<Graph based recommender system study with DeepFindr - GNN>

장석규

<그래프 데이터의 특성>

1. 데이터셋의 크기가 바뀔 수 있다.

- 고정된 인풋 크기를 원하는 NN에서 사용하기 어렵다.
- 다른 크기의 데이터를 같은 사이즈로 만드는 것은 그래프에서 정의되지 않은 연산이다.
- 임의의 인풋 크기를 갖는 그래프를 처리할 방법이 필요하다.

2. 동형 (Isomorphism)

- 다르게 보이는 데이터도 그래프 상에서는 동일하게 보일 수 있다.(예: 상하반전)
- 인접행렬을 feedforward network에 바로 적용하기 어렵다.(입력 순서에 민감하기 때문)

3. 비유클리드

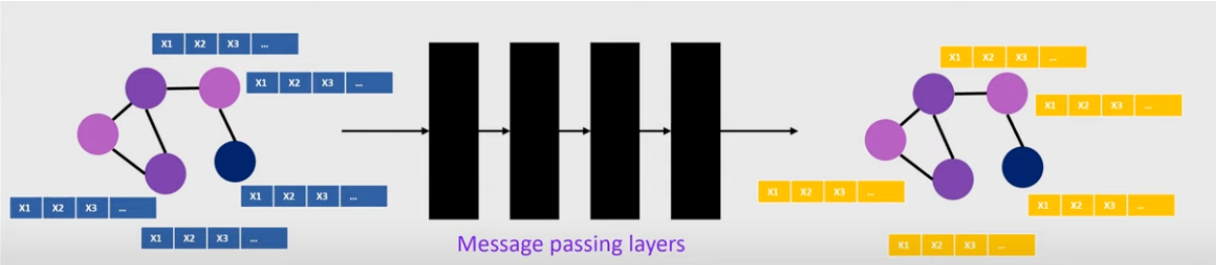
- 특정한 축이 없고 새로운 distance metric등이 필요하다.
- 이에 그래프를 활용한 딥러닝을 geometric deep learning이라 부른다.

<Representation learning>

GNN의 기본적인 개념은 representation learning이다.

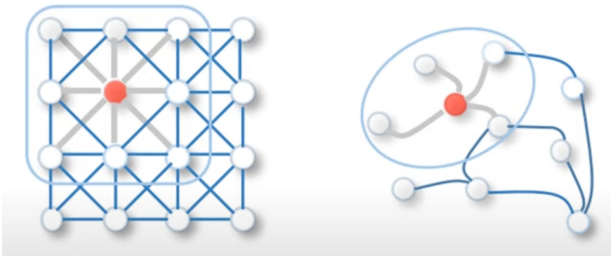
- 그래프 데이터에 적합한 표현과 그에 대한 NN을 학습한다.
- GNN의 아웃풋은 임베딩이라 불린다. (구조정보, 다른 노드의 피처 정보를 포함한다.)
- 임베딩을 통해 각 노드는 다른 노드와의 연결, 문맥, 피처 정보를 알 수 있다.
- 노드 분류를 위해서는 각 노드의 임베딩을 사용하면 된다.
- 전체 그래프 데이터를 사용하기 위해서는 모든 노드의 임베딩을 특정한 방법으로 하나로 합치고 하나의 새로운 임베딩으로 만든다.
- 풀링 연산이 그래프를 고정된 크기로 압축하기 위해 사용되기도 한다.
- 유사한 노드, 유사한 그래프는 유사한 임베딩이 된다.

<GNN>



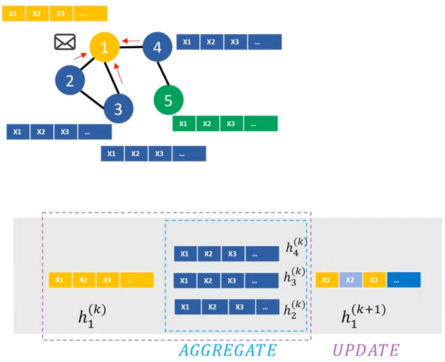
노드 피처 정보를 임베딩으로 만들기 위해서는 Message passing layers를 거쳐야 한다.

<Graph Convolution>



- Message passing layers에서 현재 정보 및 이웃 정보를 압축하기 위한 방법
- 이미지(좌)는 단순히 그리드 구조에서 옆으로 슬라이드한다.
- 그래프(우)도 이웃을 포함해 Convolution을 수행하여 새로운 임베딩을 생성한다.(모든 노드에 대해 동시에 수행)

<Details>



- 노드 1을 기준으로 convolution을 수행한다고 가정하면 먼저 직접 연결된 노드들의 정보를 수집한다.
- 이후 이웃 노드의 정보를 하나로 합치고(aggregation) 노드1의 벡터를 업데이트한다.
- 여러 번 수행하면(GNN depth) 직접 연결되지 않은 먼 이웃의 정보도 포함할 수 있다.
- 너무 많이 수행하면 Over-smoothing 문제 발생: 모든 노드가 유사한 임베딩을 가지게 된다.

$$h_u^{(k+1)} = UPDATE^{(k)}\left(h_u^{(k)}, AGGREGATE^{(k)}(\{h_v^{(k)}, \forall v \in \mathcal{N}(u)\})\right)$$

<GNN Variants>

- GCN(Graph Convolution Network): Aggregation과 update를 하나의 연산으로 합침

$$\mathbf{h}_v^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}}\right)$$

Self-loop

Sum of normalized neighbor embeddings

- Multi-layer-perceptron as Aggregator
- Graph Attention mechanism: 그래프의 노드에 중요성(가중치)을 부여
- Gated Graph Neural Network