

그래프를 이용한 기계 학습

#9 그래프 신경망이란 무엇일까? (기본)

신기정

(KAIST AI대학원)

1. 정점 표현 학습 복습

2. 그래프 신경망 기본

3. 그래프 신경망 변형

4. 합성곱 신경망과의 비교

4. 실습: DGL 라이브러리와 GraphSAGE를 이용한 정점 분류

1. 정점 표현 학습 복습

1.1 정점 표현 학습

1.2 변환식 정점 표현 학습과 귀납식 정점 표현 학습

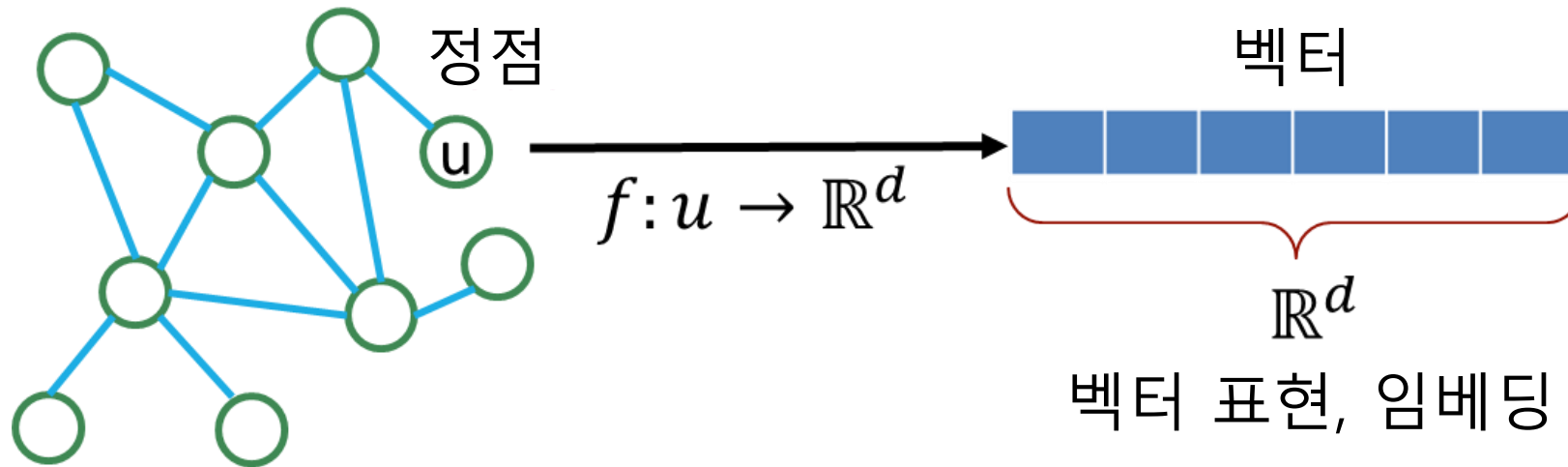
1.1 정점 표현 학습

정점 표현 학습이란 **그래프의 정점들을 벡터의 형태로 표현하는 것**입니다

정점 표현 학습은 간단히 **정점 임베딩(Node Embedding)**이라고도 부릅니다

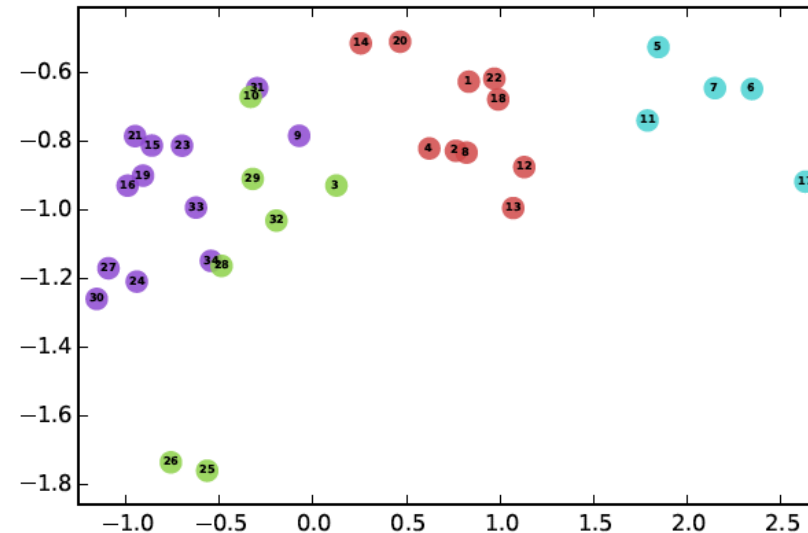
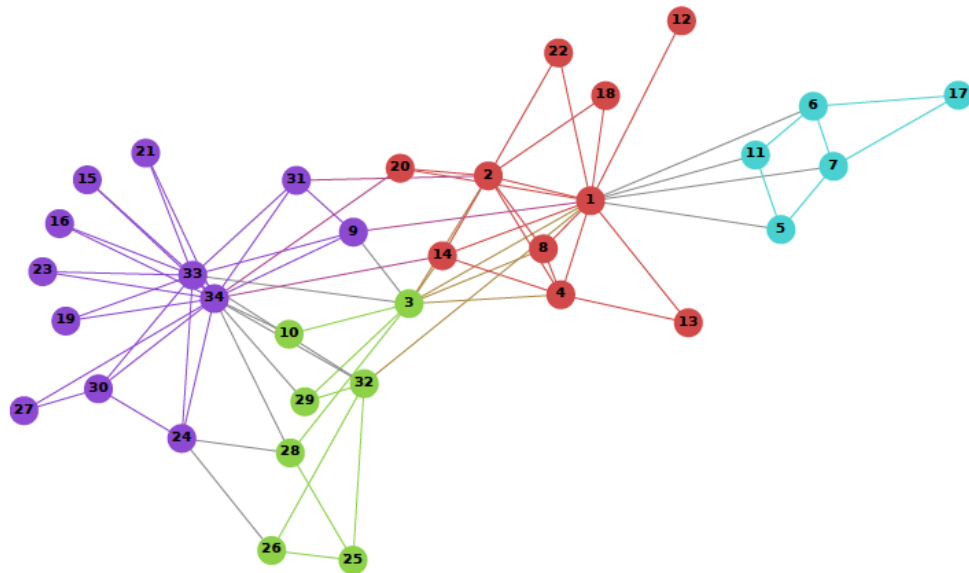
정점 표현 학습의 입력은 그래프입니다

주어진 그래프의 **각 정점 u 에 대한 임베딩, 즉 벡터 표현 z_u** 가 정점 임베딩의 **출력**입니다



1.1 정점 표현 학습

그래프에서의 정점간 유사도를 임베딩 공간에서도 “보존”하는 것을 목표로 합니다

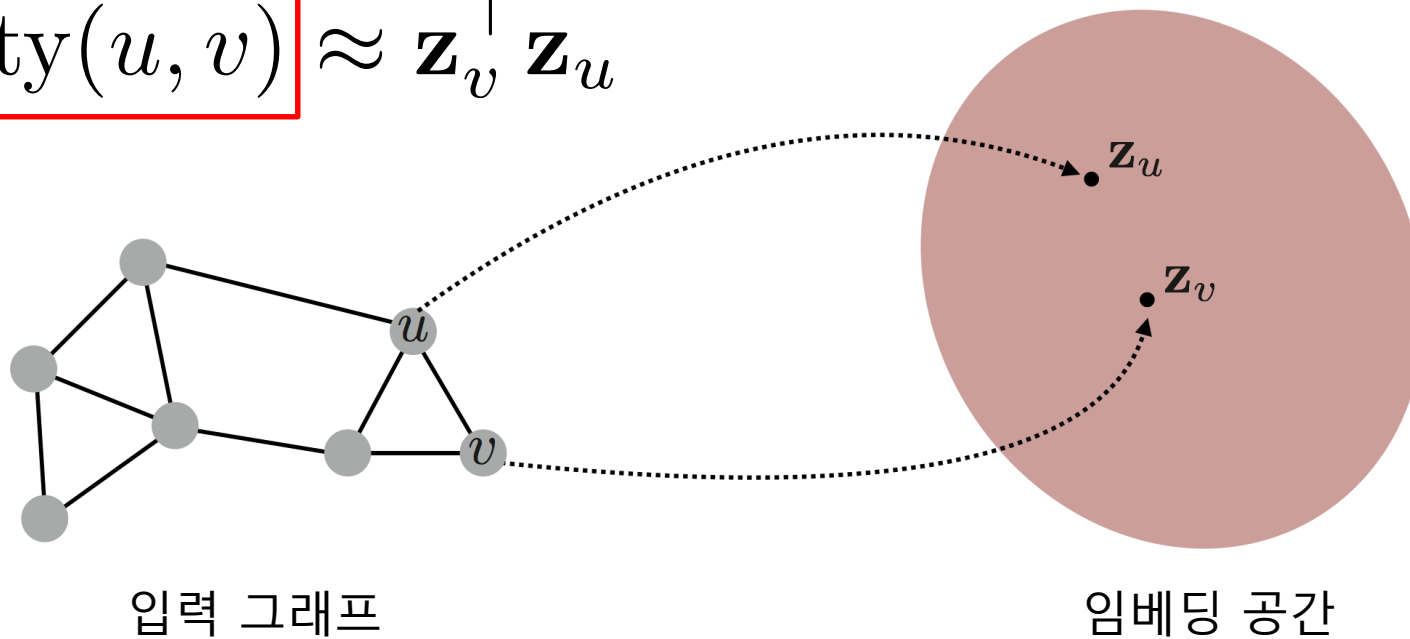


1.1 정점 표현 학습

그래프에서 두 정점의 유사도는 어떻게 정의할까요?

그래프에서 정점의 유사도를 정의하는 방법에 따라,
인접성/거리/경로/중첩/임의보행 기반 접근법으로 나뉩니다

$$\text{similarity}(u, v) \approx \mathbf{z}_v^\top \mathbf{z}_u$$

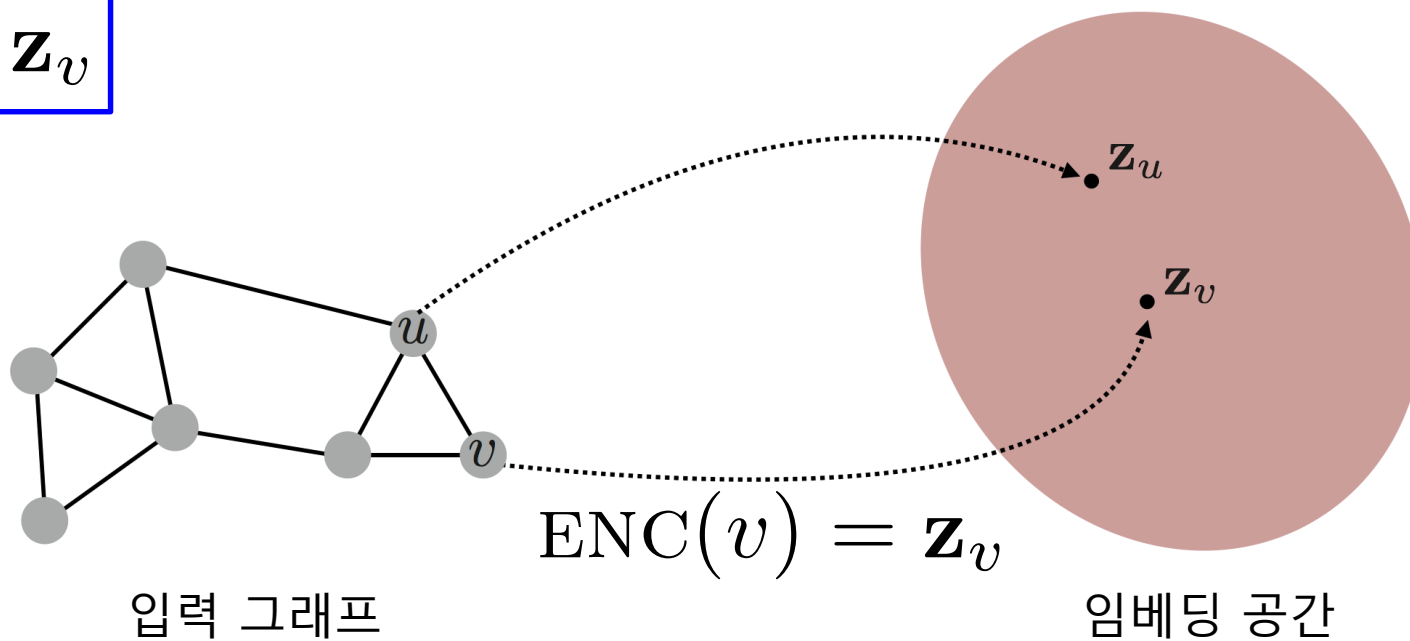


1.2 변환식 정점 표현 학습과 귀납식 정점 표현 학습

지금까지 소개한 정점 임베딩 방법들을 **변환식(Transductive) 방법**입니다

변환식(Transductive) 방법은 학습의 결과로 정점의 임베딩 자체를 얻는다는 특성이 있습니다
정점을 임베딩으로 변화시키는 함수, 즉 인코더를 얻는 귀납식(Inductive) 방법과 대조됩니다

$$\boxed{\text{ENC}}(v) = \boxed{\mathbf{z}_v}$$



1.2 변환식 정점 표현 학습과 귀납식 정점 표현 학습

출력으로 임베딩 자체를 얻는 변환식 임베딩 방법은 여러 한계를 갖습니다

- 1) 학습이 진행된 이후에 추가된 정점에 대해서는 임베딩을 얻을 수 없습니다
- 2) 모든 정점에 대한 임베딩을 미리 계산하여 저장해두어야 합니다
- 3) 정점이 속성(Attribute) 정보를 가진 경우에 이를 활용할 수 없습니다

1.2 변환식 정점 표현 학습과 귀납식 정점 표현 학습

출력으로 인코더를 얻는 귀납식 임베딩 방법은 여러 장점을 갖습니다

- 1) 학습이 진행된 이후에 추가된 정점에 대해서도 임베딩을 얻을 수 있습니다
- 2) 모든 정점에 대한 임베딩을 미리 계산하여 저장해둘 필요가 없습니다
- 3) 정점이 속성(Attribute) 정보를 가진 경우에 이를 활용할 수 있습니다

$$\text{ENC}(v) = \begin{array}{l} \text{그래프 구조와 정점의 부가 정보를} \\ \text{활용하는 복잡한 함수} \end{array}$$

오늘 소개하는 그래프 신경망(Graph Neural Network)은 대표적인 귀납식 임베딩 방법입니다

2. 그래프 신경망 기본

2.1 그래프 신경망의 구조

2.2 그래프 신경망의 학습

2.3 그래프 신경망의 활용

2.1 그래프 신경망 구조

그래프 신경망은 **그래프**와 **정점의 속성 정보**를 입력으로 받습니다

그래프의 **인접 행렬**을 **A**라고 합니다

인접 행렬 **A**은 $|V| \times |V|$ 의 이진 행렬입니다

각 정점 u 의 속성(Attribute) 벡터를 x_u 라고 합니다

정점 속성 벡터 x_u 는 m 차원 벡터이고, m 은 속성의 수를 의미합니다

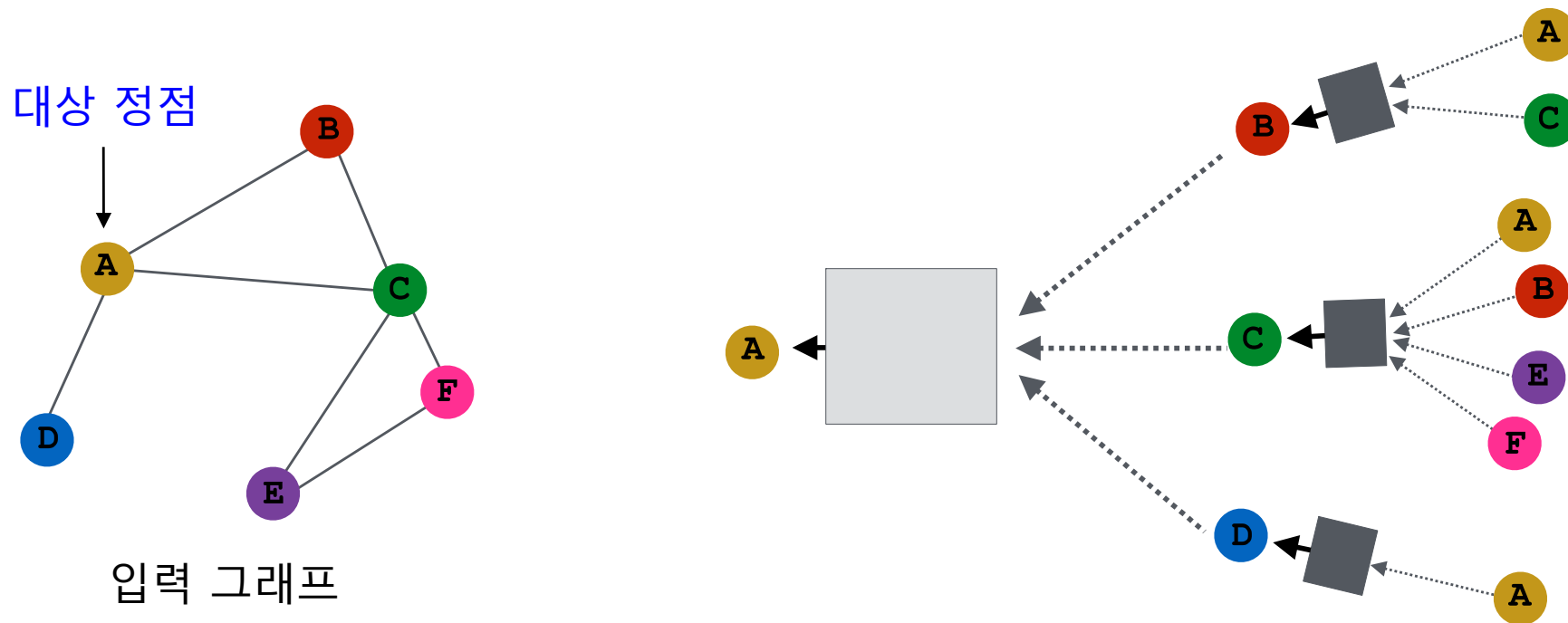
정점의 속성의 예시는 다음과 같습니다

- 온라인 소셜 네트워크에서 사용자의 지역, 성별, 연령, 프로필 사진 등
- 논문 인용 그래프에서 논문에 사용된 키워드에 대한 원-핫 벡터
- PageRank 등의 정점 중심성, 군집 계수(Clustering Coefficient) 등

2.1 그래프 신경망 구조

그래프 신경망은 **이웃 정점들의 정보를 집계하는 과정을 반복**하여 **임베딩**을 얻습니다

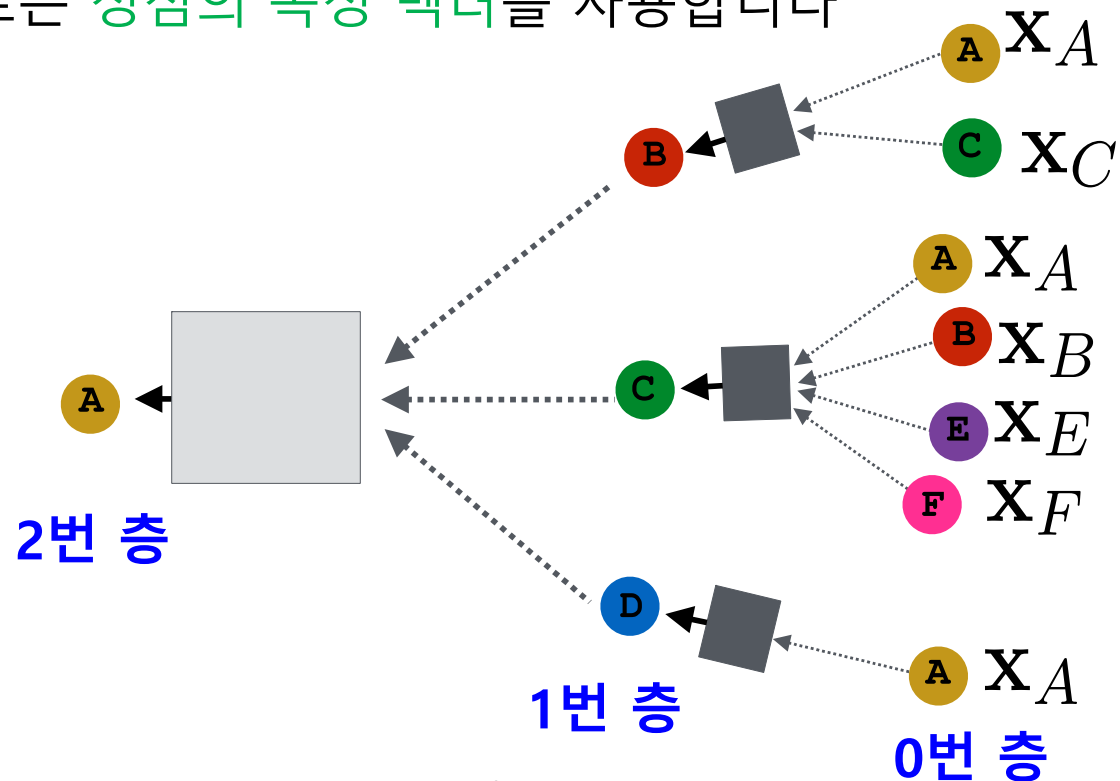
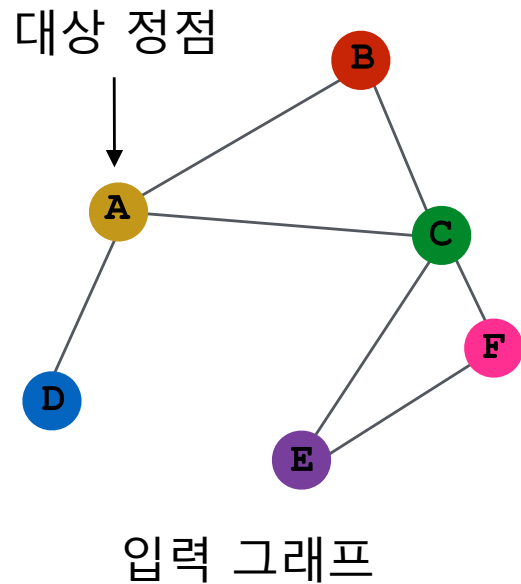
예시에서 **대상 정점**의 임베딩을 얻기 위해 **이웃들** 그리고 **이웃의 이웃들의 정보**를 집계합니다



2.1 그래프 신경망 구조

각 집계 단계를 **층(Layer)**이라고 부르고, 각 층마다 임베딩을 얻습니다

각 층에서는 **이웃들의 이전 층 임베딩을 집계하여** 새로운 임베딩을 얻습니다
0번 층, 즉 입력 층의 임베딩으로는 **정점의 속성 벡터**를 사용합니다

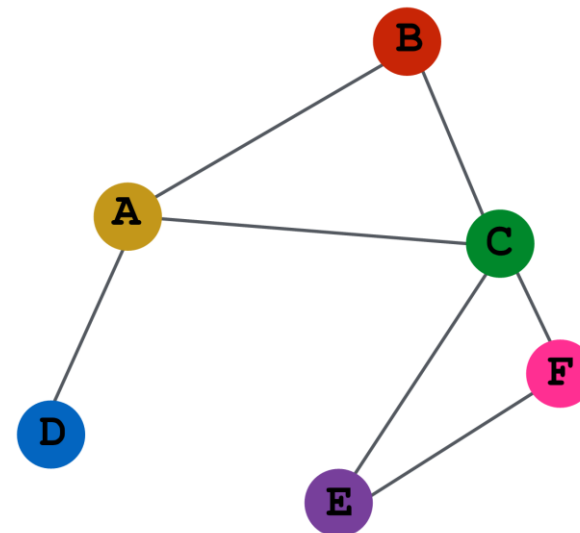
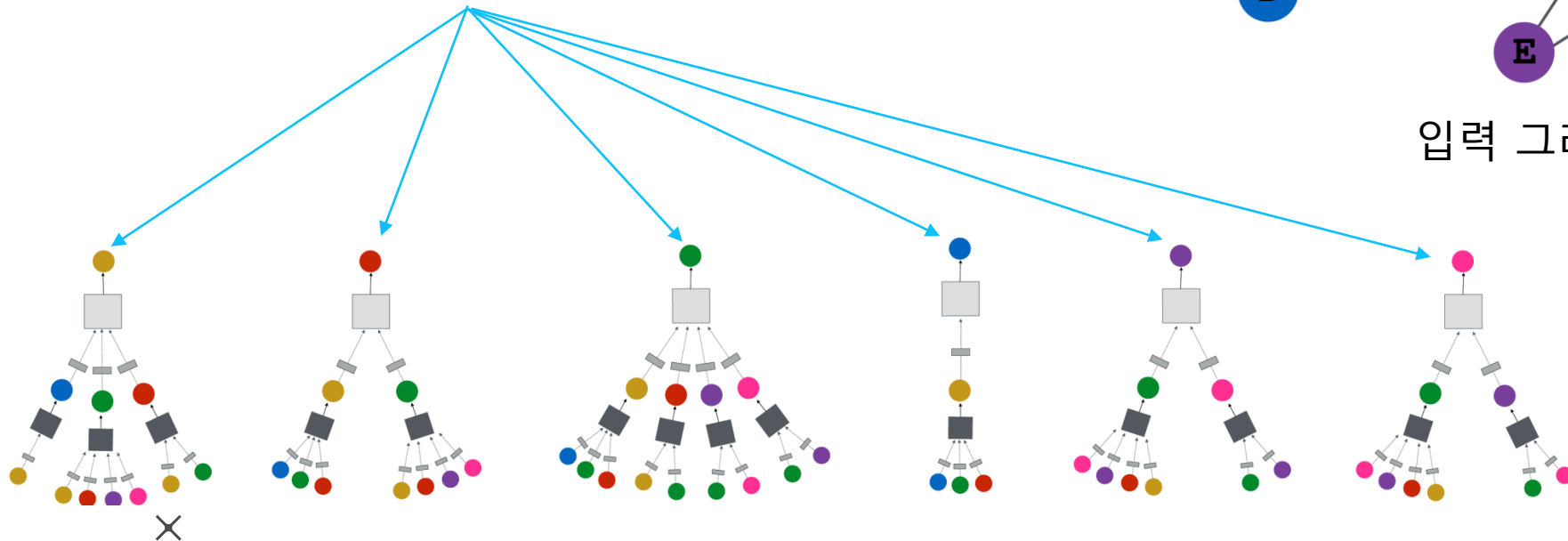


2.1 그래프 신경망 구조

대상 정점 마다 집계되는 정보가 상이합니다

대상 정점 별 집계되는 구조를
계산 그래프(Computation Graph)라고 부릅니다

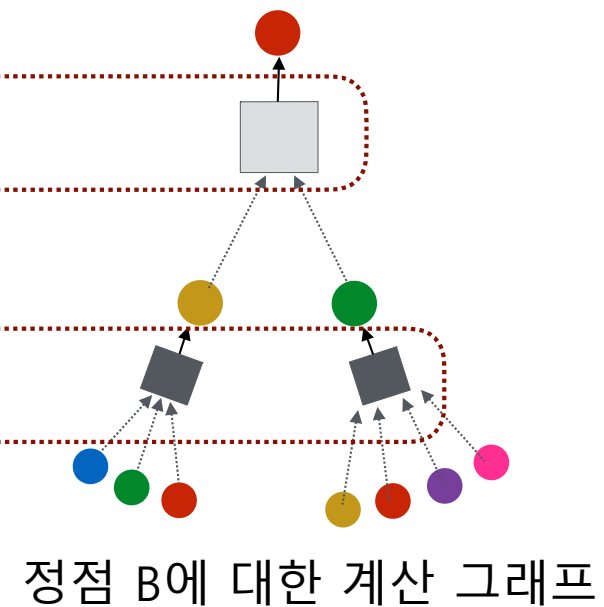
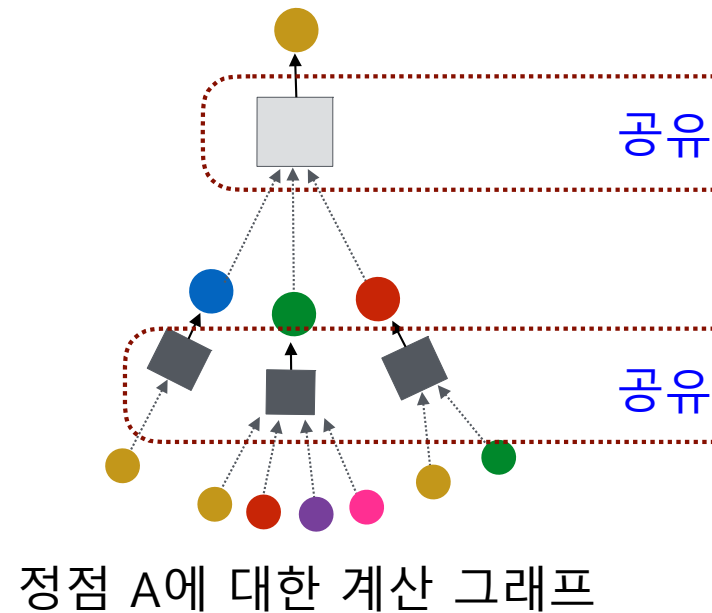
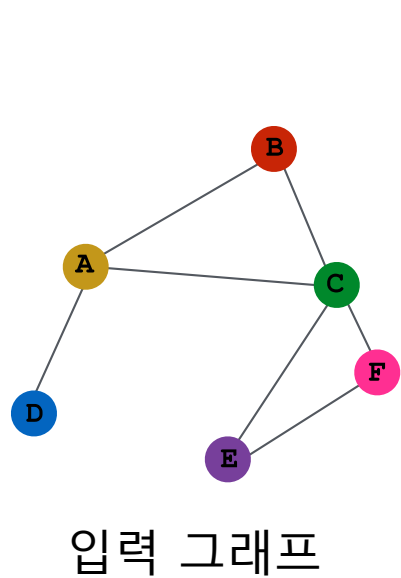
대상 정점 별 계산 그래프



입력 그래프

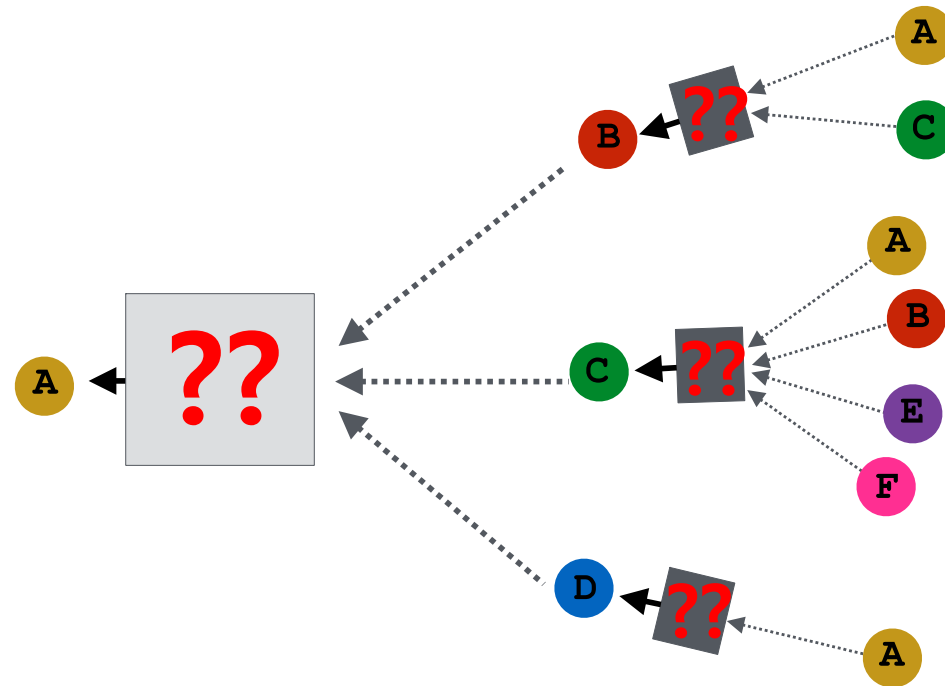
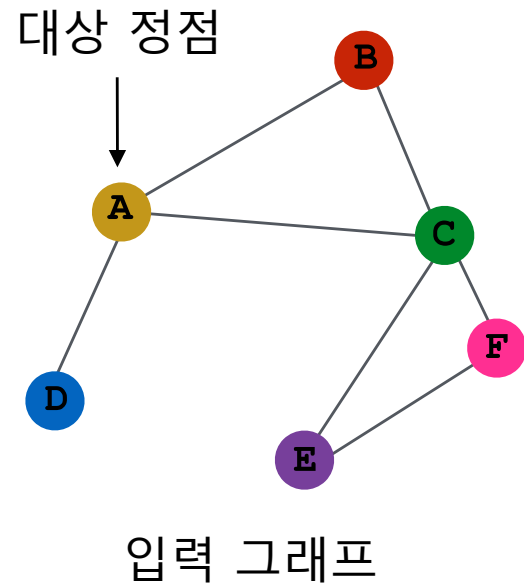
2.1 그래프 신경망 구조

서로 다른 대상 정점간에도 **층 별 집계 함수는 공유**합니다



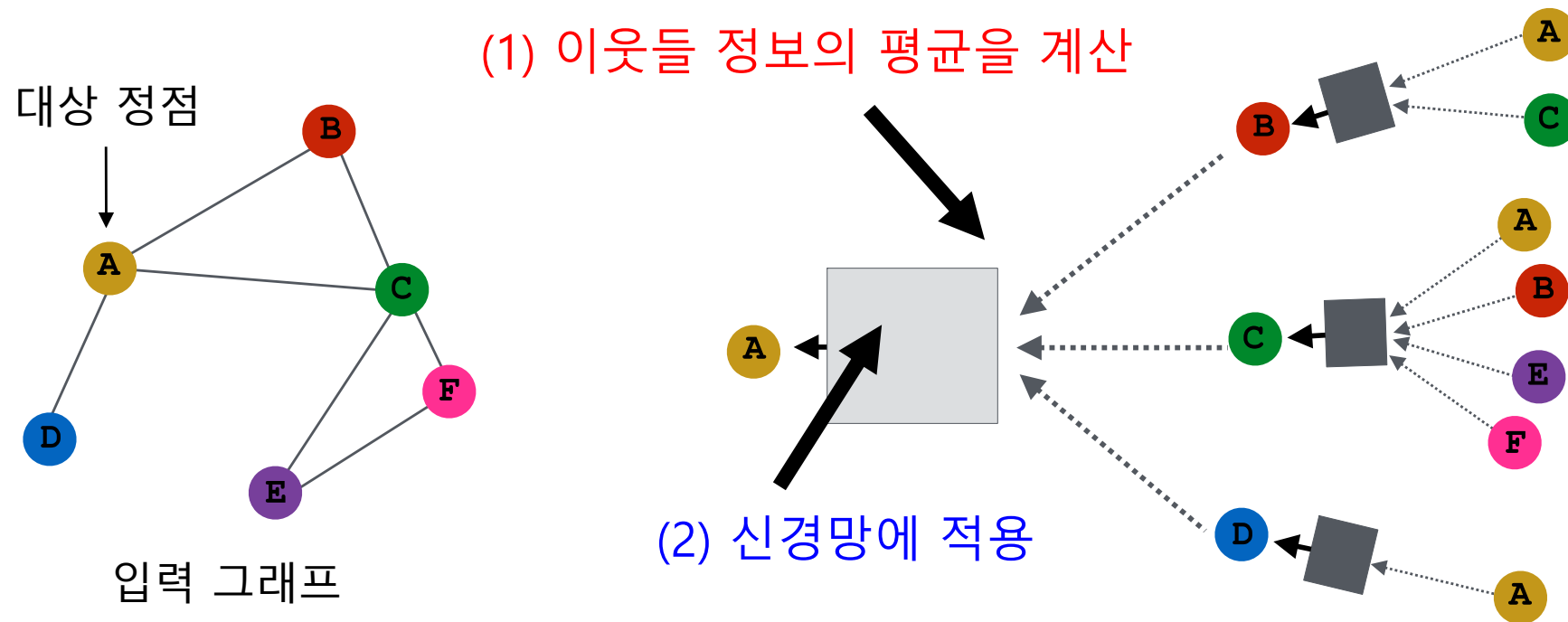
2.1 그래프 신경망 구조

서로 다른 구조의 계산 그래프를 처리하기 위해서는 어떤 형태의 **집계 함수**가 필요할까요?



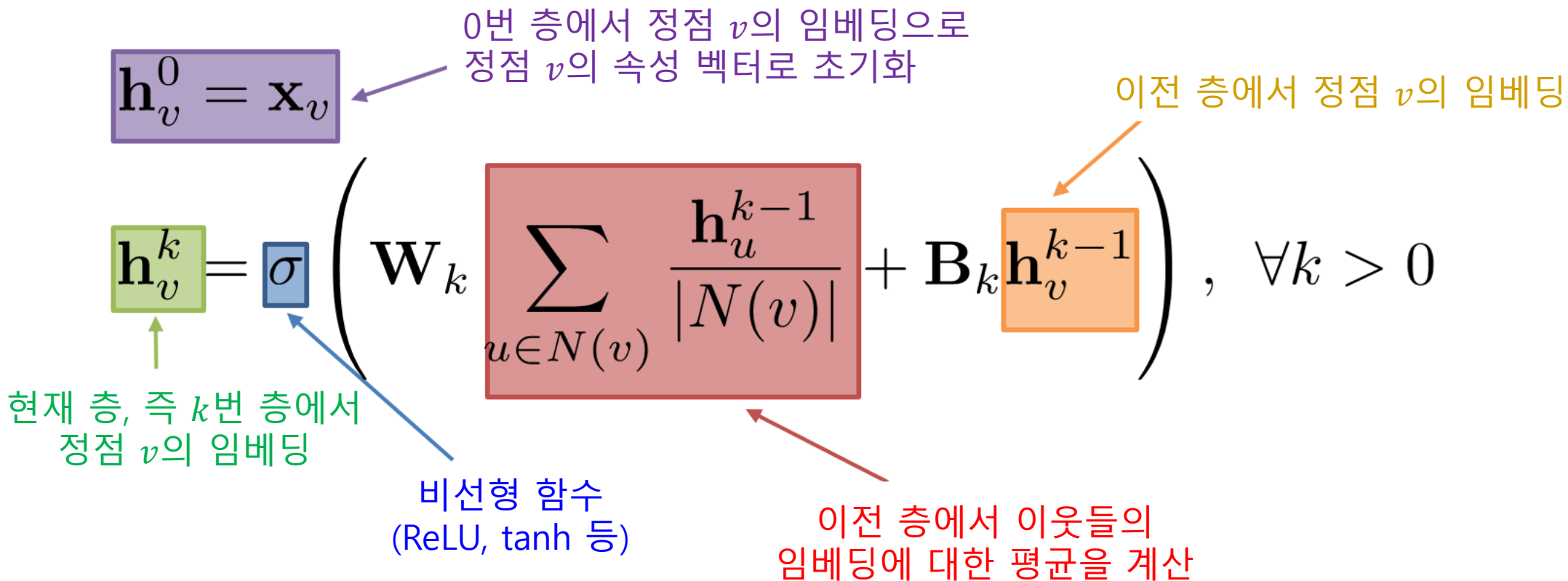
2.1 그래프 신경망 구조

집계 함수는 (1) 이웃들 정보의 평균을 계산하고 (2) 신경망에 적용하는 단계를 거칩니다



2.1 그래프 신경망 구조

집계 함수는 (1) 이웃들 정보의 평균을 계산하고 (2) 신경망에 적용하는 단계를 거칩니다



2.1 그래프 신경망 구조

마지막 층에서의 정점 별 임베딩이 해당 정점의 **출력 임베딩**입니다

0번째 층에서의 임베딩 = 입력 속성 정보

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

$$\mathbf{h}_v^k = \sigma \left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1} \right), \quad \forall k \in \{1, \dots, K\}$$

$$\mathbf{z}_v = \mathbf{h}_v^K$$

마지막 층에서의 임베딩 = 출력 임베딩

2.2 그래프 신경망의 학습

그래프 신경망의 학습 변수(Trainable Parameter)는 층 별 신경망의 가중치입니다

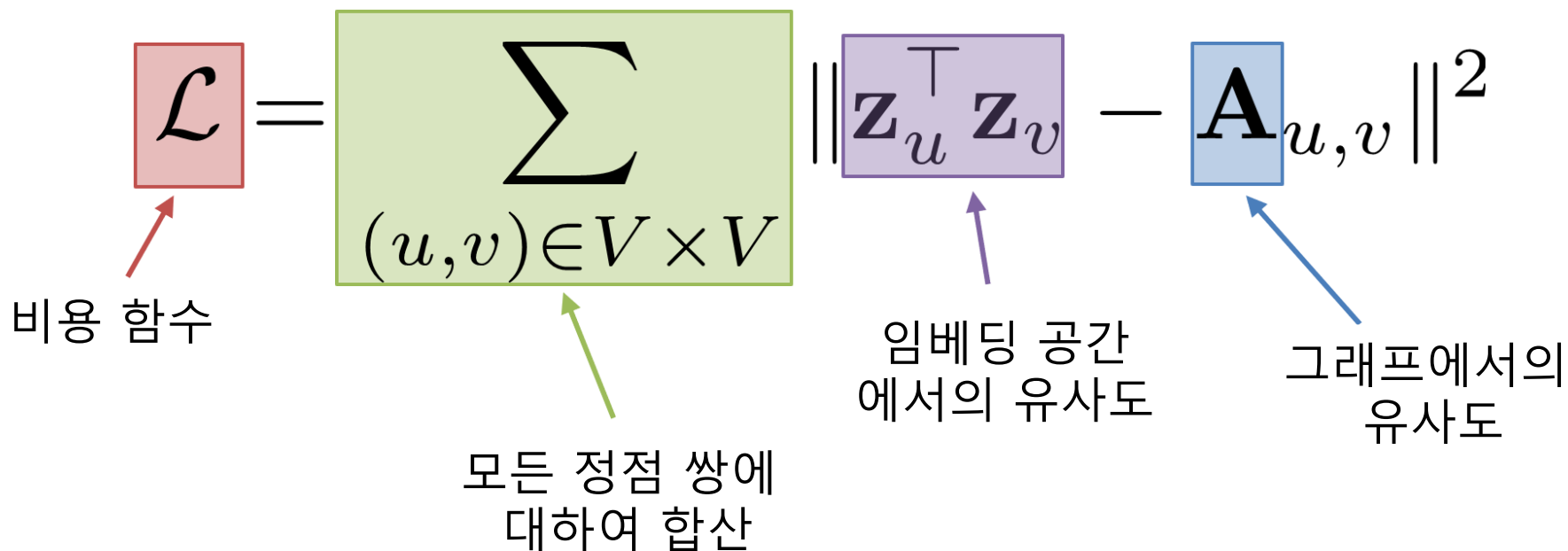
$$\begin{aligned} \mathbf{h}_v^0 &= \mathbf{x}_v \\ \mathbf{h}_v^k &= \sigma \left(\boxed{\mathbf{W}_k} \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \boxed{\mathbf{B}_k} \mathbf{h}_v^{k-1} \right), \quad \forall k \in \{1, \dots, K\} \\ \mathbf{z}_v &= \mathbf{h}_v^K \end{aligned}$$

학습 변수

2.2 그래프 신경망의 학습

먼저 **손실함수**를 결정합니다. **정점간 거리를 “보존”**하는 것을 목표로 할 수 있습니다

변환식 정점 임베딩에서처럼 **그래프에서의 정점간 거리를 “보존”**하는 것을 목표로 할 수 있습니다
만약, 인접성을 기반으로 유사도를 정의한다면, **손실 함수**는 다음과 같습니다


$$\mathcal{L} = \sum_{(u,v) \in V \times V} \| \mathbf{z}_u^\top \mathbf{z}_v - \mathbf{A}_{u,v} \|^2$$

비용 함수

모든 정점 쌍에 대하여 합산

임베딩 공간에서의 유사도

그래프에서의 유사도

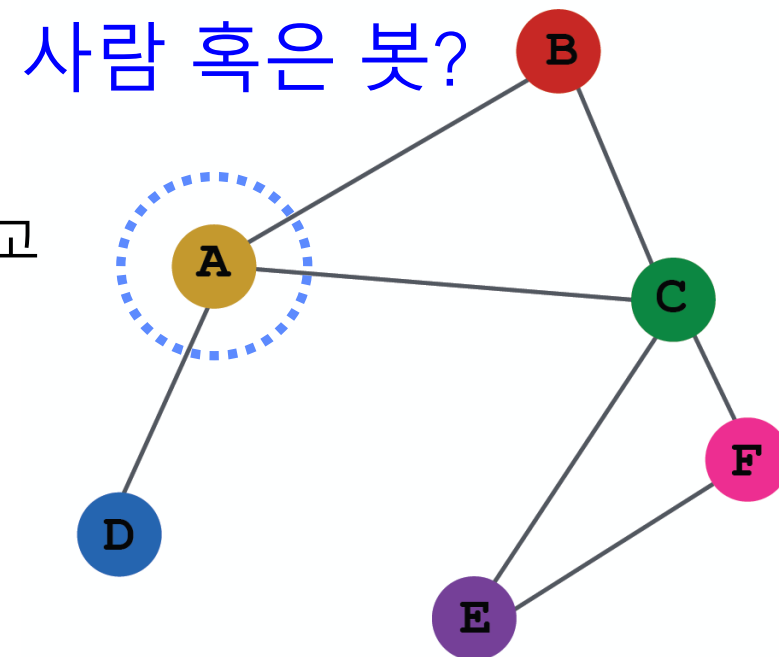
2.2 그래프 신경망의 학습

후속 과제(Downstream Task)의 손실함수를 이용한 종단종(End-to-End) 학습도 가능합니다

정점 분류가 최종 목표인 경우를 생각해봅시다

예를 들어,

- (1) 그래프 신경망을 이용하여 정점의 임베딩을 얻고
- (2) 이를 분류기(Classifier)의 입력으로 사용하여
- (3) 각 정점의 유형을 분류하려고 합니다



2.2 그래프 신경망의 학습

후속 과제(Downstream Task)의 손실함수를 이용한 종단종(End-to-End) 학습도 가능합니다

이 경우 **분류기의 손실함수**, 예를 들어 교차 엔트로피(Cross Entropy)를,
전체 프로세스의 손실함수로 사용하여 **종단종(End-to-End) 학습**을 할 수 있습니다

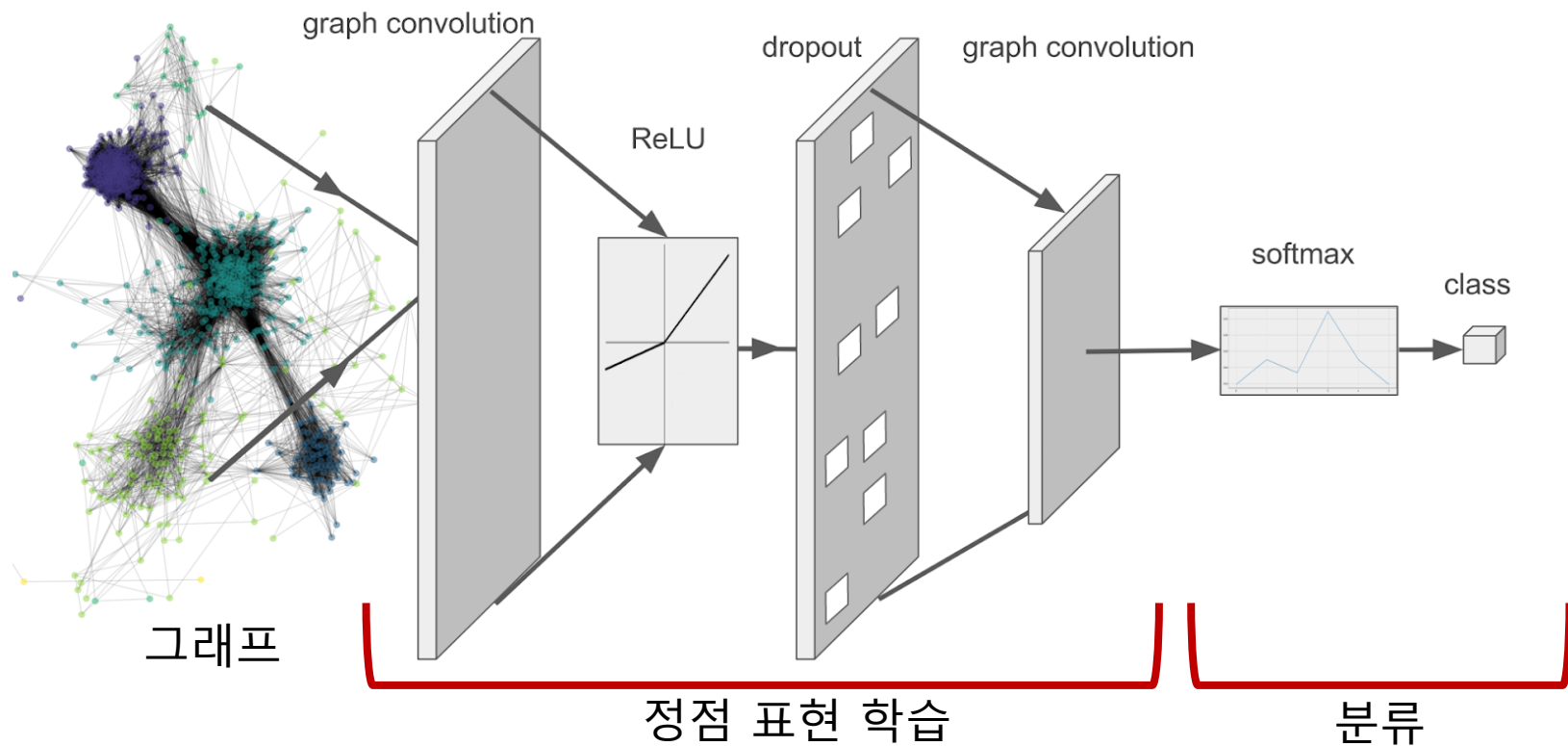
$$\mathcal{L} = \sum_{v \in V} y_v \log(\sigma(\mathbf{z}_v^T \boldsymbol{\theta})) + (1 - y_v) \log(1 - \sigma(\mathbf{z}_v^T \boldsymbol{\theta}))$$

The diagram illustrates the components of the loss function \mathcal{L} with arrows pointing to specific variables:

- A red arrow points from the text "분류기의 학습 변수" (Classification learning variable) to the parameter $\boldsymbol{\theta}$.
- Two blue arrows point from the text "정점의 실제 유형 (0 혹은 1)" (Vertex actual type (0 or 1)) to the variables y_v and $(1 - y_v)$.
- Two purple arrows point from the text "정점의 임베딩" (Vertex embedding) to the vectors \mathbf{z}_v and \mathbf{z}_v .

2.2 그래프 신경망의 학습

후속 과제(Downstream Task)의 손실함수를 이용한 종단종(End-to-End) 학습도 가능합니다



2.2 그래프 신경망의 학습

그래프 신경망과 변환적 정점 임베딩을 이용한 정점 분류

그래프 신경망의 종단종(End-to-End) 학습을 통한 분류는,
변환적 정점 임베딩 이후에 별도의 분류기를 학습하는 것보다 정확도가 대체로 높습니다
아래 표는 다양한 데이터에서의 정점 분류의 정확도(Accuracy)를 보여줍니다

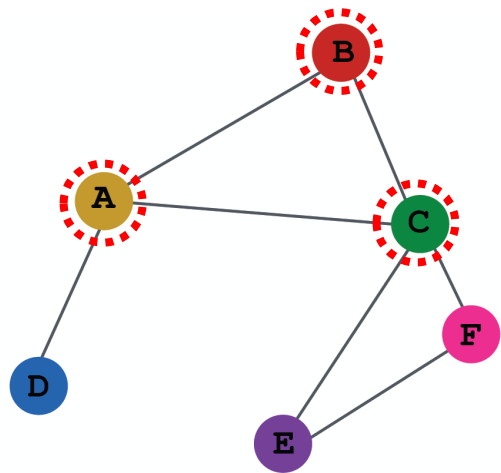
Method	Citeseer	Cora	Pubmed	NELL
ManiReg	60.1	59.5	70.7	21.8
SemiEmb	59.6	59.0	71.1	26.7
LP	45.3	68.0	63.0	26.5
DeepWalk	43.2	67.2	65.3	58.1
Planetoid*	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN	70.3 (7s)	81.5 (4s)	79.0 (38s)	66.0 (48s)

그래프 신경망을 이용한 종단종 학습

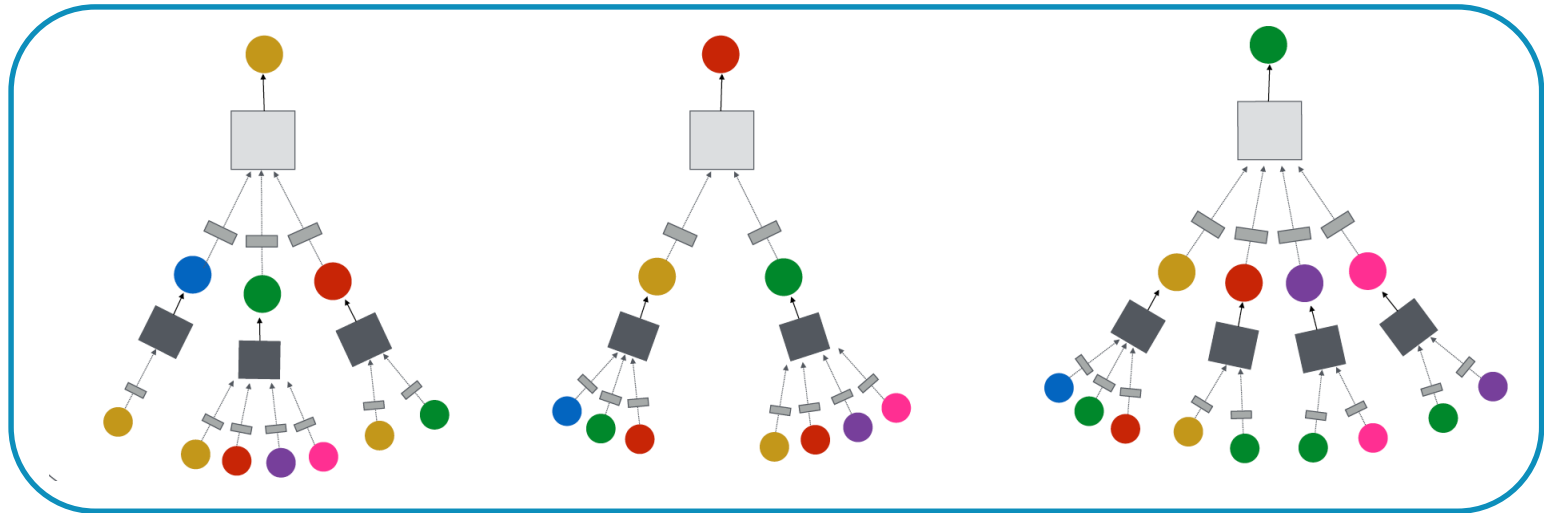
2.2 그래프 신경망의 학습

학습에 사용할 대상 정점을 결정하여 학습 데이터를 구성합니다

선택한 대상 정점들에 대한 계산 그래프를 구성합니다



입력 그래프

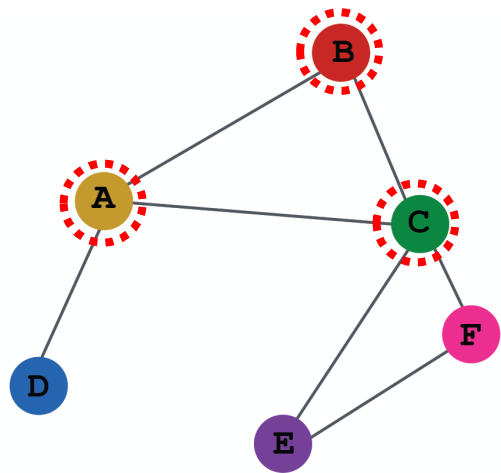


대상 정점에 대한 계산 그래프

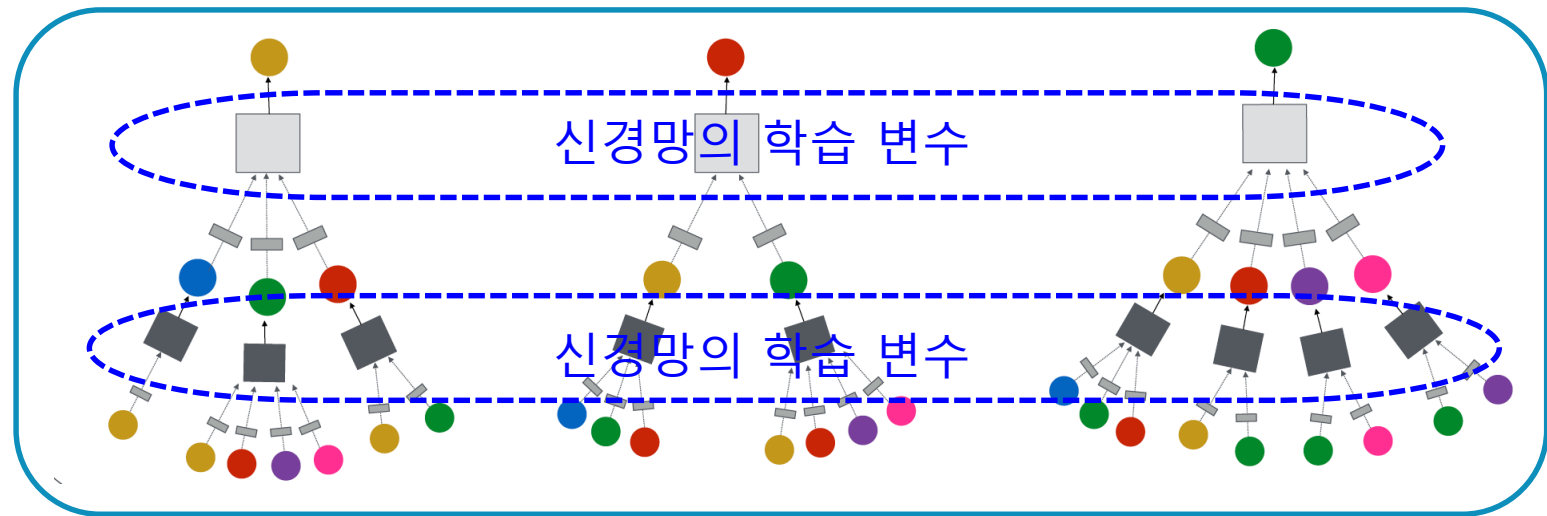
2.2 그래프 신경망의 학습

마지막으로 **오차역전파(Backpropagation)**을 통해 손실함수를 최소화합니다

구체적으로, **오차역전파**를 통해 **신경망의 학습 변수**들을 학습합니다



입력 그래프

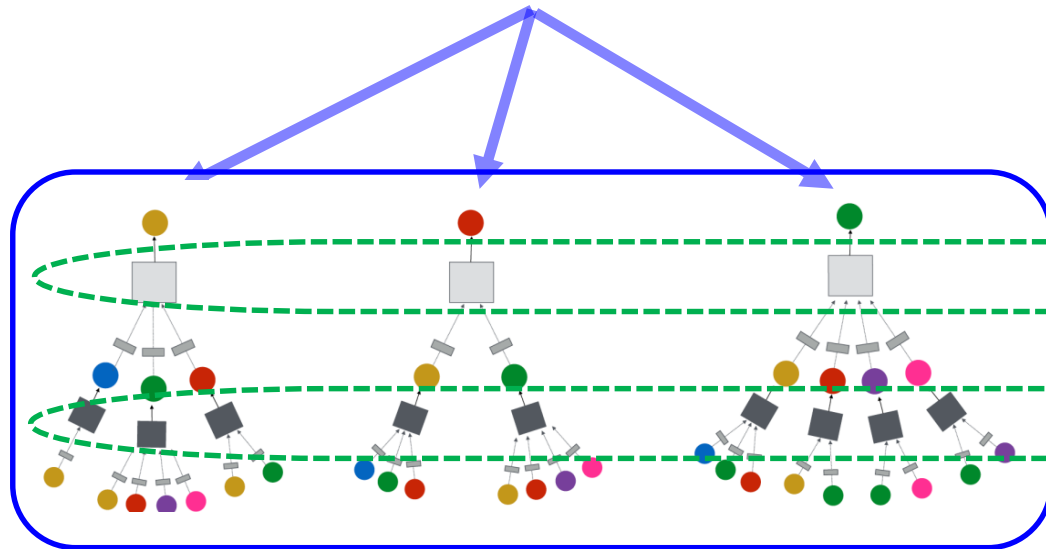


대상 정점에 대한 계산 그래프

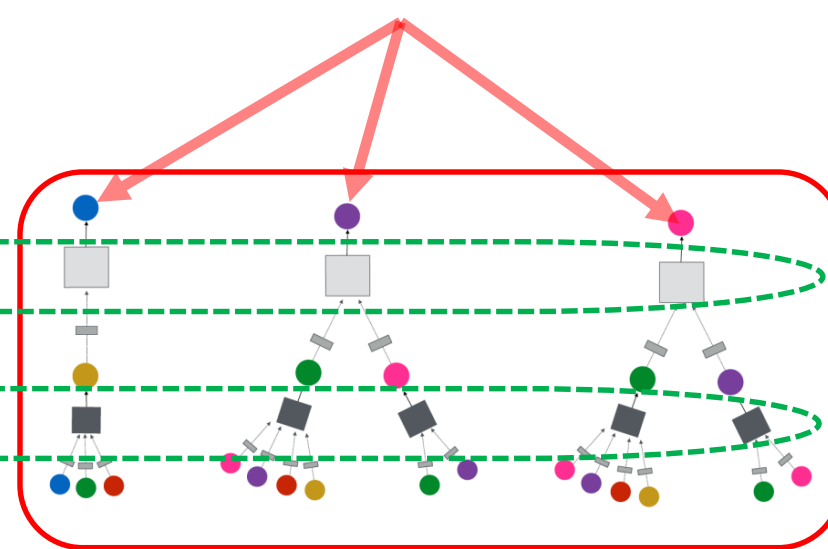
2.3 그래프 신경망의 활용

학습된 신경망을 적용하여, **학습에 사용되지 않은 정점의 임베딩**을 얻을 수 있습니다

학습에 사용된 대상 정점들



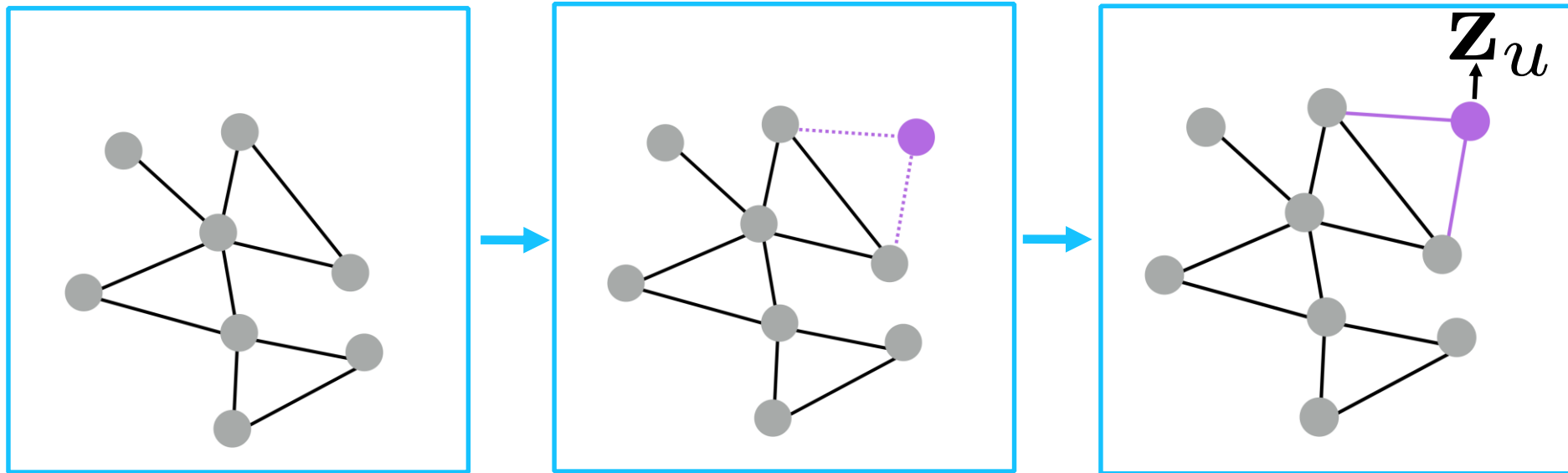
학습에 사용되지 않은 정점들



2.3 그래프 신경망의 활용

마찬가지로, **학습 이후에 추가된 정점의 임베딩도 얻을 수 있습니다**

온라인 소셜네트워크 등 많은 실제 그래프들은 시간에 따라서 변화합니다



학습 당시의 입력 그래프

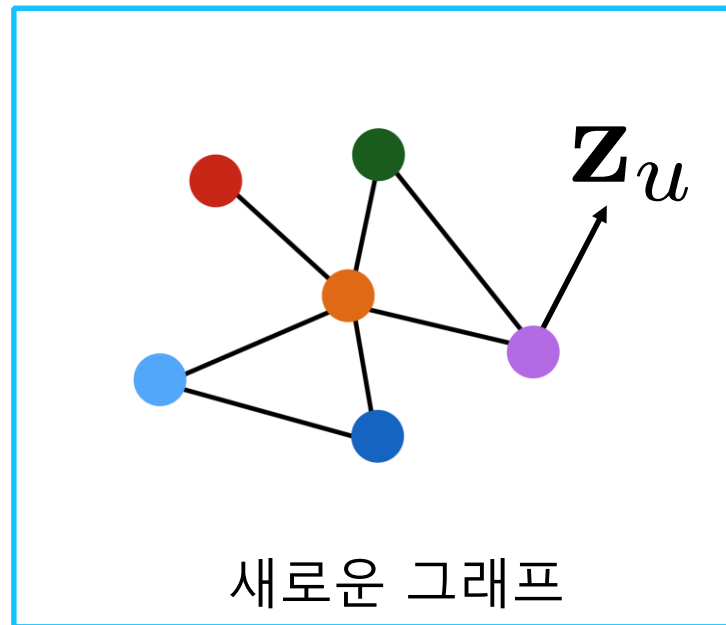
새로운 정점 도착

그래프 신경망 적용,
새로운 정점 임베딩

2.3 그래프 신경망의 활용

학습된 그래프 신경망을, **새로운 그래프에 적용**할 수도 있습니다

예를 들어, A종의 단백질 상호 작용 그래프에서 학습한 그래프 신경망을 B종의 단백질 상호작용 그래프에 적용할 수 있습니다



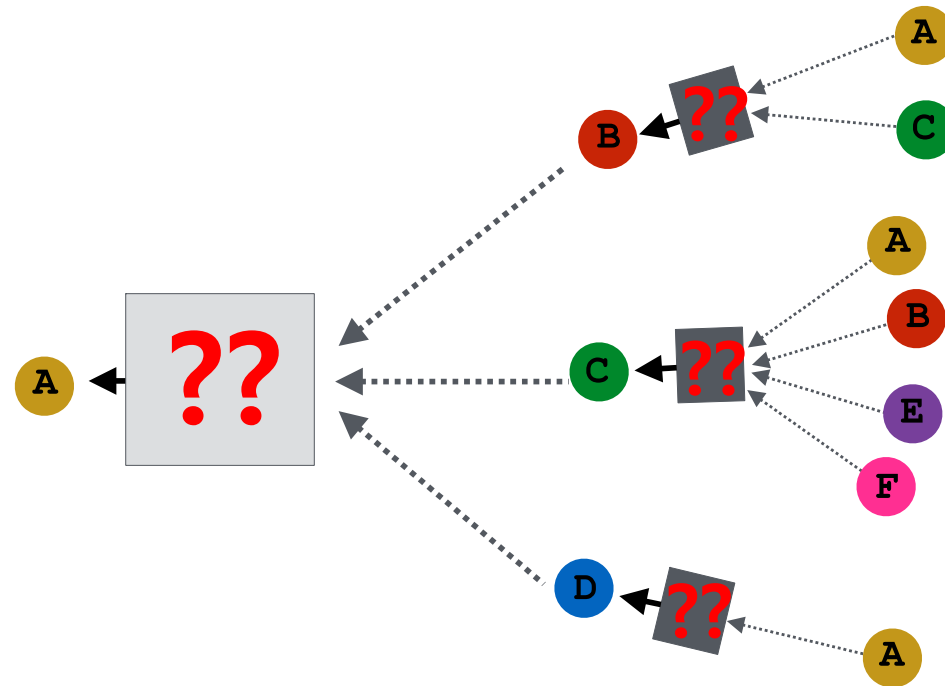
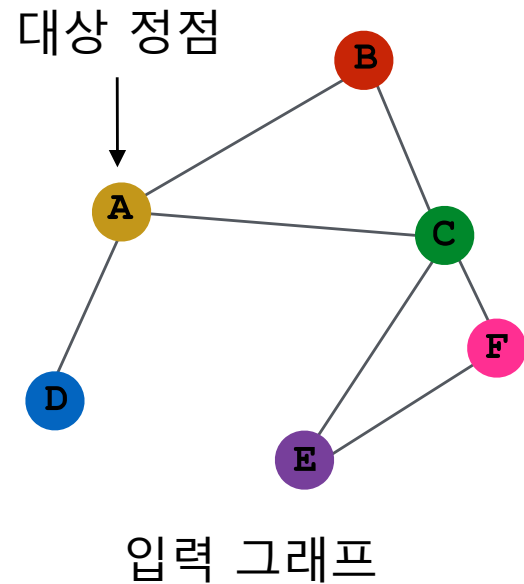
3. 그래프 신경망 변형

3.1 그래프 합성곱 신경망

3.2 GraphSAGE

3.1 그래프 합성곱 신경망

소개한 것 이외에도 다양한 형태의 **집계 함수**를 사용할 수 있습니다



3.1 그래프 합성곱 신경망

그래프 합성곱 신경망(Graph Convolutional Network, GCN)의 집계 함수입니다

$$\begin{aligned}\mathbf{h}_v^0 &= \mathbf{x}_v \\ \mathbf{h}_v^k &= \sigma \left(\mathbf{W}_k \sum_{u \in N(v) \cup v} \frac{\mathbf{h}_u^{k-1}}{\sqrt{|N(u)| |N(v)|}} \right), \quad \forall k \in \{1, \dots, K\} \\ \mathbf{z}_v &= \mathbf{h}_v^K\end{aligned}$$

3.1 그래프 합성곱 신경망

기존의 집계 함수와 비교해보겠습니다. 작은 차이지만 큰 성능의 향상으로 이어지기도 합니다

$$\mathbf{h}_v^k = \sigma \left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1} \right)$$

기존 집계 함수

VS.

동일 신경망 사용으로 학습 변수 공유

$$\mathbf{h}_v^k = \sigma \left(\mathbf{W}_k \sum_{u \in N(v) \cup v} \frac{\mathbf{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}} \right)$$

GCN의 집계 함수

정규화 방법 변화

3.2 GraphSAGE

GraphSAGE의 집계 함수입니다

이웃들의 임베딩을 AGG 함수를 이용해 합친 후,
자신의 임베딩과 연결(Concatenation)하는 점이 독특합니다

$$\mathbf{h}_v^k = \sigma \left(\left[\mathbf{W}_k \cdot \text{AGG} \left(\{\mathbf{h}_u^{k-1}, \forall u \in N(v)\} \right), \mathbf{B}_k \mathbf{h}_v^{k-1} \right] \right)$$

자신의 임베딩과 이웃의 임베딩을 연결

3.2 GraphSAGE

AGG 함수로는 평균, 풀링, LSTM 등이 사용될 수 있습니다

Mean: $AGG = \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}$ 원소별 최대

Pool: $AGG = \gamma(\{\mathbf{Q}\mathbf{h}_u^{k-1}, \forall u \in N(v)\})$

LSTM: $AGG = \text{LSTM}([\mathbf{h}_u^{k-1}, \forall u \in \pi(N(v))])$

4. 합성곱 신경망과의 비교

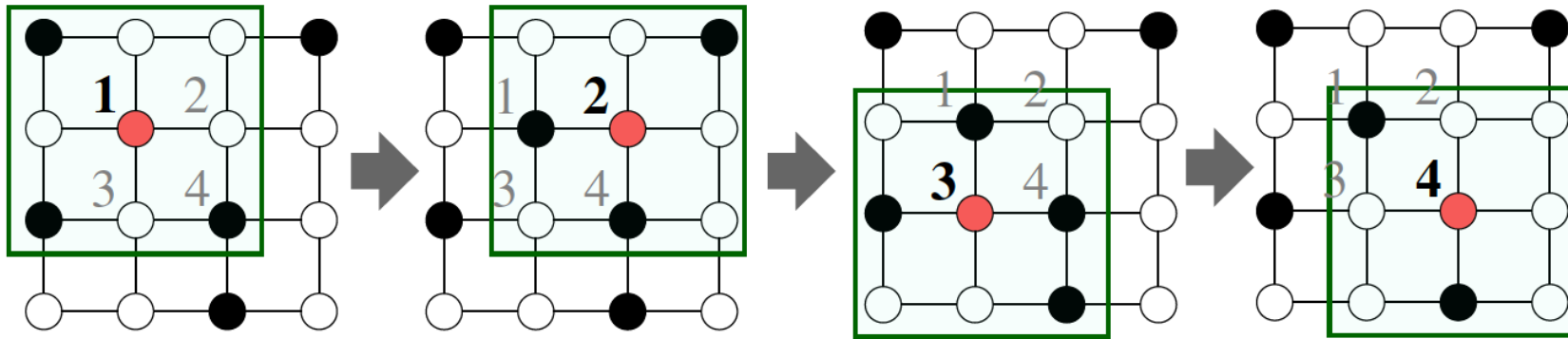
4.1 합성곱 신경망과 그래프 신경망의 유사성

4.2 합성곱 신경망과 그래프 신경망의 차이

4.1 합성곱 신경망과 그래프 신경망의 유사성

합성곱 신경망과 그래프 신경망은 모두 이웃의 정보를 집계하는 과정을 반복합니다

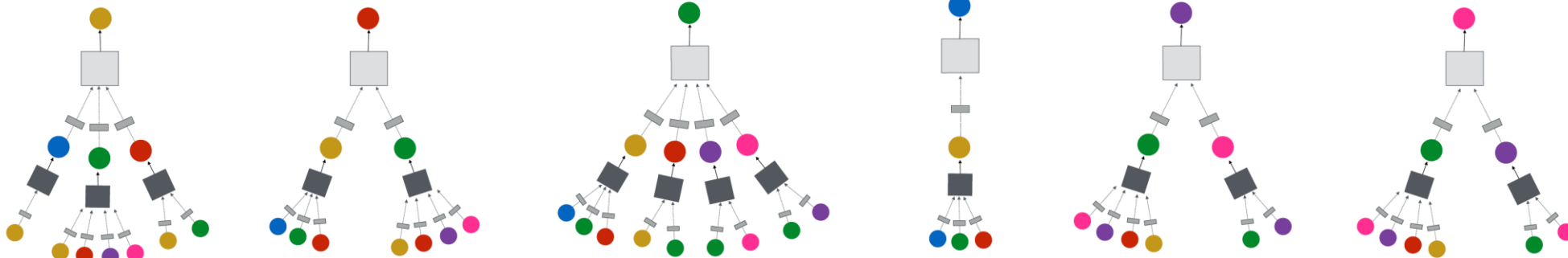
구체적으로, 합성곱 신경망은 이웃 픽셀의 정보를 집계하는 과정을 반복합니다



4.2 합성곱 신경망과 그래프 신경망의 차이

합성곱 신경망에서는 이웃의 수가 균일하지만, 그래프 신경망에서는 아닙니다

그래프 신경망에서는 정점 별로 집계하는 이웃의 수가 다릅니다



4.2 합성곱 신경망과 그래프 신경망의 차이

그래프의 인접 행렬에 합성곱 신경망을 적용하면 효과적일까요?

그래프에는 합성곱 신경망이 아닌 그래프 신경망을 적용하여야 합니다!
많은 분들이 흔히 범하는 실수입니다

합성곱 신경망이 주로 쓰이는 이미지에서는
인접 픽셀이 유용한 정보를 담고 있을 가능성이 높습니다

하지만, 그래프의 인접 행렬에서의 인접 원소는 제한된 정보를 가집니다
특히나, 인접 행렬의 행과 열의 순서는 임의로 결정되는 경우가 많습니다

5. 실습: DGL 라이브러리와 GraphSAGE를 이용한 정점 분류

5.1 데이터 불러오기

5.2 GraphSAGE 정의

5.3 GraphSAGE 학습

5.4 GraphSAGE 평가

5.1 데이터 불러오기

필요한 라이브러리를 불러옵니다

```
import numpy as np
import time
import torch
import torch.nn as nn
import torch.nn.functional as F
import dgl
from dgl.data import CoraGraphDataset
from dgl.nn.pytorch.conv import SAGEConv
from sklearn.metrics import f1_score
```

5.1 데이터 불러오기

실습에 사용할 Cora 인용 그래프를 불러옵니다

Cora 데이터셋은 2,708개의 정점(논문)과, 10,556개의 간선(인용 관계)로 구성됩니다
각 정점은 1,433개의 속성을 가지며, 이는 1433개의 단어의 등장 여부를 의미합니다
각 정점은 7개의 유형 중 하나를 가지며, 대응되는 논문의 주제를 의미합니다
학습은 140개의 정점을 사용해 이루어집니다

```
G = CoraGraphDataset()  
numClasses = G.num_classes  
G = G[0]  
features = G.ndata['feat']  
inputFeatureDim = features.shape[1]  
labels = G.ndata['label']  
trainMask = G.ndata['train_mask']  
testMask = G.ndata['test_mask']
```

```
Finished data loading and preprocessing.  
NumNodes: 2708  
NumEdges: 10556  
NumFeats: 1433  
NumClasses: 7  
NumTrainingSamples: 140  
NumValidationSamples: 500  
NumTestSamples: 1000
```

5.2 GraphSAGE 정의

GraphSAGE 정의의 첫 단계로 각 층을 정의합니다

```
class GraphSAGE(nn.Module):
    def __init__(self, graph, inFeatDim, numHiddenDim, numClasses, numLayers,
                  activationFunction, dropoutProb, aggregatorType):
        super(GraphSAGE, self).__init__()
        self.layers = nn.ModuleList()
        self.graph = graph
        self.layers.append(SAGEConv(inFeatDim, numHiddenDim, aggregatorType,
                                    dropoutProb, activationFunction))

        for i in range(numLayers):
            self.layers.append(SAGEConv(numHiddenDim, numHiddenDim, aggregatorType,
                                        dropoutProb, activationFunction))

        self.layers.append(SAGEConv(numHiddenDim, numClasses, aggregatorType,
                                    dropoutProb, activation=None))
```

5.2 GraphSAGE 정의

GraphSAGE 정의의 다음 단계로 순전파(Forward Propagation)을 정의합니다

```
class GraphSAGE(nn.Module):
    def __init__(self, graph, inFeatDim, numHiddenDim, numClasses, numLayers,
        ...

    def forward(self, features):
        x = features
        for layer in self.layers:
            x = layer(self.graph, x)
        return x
```

```
model = GraphSAGE(G, inputFeatureDim, numHiddenDim, numClasses, numLayers,
    F.relu, dropoutProb, aggregatorType)
```

5.3 GraphSAGE 학습

역전파(Backpropagation)을 통해 GraphSAGE를 학습합니다

```
def train(model, lossFunction, features, labels, trainMask, optimizer, numEpochs):  
    for epoch in range(numEpochs):  
        model.train()  
        logits = model(features)  
        loss = lossFunction(logits[trainMask], labels[trainMask])  
        optimizer.zero_grad()  
        loss.backward()  
        optimizer.step()
```

```
lossFunction = torch.nn.CrossEntropyLoss()  
optimizer = torch.optim.Adam(model.parameters(), lr=learningRate, weight_decay=weightDecay)  
train(model, lossFunction, features, labels, trainMask, optimizer, numEpochs)
```

5.4 GraphSAGE 평가

정점 분류의 성능을 평가합니다

```
def evaluateTest(model, features, labels, mask):  
    model.eval()  
    with torch.no_grad():  
        logits = model(features)  
        logits = logits[mask]  
        labels = labels[mask]  
        _, indices = torch.max(logits, dim=1)  
        macro_f1 = f1_score(labels, indices, average = 'macro')  
        correct = torch.sum(indices == labels)  
        return correct.item() * 1.0 / len(labels), macro_f1
```

5.4 GraphSAGE 평가

정점 분류의 성능을 평가합니다

```
def test(model, features, labels, testMask):  
    acc, macro_f1 = evaluateTest(model, features, labels, testMask)  
  
test(model, features, labels, testMask)
```

Test Accuracy 0.7750

Test macro-f1 0.7682

9강 정리

1. 정점 표현 학습

- 그래프의 정점들을 벡터로 표현하는 것
- 그래프에서 정점 사이의 유사성을 계산하는 방법에 따라 여러 접근법이 구분됨
- 그래프 신경망 등의 귀납식 정점 표현 학습은 임베딩 함수를 출력으로 얻음

2. 그래프 신경망 기본

- 그래프 신경망은 이웃 정점들의 정보를 집계하는 과정을 반복하여 임베딩을 얻음
- 후속 과제의 손실함수를 사용해 종단중 학습이 가능함
- 학습된 그래프 신경망을 학습에서 제외된 정점, 새롭게 추가된 정점, 새로운 그래프에 적용 가능

3. 그래프 신경망 변형

4. 합성곱 신경망과의 비교

- 그래프 형태의 데이터에는 합성곱 신경망이 아닌 그래프 신경망을 사용해야 효과적

5. 실습: DGL 라이브러리와 GraphSAGE를 이용한 정점 분류