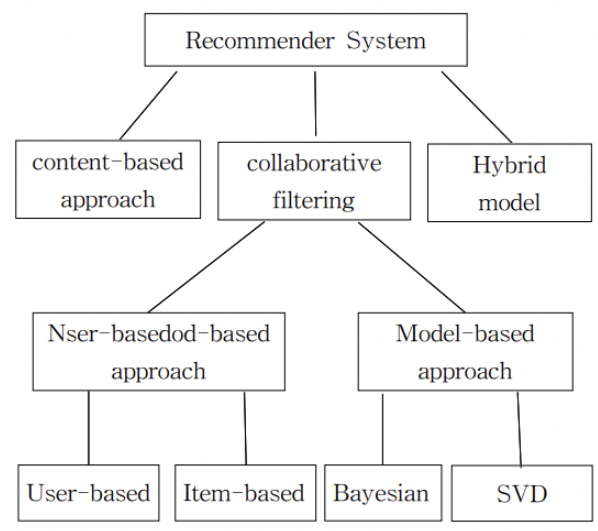
**Recommender System**

논문에 앞서 본 논문의 시작점인 추천시스템 방법론과 Matrix Factorization 추천시스템에 알아보겠다.

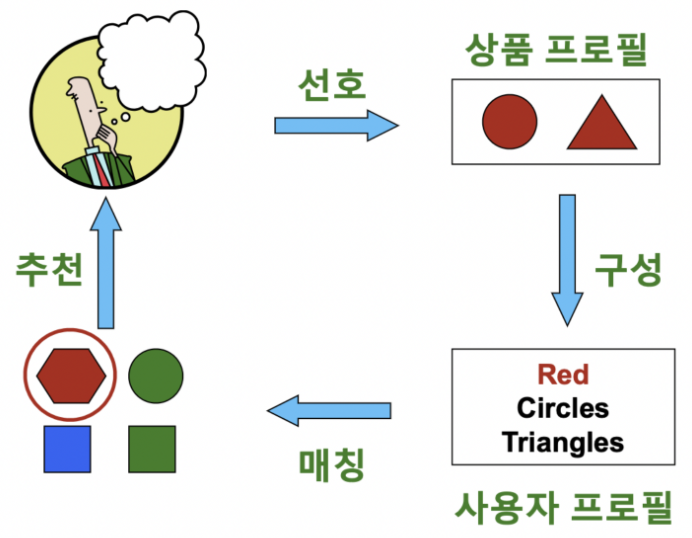


**Content-based**

컨텐츠 기반 추천시스템은 특정 아이템에 기초하여 비슷한 아이템을 추천하는

방법입니다. 아이템끼리의 유사도를 측정하며 아이템의 metadata를 사용합니다.

-> 지식그래프를 이용하여 활용가능하다.



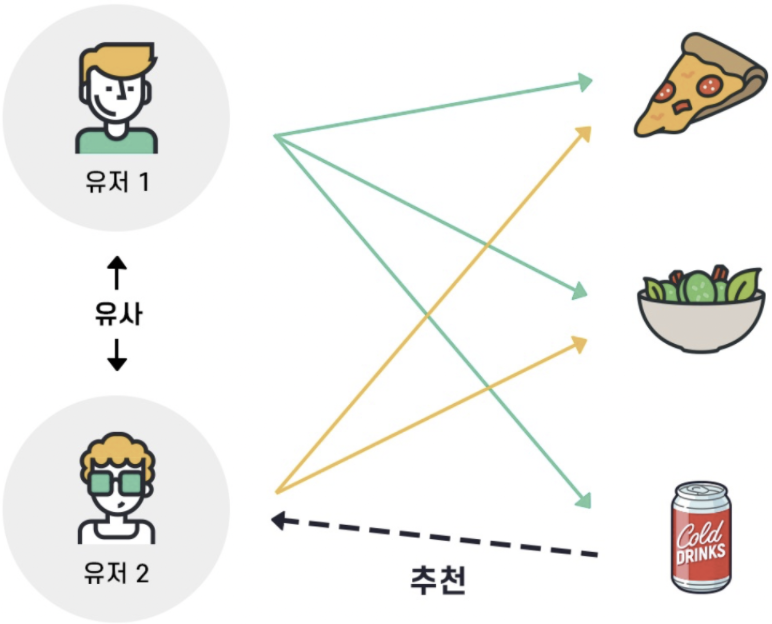
**Collaborative Filtering(CF)**

사용자 그룹이 형성되어 있고, 그들 간의 평가 점수와 선호도를 고려하여 사용자의 예측점수와 선호도가 결정되는 방법이다. 즉, 사용자와 비슷한 다른 사용자를 찾아서 그 사용자는 어떤 평가를 했는지 살펴보는 방식이다.

협업필터링방법으로는 user-based, item-based, model-based 등이 존재한다.

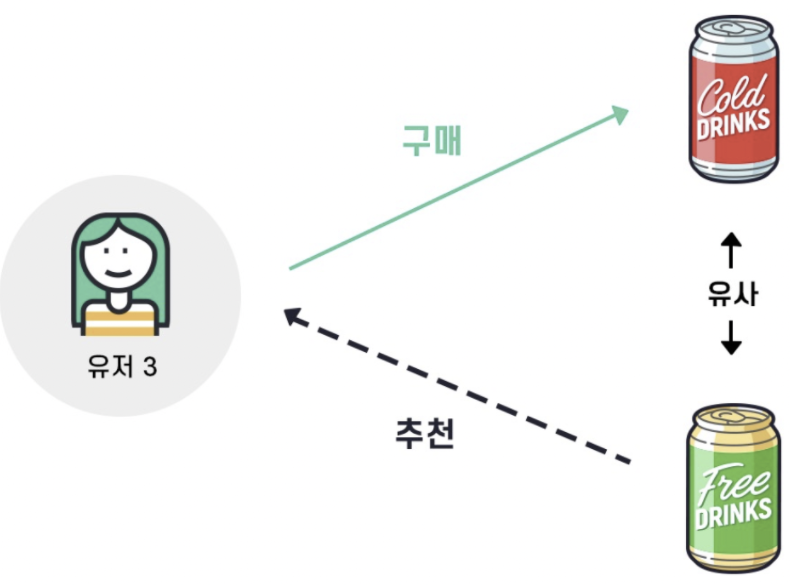
**User-based**

나와 비슷한 성향을 지닌 사용자를 기반으로 그 사람이 구매한 상품을 추천하는 방식이다.



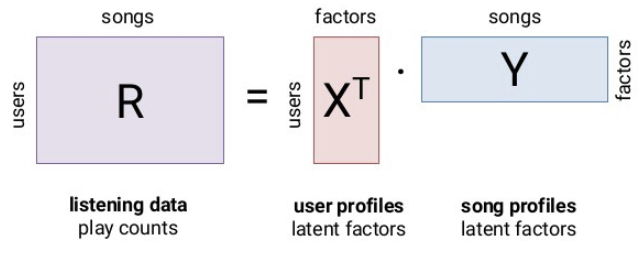
**Item-based**

내가 이전에 구매했던 아이템을 기반으로 그 상품과 유사한 다른 상품을 추천하는 방식이다.



**Matrix Factorization(MF)**

평점 행렬을 사용자와 아이템 Latent Factor 행렬로 분해하고 분해된 행렬을 다시 곱하여 예측 평점 행렬을 계산한 뒤 기존의 행렬과 예측한 행렬의 오차가 최대한 작아지도록 학습시키는 방법이다.



<https://yeomko.tistory.com/5> <matrix factorization 추천시스템>

**Graph Convolutional Matrix Completion (GCMC)**

# 

# **Idea**

* Graph 구조의 방법론을 추천 시스템 중 Matrix Factorization에 투영
* user-item rating matrix를 graph 구조로 표현
* 데이터를 matrix가 아닌 graph로 표현한 것만 제외하면, 풀고자 하는 문제는 matrix factorization과 본질적으로 동일 (추천 시스템에서의 matrix completion 문제를 **"link prediction on graph"** 관점에서 해결 - Matrix Completion as Link Prediction in Bipartite Graphs)

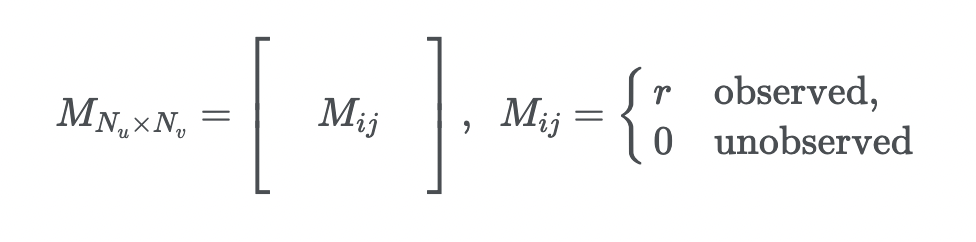
## **Intro**

### **Framework**

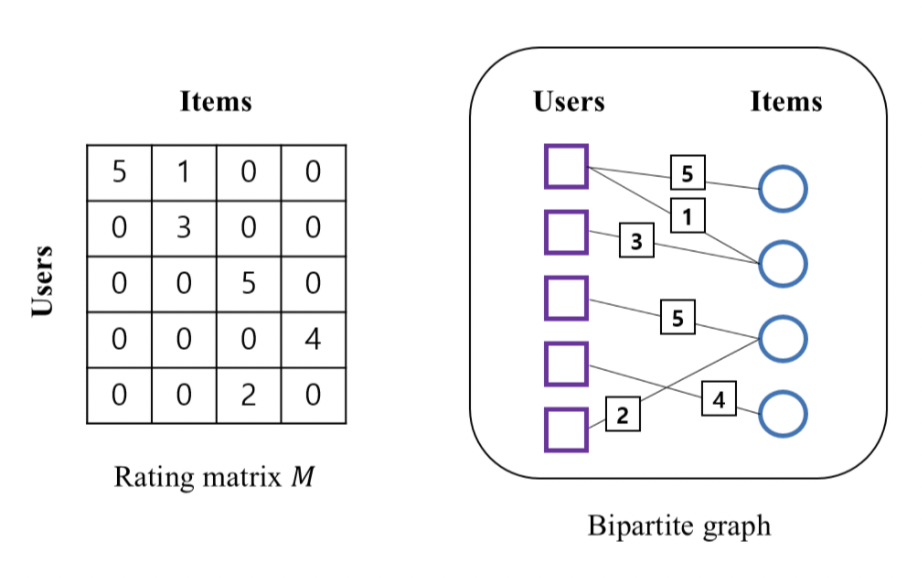
# user-item rating matrix를 user-item의 이분그래프로 표현하여 graph auto-encoder 프레임워크에 응용, 결과적으로 rating(edge, link)을 예측하는 시스템.matrix의 0점 부분을 예측한다는(채운다는) 의미에서 matrix completion이라고 할 수 있다.

### **User-item rating matrix**

user, item 수를 각각 Nu, Nv라 하고, user i가 item j에 메긴 rating을 r이라 할 때, rating matrix M은 다음과 같이 표현



### **Rating matrix 그래프 맵핑**



user와 item 각각은 node로, rating은 edge로 표현된다.

예를 들어 user1이 item1에 5점을 매긴 경우(M11=5) 그래프로 나타내면 user1, item1의 node를 연결하고, edge에 5점을 부여함.

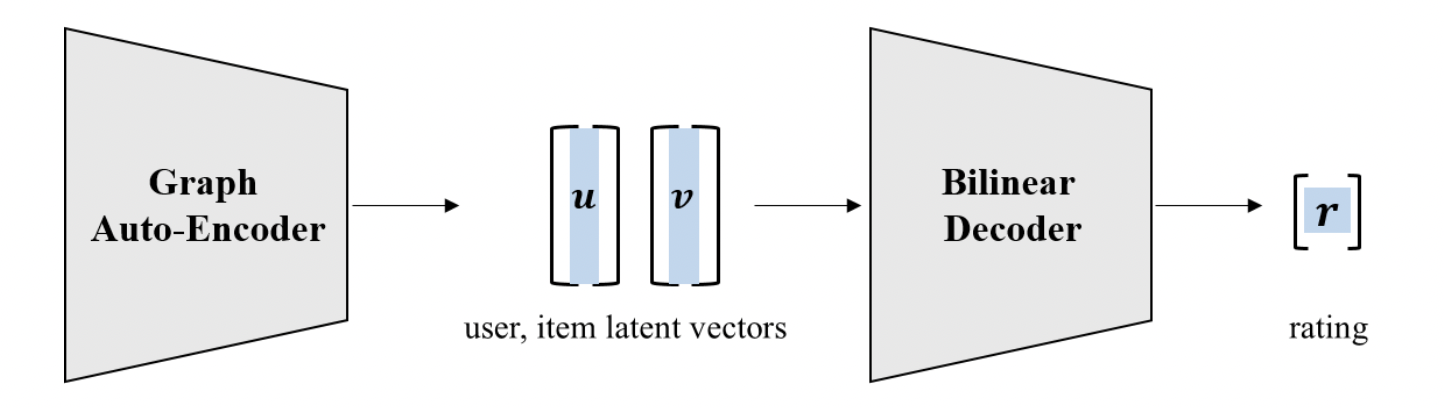
그래프로 기존의 Rating Matrix를 표현하는 방식은 다음과 같다.



W는 User와 Item의 집합을 나타내는 기호이고, E는 Edge 정보, 그리고 R은 Rating에 대한 정보를 의미한다. 각각은 Graph에서 Node, Edge, Link의 역할을 하게 된다.

## **Graph Auto Encoder**

### **Framework**

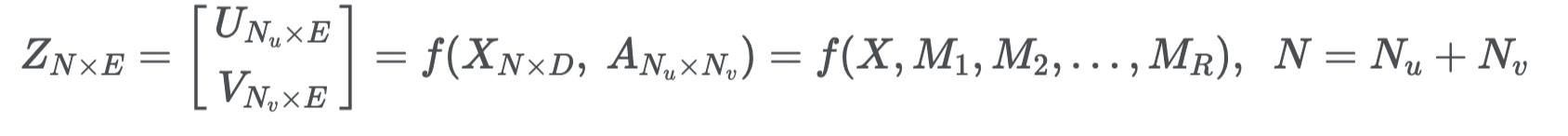


* encoder 모델: user, item 각각에 대한 latent vector를 구함(matrix factorization에서 latent vector를 구하는 것과 동일). 논문에서의 encoder모델은 ​​**graph convolutional network(gcn)**
* decoder 모델: latent vector로 원래의 rating을 예측. 일반적으로 u와 v의 내적을 사용하나 논문에서는 별도의 decoder 사용

### **Encoder**

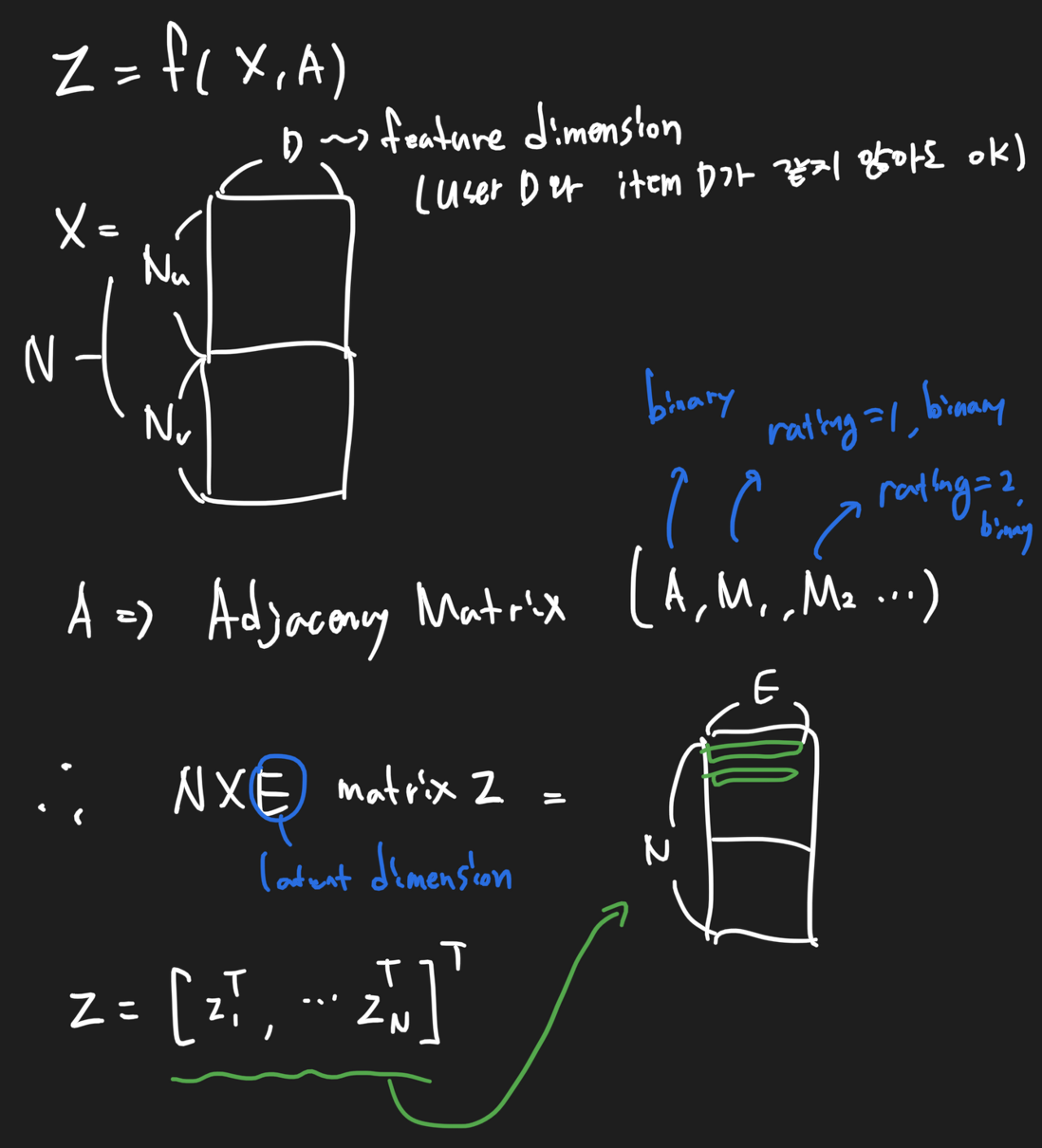
#### **a) 개념**

### encoder 모델 f(.)는 다음과 같이 정의된다. 이는 앞서 user, item각각에 대한 latent vector를 구하는 것이라 하였다.

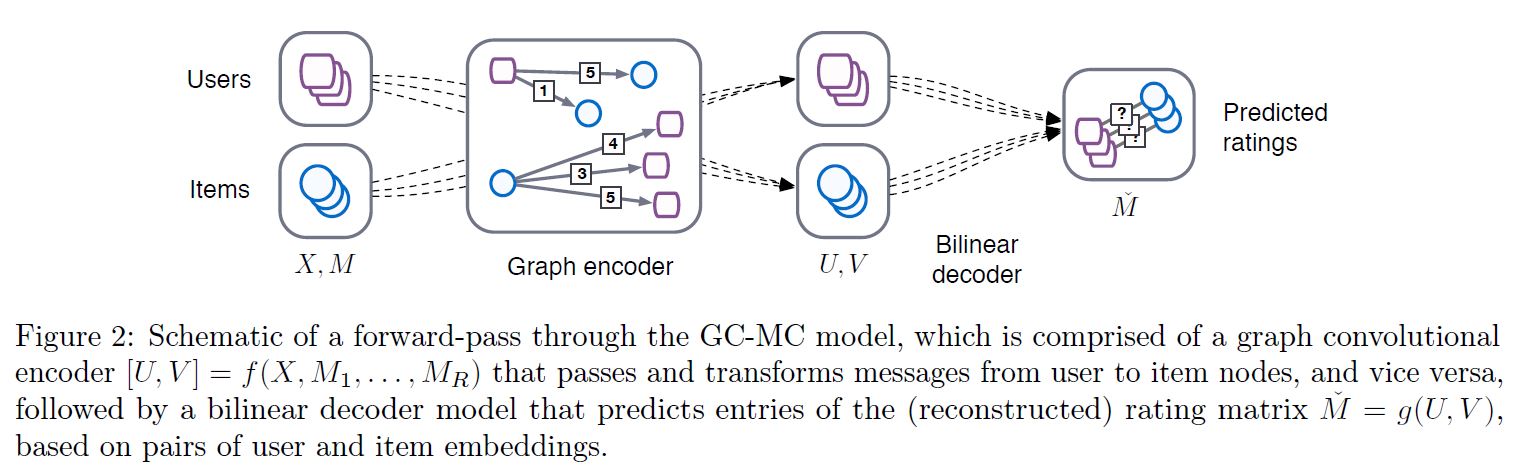


|  | 설명 | 표현 |
| --- | --- | --- |
| U, V | user latent vector, item latent vector로 이뤄진 matrix | E는 latent vector 차원 |
| XN×D | user, item 각각에 대한 dense vector로 이뤄진 feature matrix |  |
| ANu×Nv  (or Mr) | graph adjacency matrix | edge가 있는 부분은 1 없는 부분은 0  이를 각 rating에 대해 표현할 수도 있는데, r점에 대한 adjacency matrix  Mr은 Mr∈0,1Nu×Nv로 정의됨. |

정리하자면, encoder 모델은 feature matrix X와, adjacency matrix A(orMr)를 갖고 latent vector를 구하는 것이라 할 수 있다.



#### **b) Graph convolutional encoder**

논문이 제시한 encoder 모델은 graph convolutional network(gcn)이다.

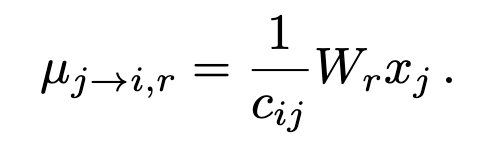
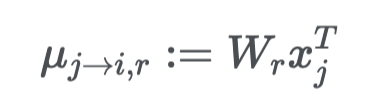
본 논문에서는 1~5 사이의 Rating을 모두 다른 채널로 나누어 학습에 사용한다. 즉, Rating 마다 별도의 채널을 사용하는 그래프 네트워크를 만든다는 것이다. 이는 파라미터의 관점에서 5개의 다른 shared parameter가 존재한다는 것을 의미하기도 한다.

CNN에서 중심점의 근방 n개 (ex : 3x3 filter)만을 convolutional 하듯이, 네트워크에서도 node의 1-depth 만큼의 연결만을 들여다본다. 그리고 그 각각의 iteration을 shared parameter로 학습하는 것이다. 이것을 **Local Graph Convolution** 이라고 하며 shared 파라미터를 Edge-type Specific Parameter라고 한다. (WR로 표기하며, 후에 나오는 hidden layer ↔ encoder 간의 파라미터 W와는 별개이다)

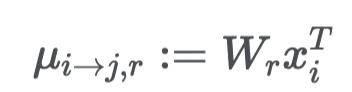
이해하기 쉽게 풀어쓰자면 1점을 부여한 user-item의 모든 관계를 W1로 학습하여 generalization 하고, 2점을 부여한 user-item의 모든 관계를 W2로 학습하여 generalization 한다는 것이다. 그리고 이를 이용해 X와 WR 두 행렬로 연산된 결과를 활용하겠다는 것이다. 마치 Matrix Factorization에서 user-item rating 값을 latent factor로 generalization 하는 것과 유사하다는 것을 알 수 있다. 다만 GC에서는 MF처럼 내적의 연산결과를 추정하는 것이 아니라, 관계 자체를 generalization 한다는 것에서 조금 더 나은 아이디어라고 볼 수 있다. 그리고 User와 Item의 Densed Feature를 활용할 수 있다는 점에서 꽤나 큰 성능 차이를 보일 것이다.

**Weight sharing(message)**

message는 edge를 통해 두 노드가 weight를 주고받는다는 측면에서 생겨난 개념이다. user i와 item j가 rating edge r로 연결될 때 message는 아래와 같이 정의된다.

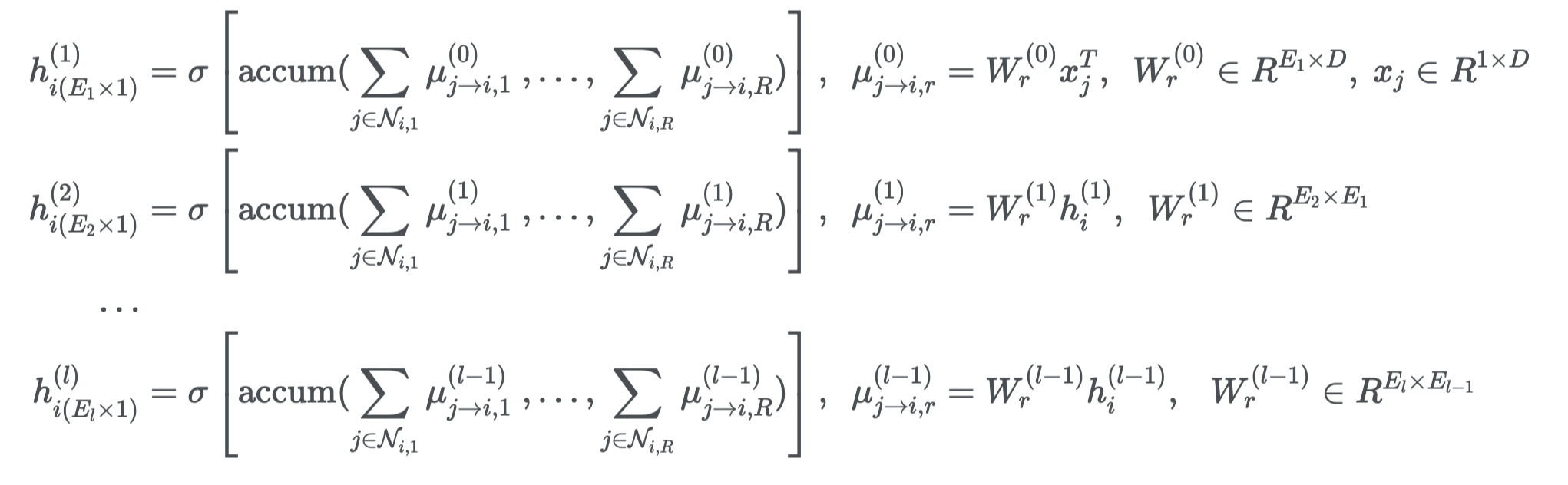


결국 user i에 대한 latent vector는 인접노드인 item j의 영향을 받으며, Wr( rating r에 대한 weight)공간에 item j를 표현(선형변환)하여 정의된다는 것이다. 두 노드는 서로 연결 되어있으니 반대도 성립한다.

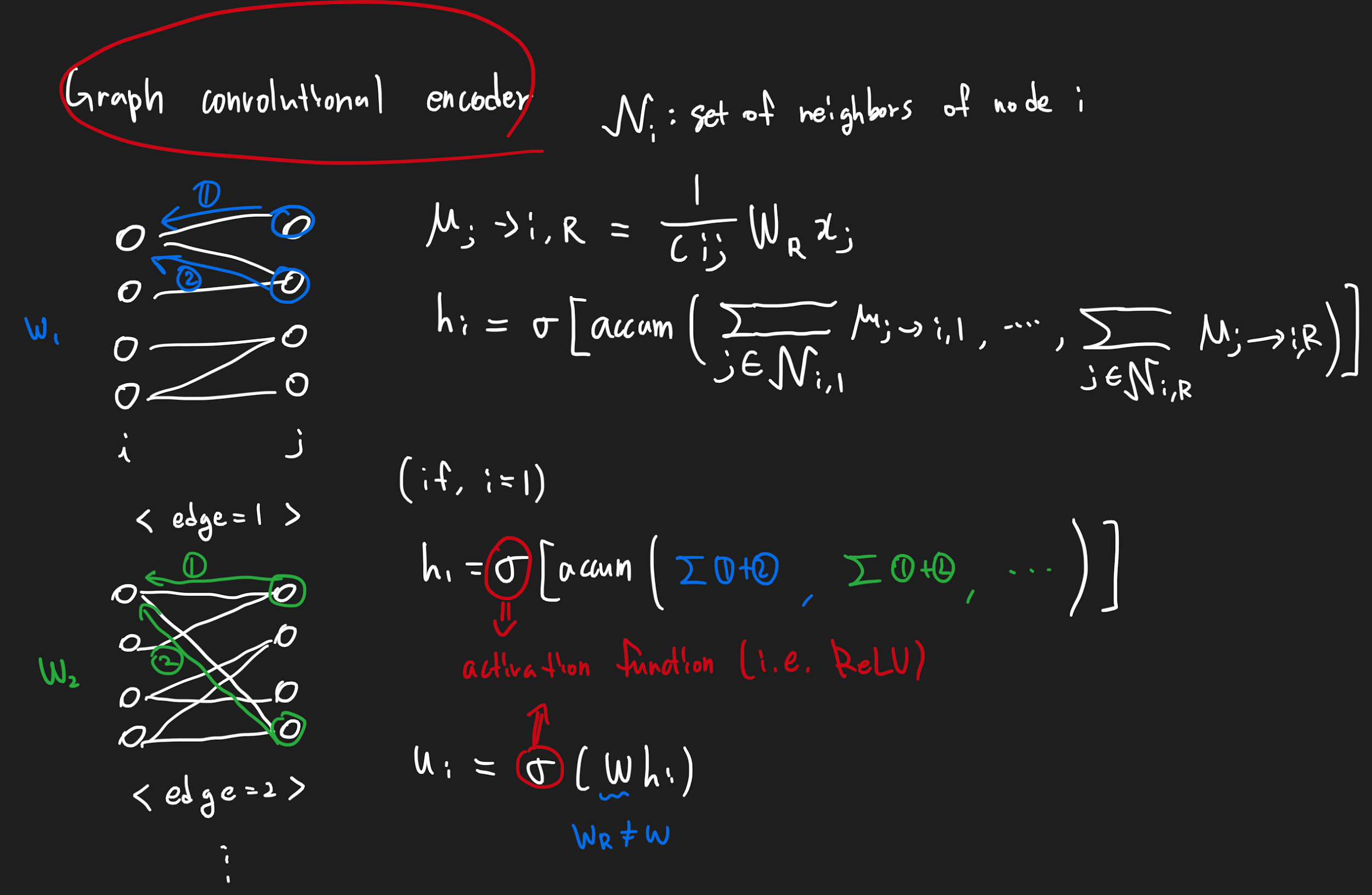
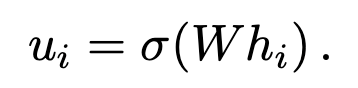


식을 보면 알 수 있듯, 인접한 노드끼리는 같은 weight를 공유하고 있는 것이 핵심이다. 결국 학습을 거듭할수록 Wr 공간은 rating r을 갖는 user, item의 정보들로 표현이 될 것이다.

Message passing 단계를 거치면 아래의 그림과 같이 hi가 생겨나게 되는데, 이 결과가 바로 NN의 히든 레이어 역할을 하게 된다.

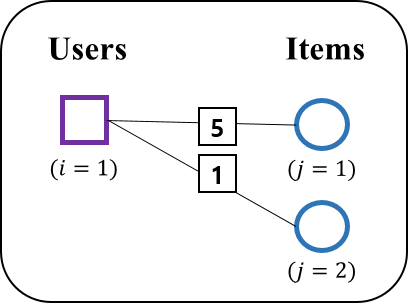


히든 레이어를 아래와 같이 한 번 더 통과시켜주면, output이 바로 임베딩 벡터가 된다.

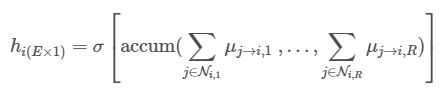


#### **c) example**

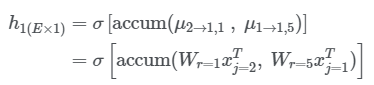
지금까지 개념적으로만 살펴봤다면 이제 구체적인 예를 들어보자. 해당 예시는 다음과 같이 user1이 item1과 item2에 각각 5점, 1점을 부여한 상황을 가정한다.



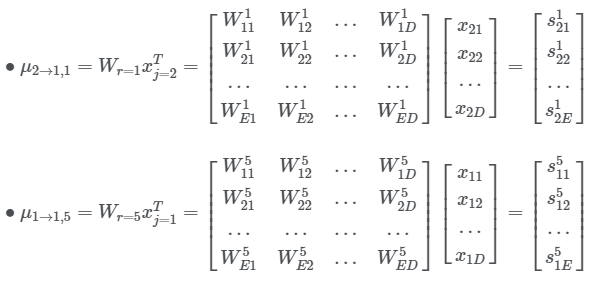
앞서 hi는 다음과 같이 정의된다 하였다. 여기서 Ni,R은 user i와 인접한 노드 중, rating = r을 만족하는 item 노드 집합을 나타낸다.



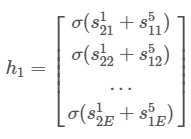
우리는 user1에 대해 보고있으니 i = 1이며, rating edge가 2개이므로 N1,1={2}, N1,5={1}가 존재한다. 이를 바탕으로 할 때, h1은 다음과 같다.



이어 message μ가 다음과 같이 계산된다고 하자.



이때 accum(.) = sum(.)으로 가정하면 h1은 다음과 같이 계산된다. 여기서 σ(.)는 activation function으로 논문에서는 relu를 사용했다.



결국 user 1에 대한 latent vector h1은 user1과 인접한 노드 item1, item2의 정보를 사용하여 계산된다는 것에 주목하자. 여기에 다른 item의 정보는 사용되지 않았다. 이는 cnn의 convolution이 영역을 한칸씩 옮기며 주변 탐색해 가는 원리랑 비슷하다.

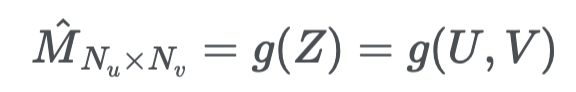
또한 앞서 여기서는 adjacency matrix A를 사용하지 않는다고 했는데, 그 이유는 단순하다. A를 곱하는 이유는 (matrix 연산을 직접 해보면 알수 있듯) 인접 노드만 가져오기 위함인데, 여기서는 이미 인접 노드만 가져와서 계산을 하기때문에 굳이 고려할 필요가 없는 것이다.

<https://untitledtblog.tistory.com/152> <GCN 기본개념>

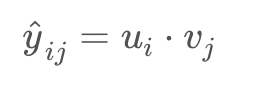
### **Decoder**

#### **a) 개념**

decoder 모델 g(.)는 다음과 같이 정의된다. 이는 앞서 latent vector를 갖고 rating을 예측하는 것이라 하였다.



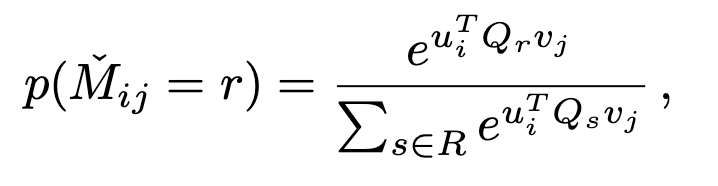
가장 Classic하게 평점을 예측하는 방법은 User 임베딩 벡터와 Item 임베딩 벡터를 바로 내적 계산하는 것이다.



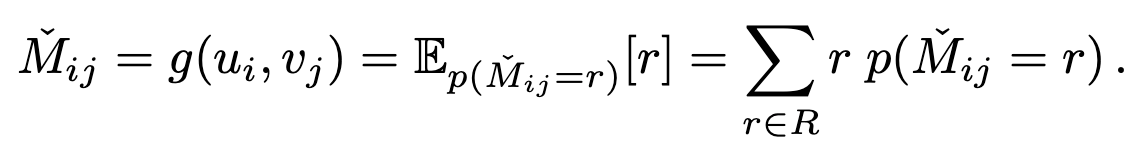
<https://yeomko.tistory.com/5> <MF 내적>

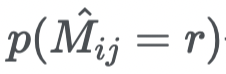
#### **b) Bilinear decoder**

GCMC에서는 아래의 softmax[[1]](#footnote-0)의 개념을 사용하여 각 점수(r=1,2,...R)를 확률표현으로 나타낸다.



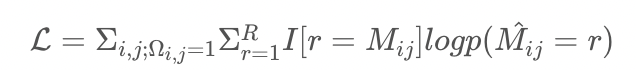
연산 결과 rating에 따른 확률의 분포가 생성되고, 이 수식에는 trainable parameter인 Q가 사용된다(ExE dimension). 그리고 최종적인 rating prediction은 아래의 수식을 한 번 더 거쳐 완성된다.



즉, item j에 대한 user i의 rating은 rating을 가중평균해서 예측하며, 이때 가중치로 를 사용한다. 여기서 Qr은 encoder모델의 Wr과 같은 parameter matrix다.

### **Model Training**

#### **a) Loss function**

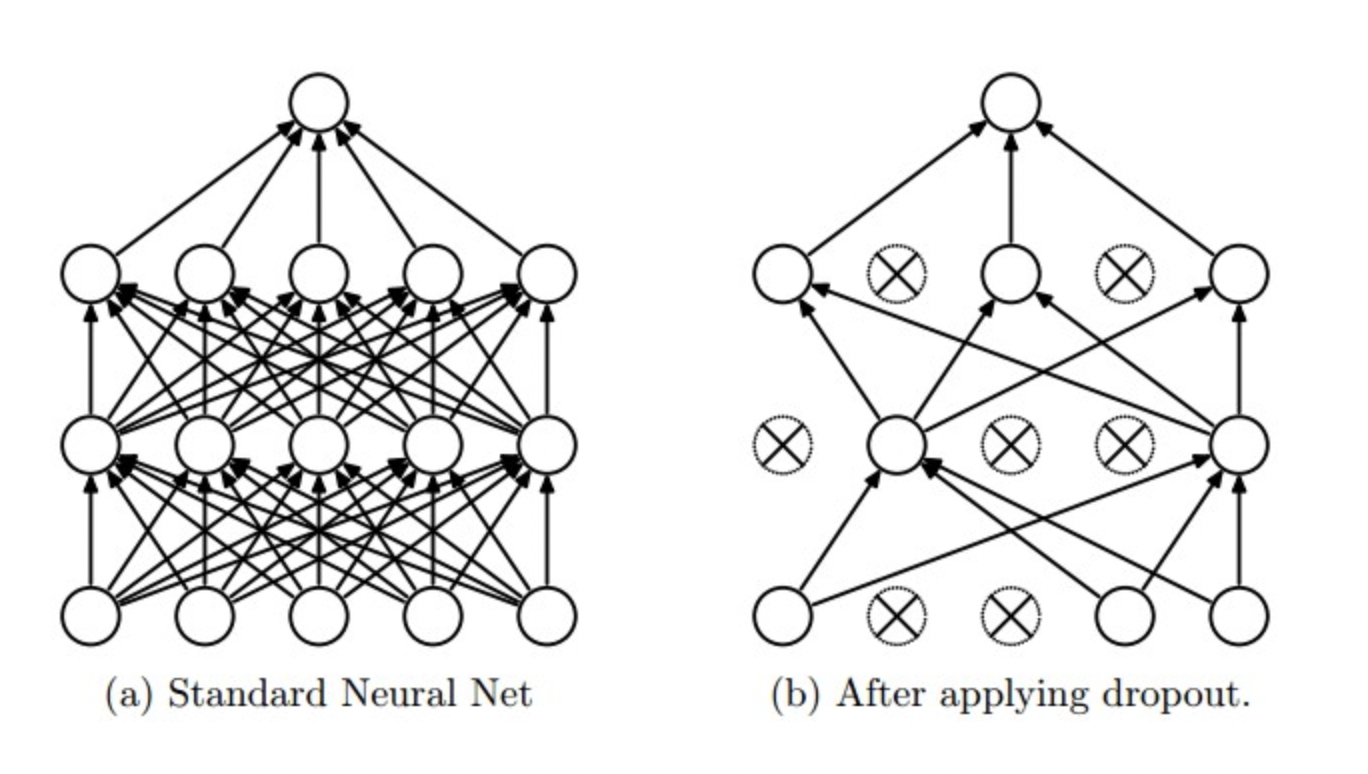


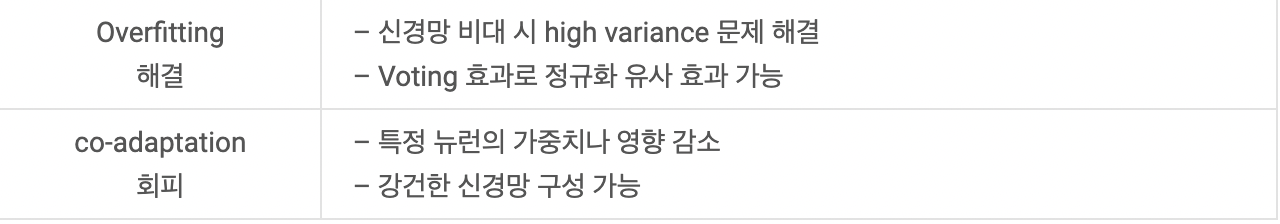
\* I 함수: Indicator Function. [ ] 안이 참일 때 1, 그렇지 않을 때 0의 값 (일종의 **Mask** 역할로 평점 기록이 존재하는 Link에 대해서만 negative log likelihood를 적용) 결과적으로 negative likelihood L을 최소화 하는 parameter Θ = {W,Wr,Qr}를 찾는 것이 목표

#### **b) Node dropout**

과적합(overfitting)의 위험을 줄이고 학습속도를 개선하는 등의 문제를 해결하기 위한 장치로 특정 Node에 대해 pdropout의 확률로 밖으로 나가는 모든 Message를 무작위로 drop out하는 방식으로 학습하는 것을 의미한다.

Message는 Dropout 단계 이후에 Rescaled된다. 본 논문에서는 Node Dropout을 적용한 결과 임베딩 결과물이 특정 User나 Item에 좀 더 독립적인 것을 확인할 수 있었으며 추가적으로 Hidden Layer Unit에 Regular Dropout을 적용하였다.





#### **c) Mini batch**

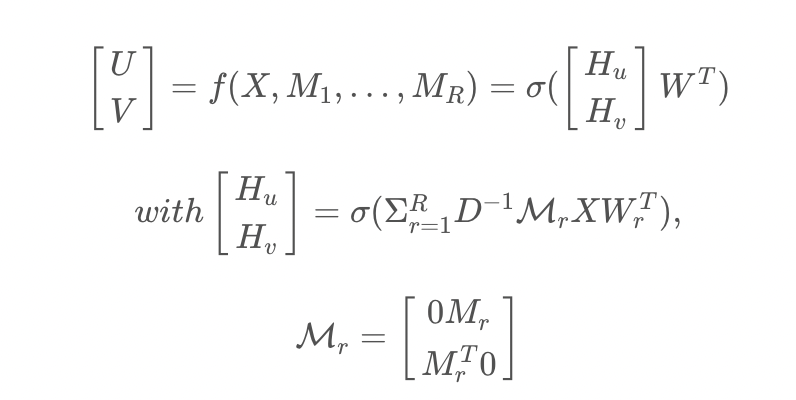
#### 미니 배치는 User-Item Pair 총합에서 오직 고정된 수의 Contribution 만을 추출하여 학습에 사용한다는 것을 의미한다. 이렇게 함으로써 현재 Batch에서 존재하지 않는, 각 평점 Class의 User/Item 행을 제거할 수 있다.

효율적인 정규화, 모델 학습에 필요한 메모리 경감의 효과가 있다.

### **Vector Implementation**

실제로 코드를 구현하여 모델을 학습할 때, 행렬을 활용한 효율적인 연산 방법.

User Feature와 Item Feature의 수가 D로 같다고 가정한 경우 Embedding 행렬은 아래와 같이 구할 수 있다.



이 때 D는 대각행렬이며, 대각원소 Dii = |Ni| 을 의미

### **Input Feature Representation and Side Information**

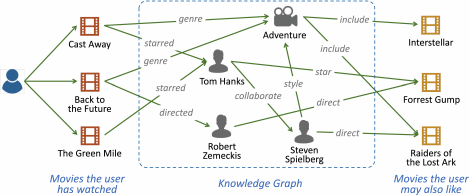
### Feature가 다른 User(Item)나 그들의 관심 사항을 제대로 구분할만한 충분한 정보를 갖고 있지 못한 경우 정보 병목현상이 발생할 수 있다. -> Side Information으로 활용.

### Node i에 대해 User/Item Feature Vector 를 만들고, 이를 Dense Hidden Layer에 독립적인 채널을 통해 직접적으로 투입한다.

### 

이 때 , 는 학습 가능한 Weight 행렬이며, User Weight Matrix와 Item Weight Matrix는 구분된다. Graph Convolutional Layer를 위한 Node Feature를 담고 있는 Input Feature Matrix X는 Graph에 있는 모든 Node에 대한 고유한 One-Hot 벡터와 함께 Identity Matrix로 선택된다. 본 논문에서 사용된 데이터셋에 한해서는, User/Item Content 정보는 제한적인 크기를 갖고 있기 때문에 이들을 Side Information으로 이용하고자 한다.

Side Information은 꼭 Per-Node Feature 벡터의 형태일 필요는 없다. 사실 그래프 구조여도 되고, NLP 혹은 이미지 데이터일수도 있다. 물론 이 경우 위의 식은 RNN, CNN, GNN와 같이 미분 가능한 다른 모듈로 적절히 대체되어야 한다.



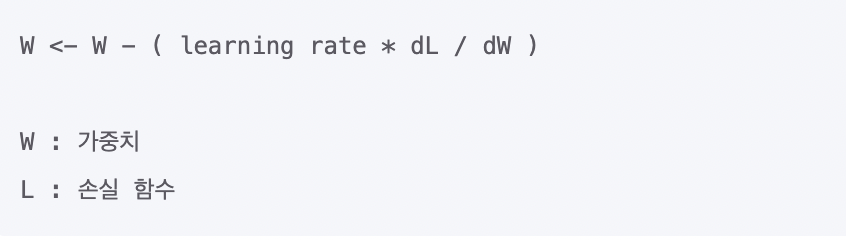
[https://zzaebok.github.io/knowledge\_g](https://zzaebok.github.io/knowledge_graph/recommender_system/KG_recommend/)

[raph/recommender\_system/KG\_recommend/](https://zzaebok.github.io/knowledge_graph/recommender_system/KG_recommend/) <지식그래프를 이용한 추천시스템>

# **Optimization**

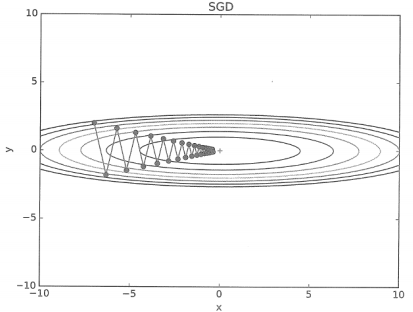
신경망 학습의 목적은 손실 함수의 값을 가능한 한 낮추는 매개변수를 찾는 것이며, 이러한 문제를 푸는 것을 최적화(optimization)이라고 한다.

#### **확률적 경사 하강법 - SGD(Stochastic Gradient Descent)**

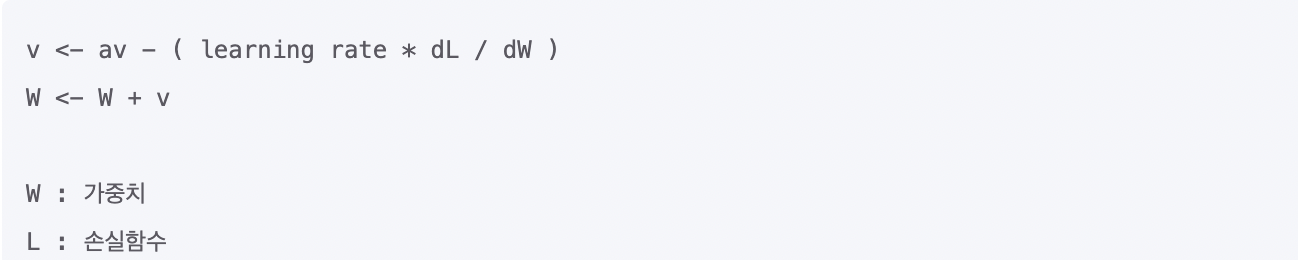


최적의 매개변수 값을 찾는 단서로 매개변수의 기울기(미분)을 이용.

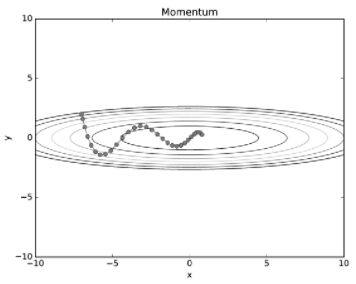
매개변수의 기울기를 구해, 기울어진 방향으로 매개변수 값을 갱신하는 일을 계속 반복한다.



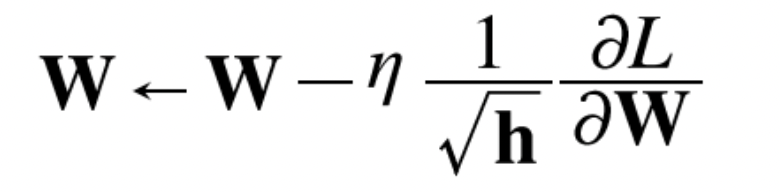
#### 모멘텀(Momentum)



SGD와 차이점을 보면 av 값을 더해준게 눈에 띈다. 여기서 a는 고정된 상수값이고(ex 0.9) v는 물체의 속도라고 생각하면 된다. 해당 방향으로 진행할 수록 공이 기울기를 따라 구르듯 힘을 받는다.



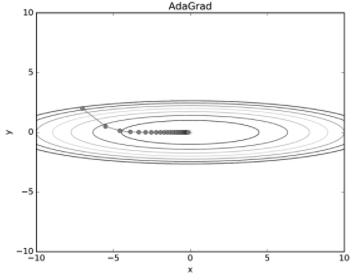
#### AdaGrad



학습에서는 Learning Rate가 중요하다. 이 값이 너무 작으면 학습 시간이 너무 길어지고, 반대로 너무 크면 발산하여 학습이 제대로 이뤄지지 않는다.

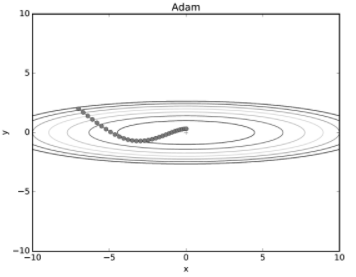
이러한 학습률을 정하는 효과적 기술로 학습률 감소(learning rate decay)가 있다. 이는 학습을 진행하면서 학습률을 점차 줄여나가는 방법이다.

학습률을 서서히 낮추는 가장 간단한 방법은 전체 학습률 값을 일괄적으로 낮추는 것이지만, 이를 더 발전시킨 것이 AdaGrad이다. AdaGrad는 ‘각각의’ 매개변수에 ‘맞춤형’값을 만들어준다.



#### Adam

모멘텀 + AdaGrad 정확히 파고들면 다르지만, 직관적으로 해석하면 모멘텀과 AdaGrad를 융합한 듯한 방법이다.



### 본 논문에서는 Adam 최적화 함수를 사용한다.

### **Conclusions**

논문이 제시한 아이디어는 graph구조를 사용해서 rating matirx를 예측하는 시스템으로, 크게 encoder와 decoder로 이뤄져 있으며. encoder를 통해 user, item에 대한 latent vector를 구하고, decoder를 통해 rating을 예측하는 시스템이다.

encoder는 Graph Convolutional Layer를 포함하는데, 이 layer는 User-Item 상호작용 Graph에서 오가는 Message 속에서 User/Item Embedding을 구성하게 된다. Bilinear Decoder와 결합하여 새 평점은 labeled edge의 형태로 예측된다.

본 논문에서 제안한 모델은 여러 Benchmark 모델을 능가하는 결과를 보여주었으며 확률적 미니배치를 통해 더 큰 데이터셋에서 학습될 수 있다는 것을 보여준다.

미래에 이 모델을 더욱 크고 Multi-modal 형태의 데이터 (Text, Image, Graph-based Information으로 구성된) 에도 적용할 수 있기를 바라며 이러한 경우 GCMC 모델과 Recurrent 또는 Convolutional Neural Network를 결합하여 사용할 수 있을 것이다. 마지막으로 Attention Mechanism은 이러한 종류의 모델의 성능을 더욱 향상시키는 방법이 될 수 있을 것이다.

## 

### **참고 자료**

논문 <https://arxiv.org/pdf/1706.02263.pdf>

논문 코드

<https://github.com/riannevdberg/gc-mc>

논문 리뷰

<https://greeksharifa.github.io/machine_learning/2020/12/06/GCMC/>

<https://yamalab.tistory.com/133>

<https://leehyejin91.github.io/post-gcmc/>

1. Softmax(소프트맥스)는 입력받은 값을 출력으로 0~1사이의 값으로 모두 정규화하며 출력 값들의 총합은 항상 1이 되는 특성을 가진 함수. 결과 값의 정도에 따라 추가 판단하기도 한다.

   <https://m.blog.naver.com/wideeyed/221021710286> [↑](#footnote-ref-0)