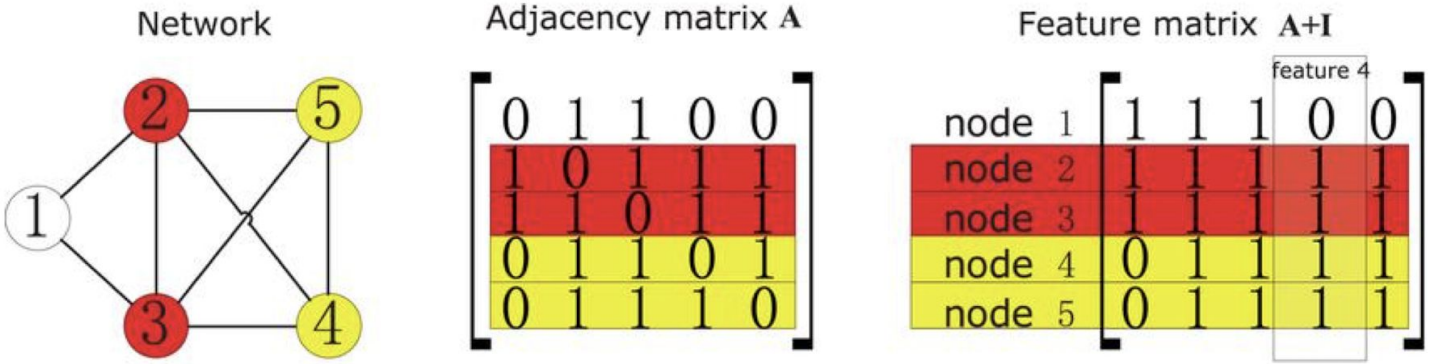
11/2

1. GNN - Graph

그래프란?
Edge로 연결된 node들의 집합, adj와 feature matrix로 표현됨

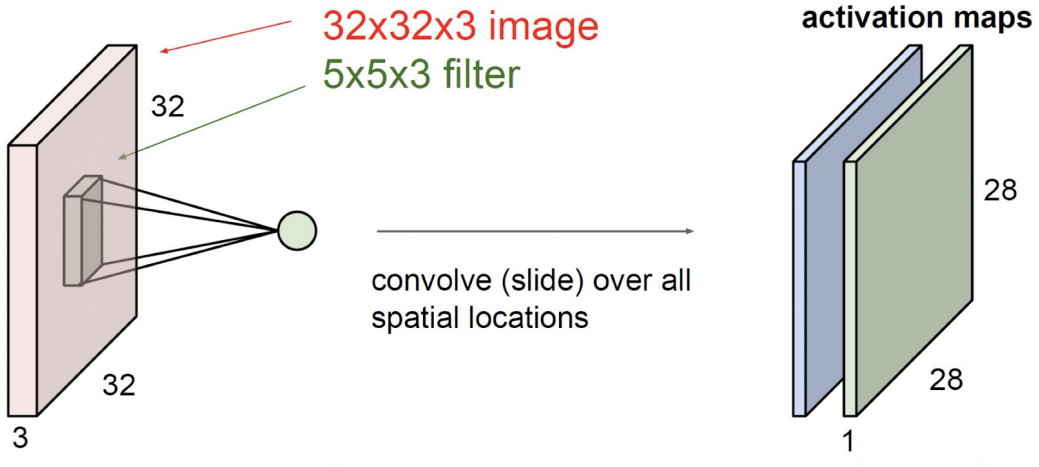
Vertex (Node)
Node Feature Matrix

Edge
Adjacency Matrix



1. GNN - CNN에서의 Convolution

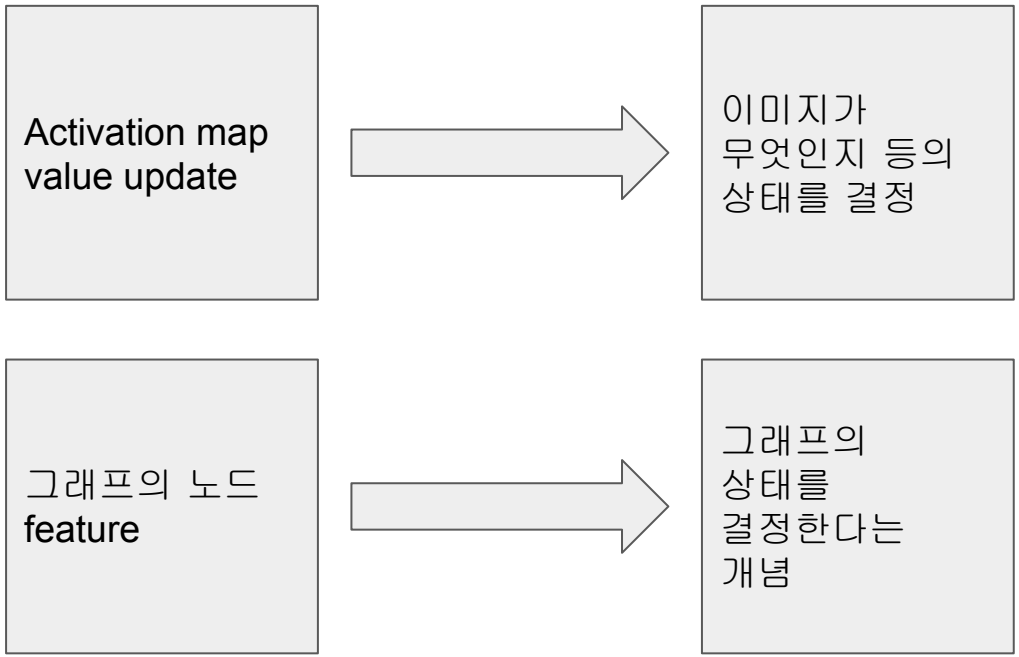
-이미지에 필터를 씌우고, local의 값을 뽑아냄
-layer가 깊어질수록 더 많은, 고차원적인 정보를 뽑아낸다.
-한 layer에서 같은 가중치를 공유하며, 필터를 슬라이드하면서 뽑아낸다.



1. GNN - CNN

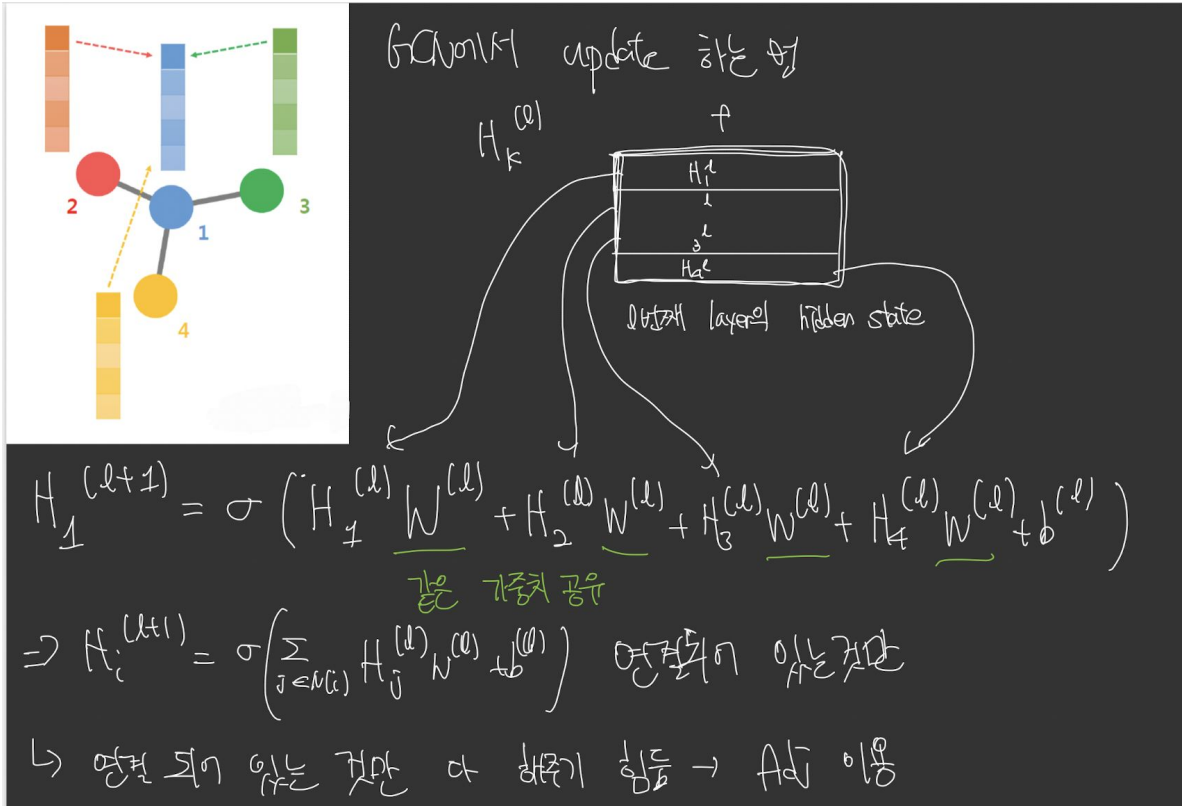
CNN의 원리가
각 layer의 activation map의 value를 update를 해서
그 value가 이미지의 상태를 결정한다.

GCN에 적용을 해보면, 노드의 feature가 그래프의 상태를 결정한다.
> 노드의 feature를 update를 하면 같은 효과를 나타낼 수 있다.

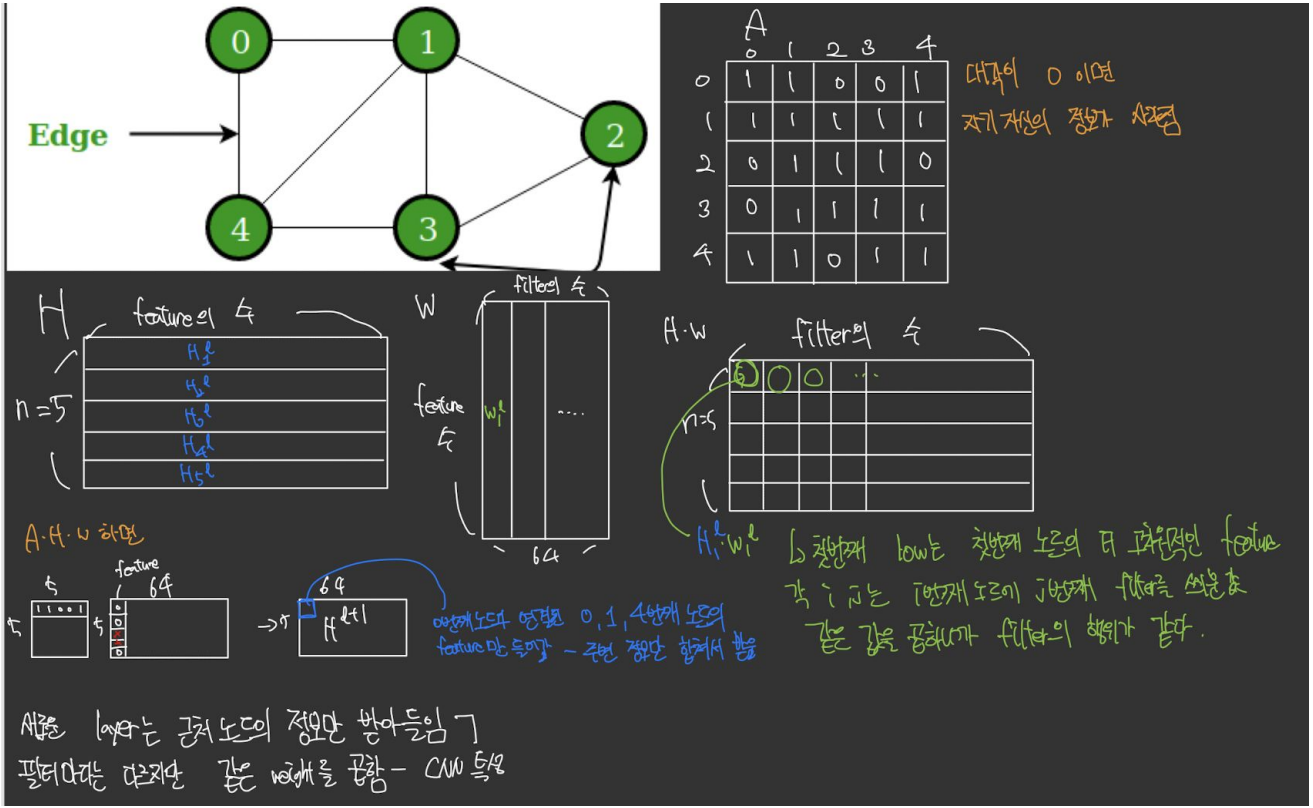


>> 그래프의 노드 feature을 update 하면 같은 효과를 나타낼 수 있다.

1. GNN - update


2
1
3
4
update 하는 법
이전에 layer의 hidden state
같은 가중치 공유
영향이 있는것만
영향이 있는 것만 다 해주기 힘들 → Adj 이용

1. GNN - update Adj matrix 이용



1. GNN - update Adj matrix 이용

결국 AHW를 하면 연결된 노드의 feature만 들어가고 주변 정보만 합쳐서 뽑는다.

update하는 과정이 convolution 연산과 같이 주변의 정보들을 이용하고, 같은 가중치를 한다는 특징이라서 GCN이라고 부른다

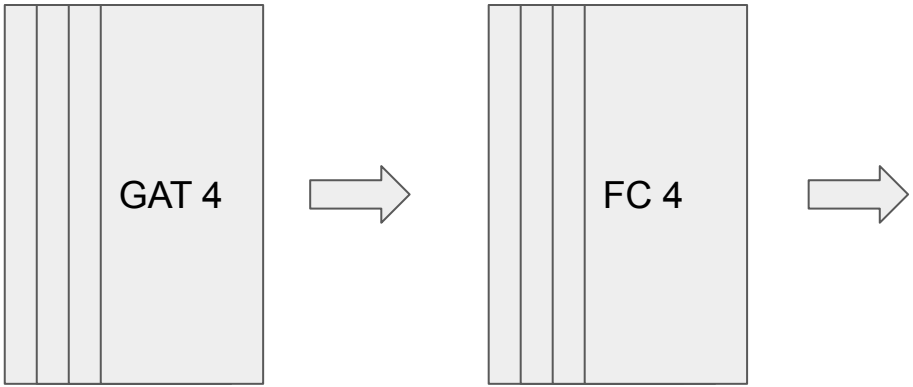
그리고 layer이 깊어질수록 모든 그래프를 cover할 수 있다.

2. GNN libray

PyG *(PyTorch Geometric)* is a library built upon PyTorch to easily write and train Graph Neural Networks (GNNs) for a wide range of applications related to structured data.

2. GNN libray

```
GNN_Model(
  (gconv1): ModuleList(
    (0): GAT_gate(
      (W): Linear(in_features=140, out_features=140, bias=True)
      (gate): Linear(in_features=280, out_features=1, bias=True)
      (leakyrelu): LeakyReLU(negative_slope=0.2)
    )
    (1): GAT_gate(
      (W): Linear(in_features=140, out_features=140, bias=True)
      (gate): Linear(in_features=280, out_features=1, bias=True)
      (leakyrelu): LeakyReLU(negative_slope=0.2)
    )
    (2): GAT_gate(
      (W): Linear(in_features=140, out_features=140, bias=True)
      (gate): Linear(in_features=280, out_features=1, bias=True)
      (leakyrelu): LeakyReLU(negative_slope=0.2)
    )
    (3): GAT_gate(
      (W): Linear(in_features=140, out_features=140, bias=True)
      (gate): Linear(in_features=280, out_features=1, bias=True)
      (leakyrelu): LeakyReLU(negative_slope=0.2)
    )
  )
  (FC): ModuleList(
    (0): Linear(in_features=140, out_features=128, bias=True)
    (1): Linear(in_features=128, out_features=128, bias=True)
    (2): Linear(in_features=128, out_features=128, bias=True)
    (3): Linear(in_features=128, out_features=1, bias=True)
  )
  (embede): Linear(in_features=56, out_features=140, bias=False)
)
```



to classify whether the protein complex model is correct or incorrect

3. GNN_DOVE 메일

Dear Jin,

Thank you so much for your interest in our work GNN-DOVE! I am happy to hear that you have tried it and successfully run it locally.

For training, the same questions have been widely asked by the users. Therefore, I have attached detailed instructions at https://github.com/kiharalab/GNN_DOVE/issues/2 and https://github.com/kiharalab/GNN_DOVE/issues/4. I hope that can help you fully understand the training process of GNN-DOVE.

If you have any further questions, feel free to email me.

Best,
Xiao Wang

https://github.com/kiharalab/GNN_DOVE/issues/2