder-based collaborative filtering.” In International Conference on Neural Information Processing (pp. 284-291). Springer, Cham. 2014.

[8] Strub, Florian, Jérémie Mary, and Romaric Gaudel. “Hybrid Collaborative filtering with autoencoders.” arXiv preprint arXiv:1603.00806 (2016).

[9] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016, November). “Tensorflow: a system for large-scale machine learning.” In OSDI (Vol. 16, pp. 265-283).

[10] Adomavicius, Gediminas, and Alexander Tuzhilin. “Context-aware recommender systems.” Recommender systems handbook. Springer, Boston, MA, 2011. 217-253.

[11] Bennett, J., & Lanning, S. (2007, August). “The netflix prize.” In Proceedings of KDD cup and workshop (Vol. 2007, p. 35).
[12] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. “Deep learning.” nature 521.7553 (2015): 436. 2015.
[13] Salakhutdinov, Ruslan, Andriy Mnih, and Geoffrey Hinton. “Restricted Boltzmann machines for collaborative filtering.”, Proceedings of the 24th international conference on Machine learning. ACM, 2007.
[14] Sedhain, Suvash, et al. “Autorec: Autoencoders meet collaborative filtering.” Proceedings of the 24th International Conference on World Wide Web. ACM, 2015.
[15] Covington, P., Adams, J., & Sargin, E. (2016, September). “Deep neural networks for youtube recommendations.” In Proceedings of the 10th ACM Conference on Recommender Systems (pp. 191-198). ACM. 2016.
[16] Oh, K. J., Lee, W. J., Lim, C. G., & Choi, H. J. (2014, February). “Personalized news recommendation using classified keywords to capture user preference.” In Advanced Communication Technology (ICACT), 2014 16th International Conference on (pp. 1283-1287). IEEE. 2014.
[17] Chen, Xue-Wen, and Xiaotong Lin. “Big data deep learning: challenges and perspectives.” IEEE access 2 (2014): 514-525.
[18] Lin, J., & Kolcz, A. (2012, May). “Large-scale machine learning at twitter.” In Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (pp. 793-804). ACM. 2012.
[19] Jones, Nicola. “Computer science: The learning machines.” Nature News 505.7482 (2014): 146. 2014.
[20] Wang, Y. Y., Yu, D., Ju, Y. C., & Acero, A. (2011). “Voice search.” Spoken Language Understanding: Systems for Extracting Semantic Information from Speech, 119-146.
[21] Artem Oppermann, “Deep Autoencoders For Collaborative Filtering”, URL, https://towardsdatascience.com/

[22] Kuchaiev, Oleksii, and Boris Ginsburg. “Training deep autoencoders for collaborative filtering.” arXiv preprint arXiv:1708.01715 (2017).
[23] Strub, Florian, and Jeremie Mary. “Collaborative filtering with stacked denoising autoencoders and sparse inputs.“ NIPS workshop on machine learning for eCommerce. 2015.
[24] MovieLens, URL, http://grouplens.org/datasets/movielens/
[25] Wikipedia, “Tensor”, URL, https://en.wikipedia.org/wiki/Tensor
[26] BigTensor, URL, https://datalab.snu.ac.kr/bigtensor/
[27] Yelp, URL, http://www.yelp.com/dataset_challenge/

ABSTRACT

Recommendation system on 3-dimensional tensor using Deep Autoencoder

Park, Jina
Department of Computer Science & Engineering
The Graduate School
Ewha Womans University

The purpose of this study is to propose Autoencoder, a state-of-the-art deep learning neural network architecture for using Collaborative Filtering method on Recommendation system, and also Deep Autoencoder, extension version of Autoencoder in TensorFlow.

Matrix factorization, kind of useful Collaborative filtering methods in former days, was used the most and utilized to obtain user's taste of item, and it was used for dimensional reduction area also.

These days, as Interest in Deep learning and Data analysis is increasing, many related studies to develop performance for Collaborative filtering with Deep learning has advanced considerably.

Autoencoder is used by Recommendation system to predict ratings from users, and this paper also proposes Deep Autoencoder by adding additional hidden layers to original architecture of Autoencoder.

An Autoencoder is totally composed of 3 layers, which are Input layer, Output layer, and One-Hidden layer, so we added more hidden layers to original

architecture of Autoencoder so that we implemented Deep Autoencoder with 3 to 5 hidden layers, more deeper architecture. In this paper, therefore we make a comparison between performance of them.

This paper is aimed to raise much better performance and accuracy of Deep Autoencoder than that of Autoencoder. Several existing studies used 2-dimensional arrays as input dataset, and the dataset is made up of User, Item and Rating. They analyzed movie or restaurant dataset and found a correlation between Item-based rating and User-based rating.

In this research, we use Tensor dataset, 3-dimensional tensor, composed of User, Item, Time, and Rating by User as input dataset. After that, we propose recommendation system based User, Item and Time using Deep Autoencoder.

As a result, we figured out that Deep Autoencoder model with 3 to 5 hidden layers generalized much better result than Autoencoder and also found the correlation between on 3-dimensional dataset.