CMLG Course Lab II

Siqi Miao, Haoyu Wang
CS@Purdue

Outline

- How to store graph data for ML?
- How to implement GNNs for different tasks?
- Q & A

How to store graph data for ML?

How to store image data?

- A sample:
    - $x = ?, y = ?$
    - $x \in \mathbb{R}^{?}, y \in \mathbb{Z}^{+?}$

How to store graph data for ML?

How to store image data?

- A sample:
  - $x = ?, y = ?$
  - $x =$ $y =$ Dog (or, the kth class)
  - $x \in \mathbb{R}^?, y \in \mathbb{Z}^{+?}$

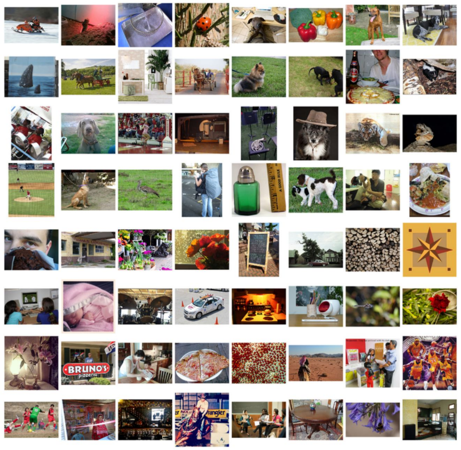
How to store graph data for ML?

How to store image data?

- A sample:
  - $x = ?, y = ?$
  - $x =$ $y =$ Dog (or, the kth class)
  - $x \in \mathbb{R}^?, y \in \mathbb{Z}^{+?}$
  - $x \in \mathbb{R}^{W \times H \times 3}, y \in \mathbb{Z}^+$

How to store graph data for ML?

How to store image data?

- The entire dataset with ***N*** samples:
  - $x =$  $y =$ all ***N*** labels
  - $x \in \mathbb{R}^?, y \in \mathbb{Z}^{+?}$

How to store graph data for ML?

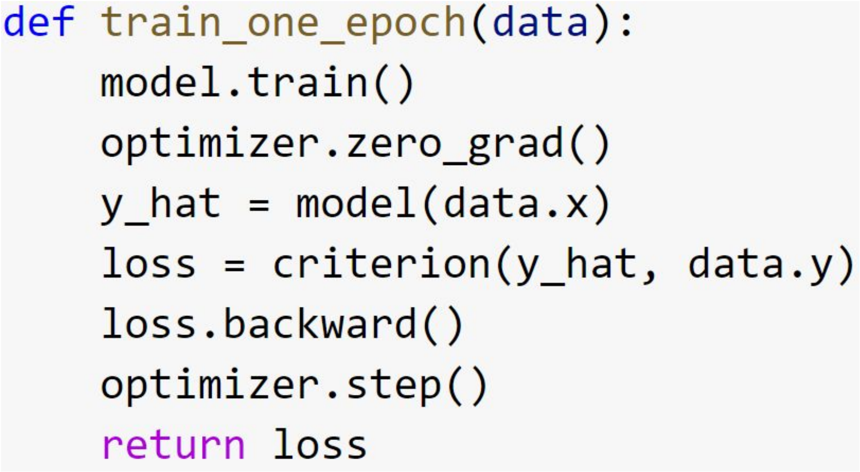
How to train an image classifier?

- The entire dataset with **N** samples:
  - CNN: f , images: x , labels: y
  - ***1. Forward pass:***
    - $\hat{y} = f(x)$
  - ***2. Compute loss***
    - $\text{loss} = \mathcal{L}(y, \hat{y})$
  - ***3. Backward pass***
    - $\text{Grad} = \frac{d\mathcal{L}}{df}$
  - ***4. Update*** f ***using*** Grad

How to store graph data for ML?

How to train an image classifier?

- The entire dataset with <u>**N**</u> samples:
  - CNN: f , images: x , labels: y
  - ***<u>1. Forward pass:</u>***
    - $\hat{y} = f(x)$
  - ***<u>2. Compute loss</u>***
    - $\text{loss} = \mathcal{L}(y, \hat{y})$
  - ***<u>3. Backward pass</u>***
    - $\text{Grad} = \frac{d\mathcal{L}}{df}$
  - ***<u>4. Update f using</u>*** Grad



```python
def train_one_epoch(data):
    model.train()
    optimizer.zero_grad()
    y_hat = model(data.x)
    loss = criterion(y_hat, data.y)
    loss.backward()
    optimizer.step()
    return loss
```

It's impractical to do full-batch training when <u>N</u> is large!

How to store graph data for ML?

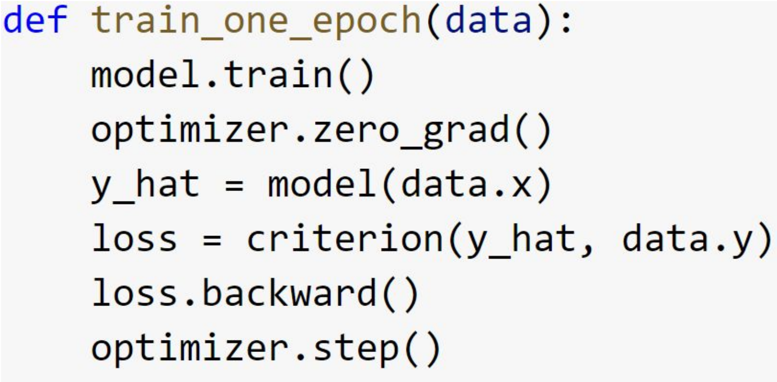
How to store image data?

- A batch of ***B*** samples:
  - $x =$ $y =$ all ***B*** labels
  - $x \in \mathbb{R}^?, y \in \mathbb{Z}^{+?}$

How to store graph data for ML?

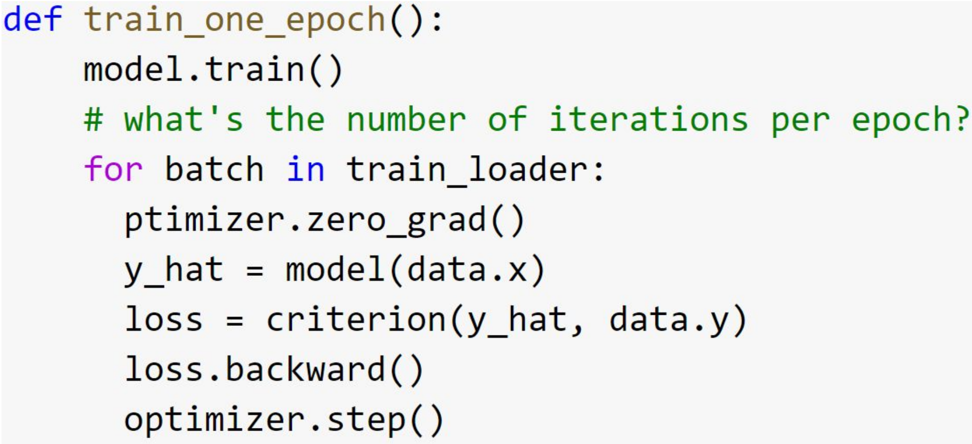
How to train an image classifier?

- A batch has ***B*** samples; the dataset has ***N*** samples:



```
def train_one_epoch(data):
    model.train()
    optimizer.zero_grad()
    y_hat = model(data.x)
    loss = criterion(y_hat, data.y)
    loss.backward()
    optimizer.step()
```

Full-batch training



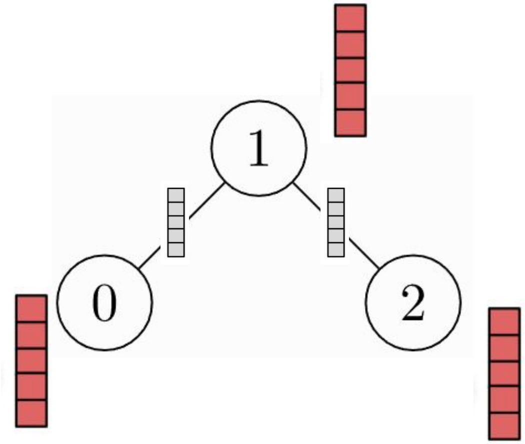
```
def train_one_epoch():
    model.train()
    # what's the number of iterations per epoch?
    for batch in train_loader:
      ptimizer.zero_grad()
      y_hat = model(data.x)
      loss = criterion(y_hat, data.y)
      loss.backward()
      optimizer.step()
```

Mini-batch training

How to store graph data for ML?

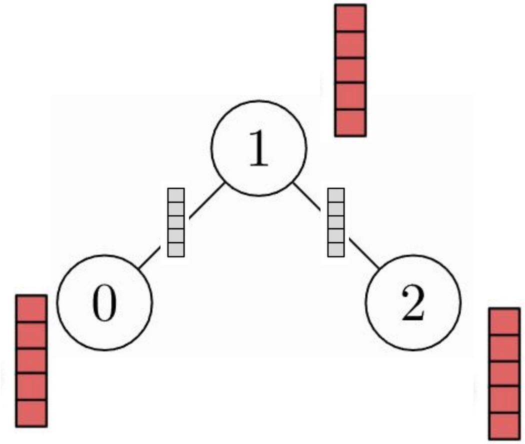
How to store graph data?

- A graph with ***V*** nodes and ***E*** edges:
  - $x = ?, y = ?$
  - What should be included as features?



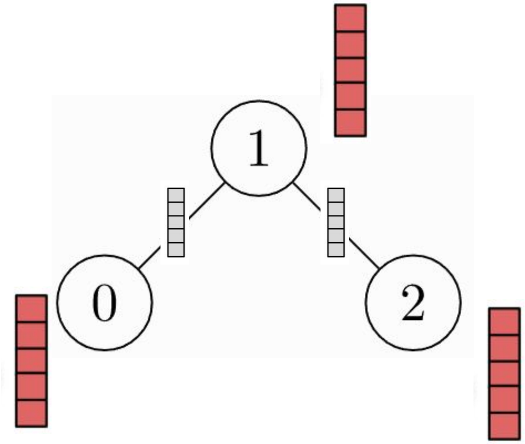
How to store graph data for ML?

How to store graph data?



- A graph with ***V*** nodes and ***E*** edges:
  - $x =?, y =?$
  - What should be included as features?
    - Node features $x \in \mathbb{R}^?$
    - Edge features $\text{edge\_attr} \in \mathbb{R}^?$
    - Adjacency matrix…?
      - Edge list! $\text{edge\_index} \in \mathbb{Z}^{+?}$

How to store graph data for ML?



How to store graph data?

- A graph with ***V*** nodes and ***E*** edges:
  - $x = ?, y = ?$
  - What should be included as features?
    - Node features $x \in \mathbb{R}^?$
    - Edge features $\text{edge\_attr} \in \mathbb{R}^?$
    - Adjacency matrix...?
      - Edge list! $\text{edge\_index} \in \mathbb{Z}^{+?}$
  - What can y be?

How to store graph data for ML?

How to store graph data?

- A graph with ***V*** nodes and ***E*** edges:
  - What can y be?
    - Node classification labels

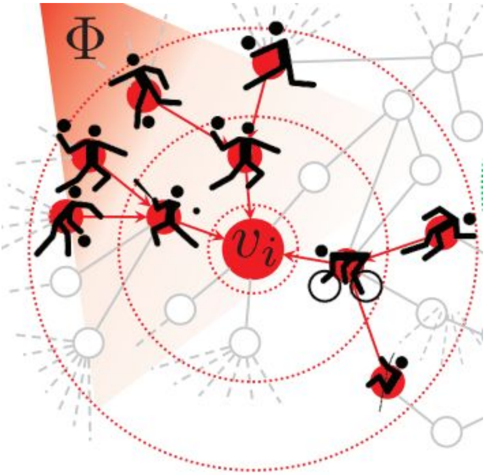


Image src: link

How to store graph data for ML?

How to store graph data?

- A graph with ***V*** nodes and ***E*** edges:
  - What can y be?
    - Link prediction labels

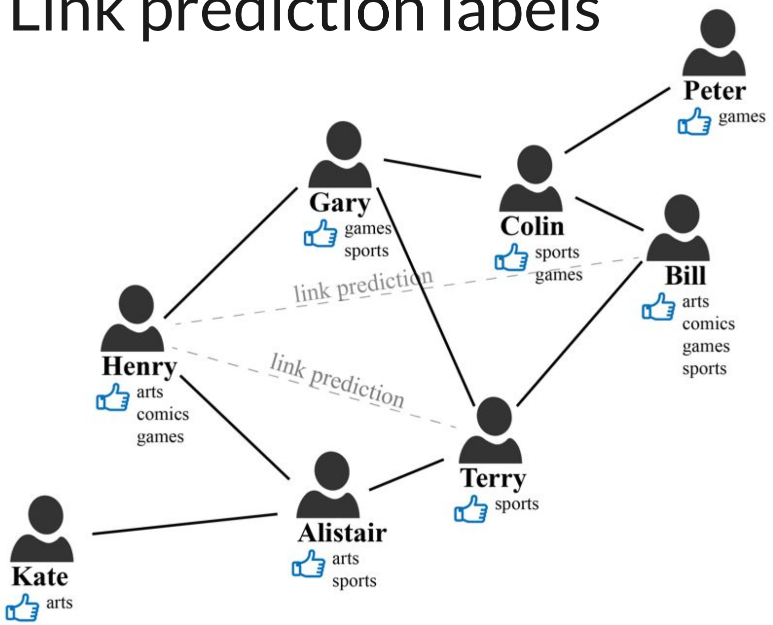


Image src: link

How to store graph data for ML?

How to store graph data?

- A graph with ***V*** nodes and ***E*** edges:
  - What can y be?
    - Graph classification labels

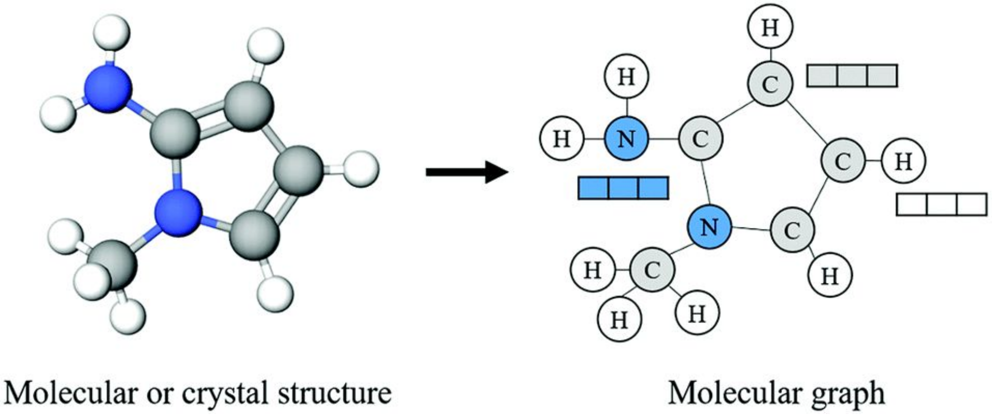


Image src: link

How to store graph data for ML?

How to store graph data?

- A graph with ***V*** nodes and ***E*** edges:
  - What can y be?
    - Node classification labels

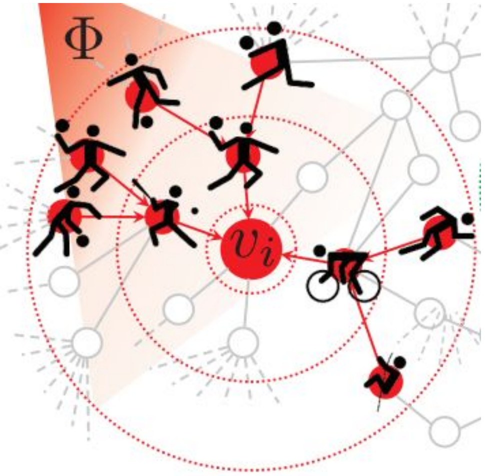


Image src: link

How to store graph data for ML?

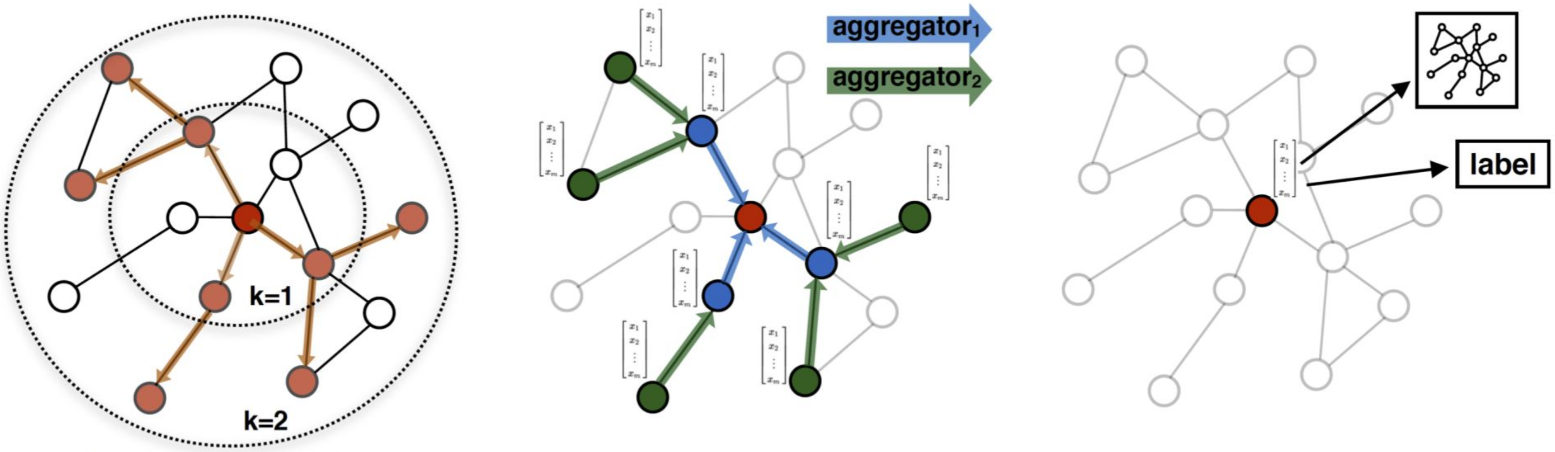
How to train a node classifier?

- A graph with ***V*** nodes and ***E*** edges:
  - Full-batch training
    - What if ***V*** is 1 million?
  - Mini-batch training
    - How?

How to store graph data for ML?

How to train a node classifier?

- A graph with ***V*** nodes and ***E*** edges:
  - Mini-batch training



1. Sample neighborhood

2. Aggregate feature information from neighbors

3. Predict graph context and label using aggregated information

Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

Image src: link

How to store graph data for ML?

How to store graph data?

- A graph with ***V*** nodes and ***E*** edges:
  - What can y be?
    - Graph classification labels

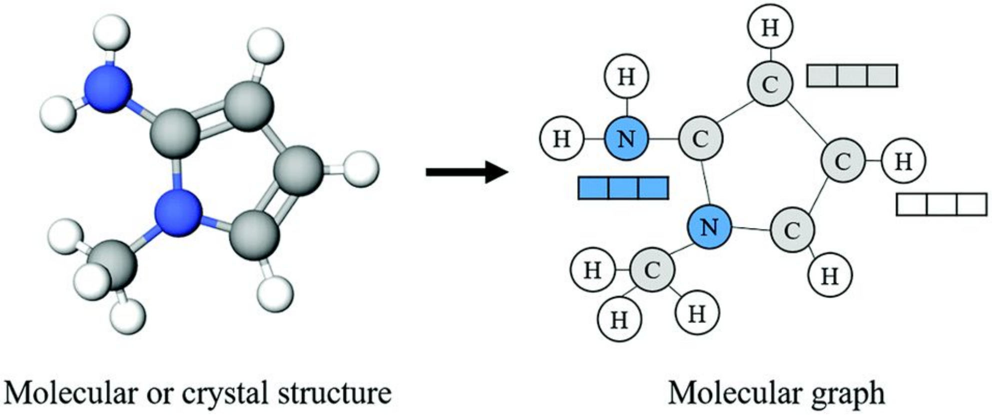
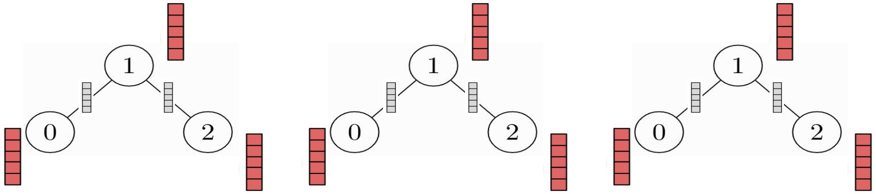


Image src: link

How to store graph data for ML?

How to store graph data?

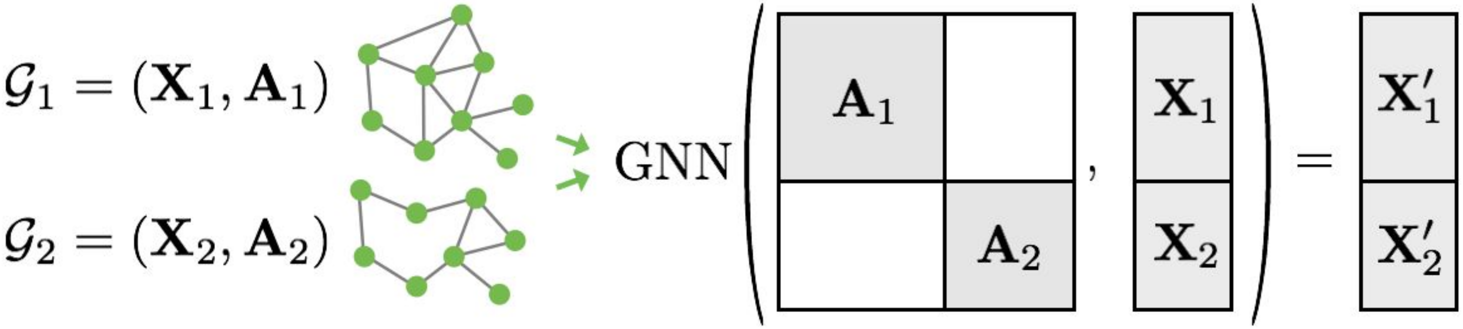
- How about a batch of ***B*** graphs:
  - What should be included as features?
    - Node features $x \in \mathbb{R}^?$
    - Edge features $\text{edge\_attr} \in \mathbb{R}^?$
    - Adjacency matrix…?
      - Edge list! $\text{edge\_index} \in \mathbb{Z}^{+?}$



How to store graph data for ML?

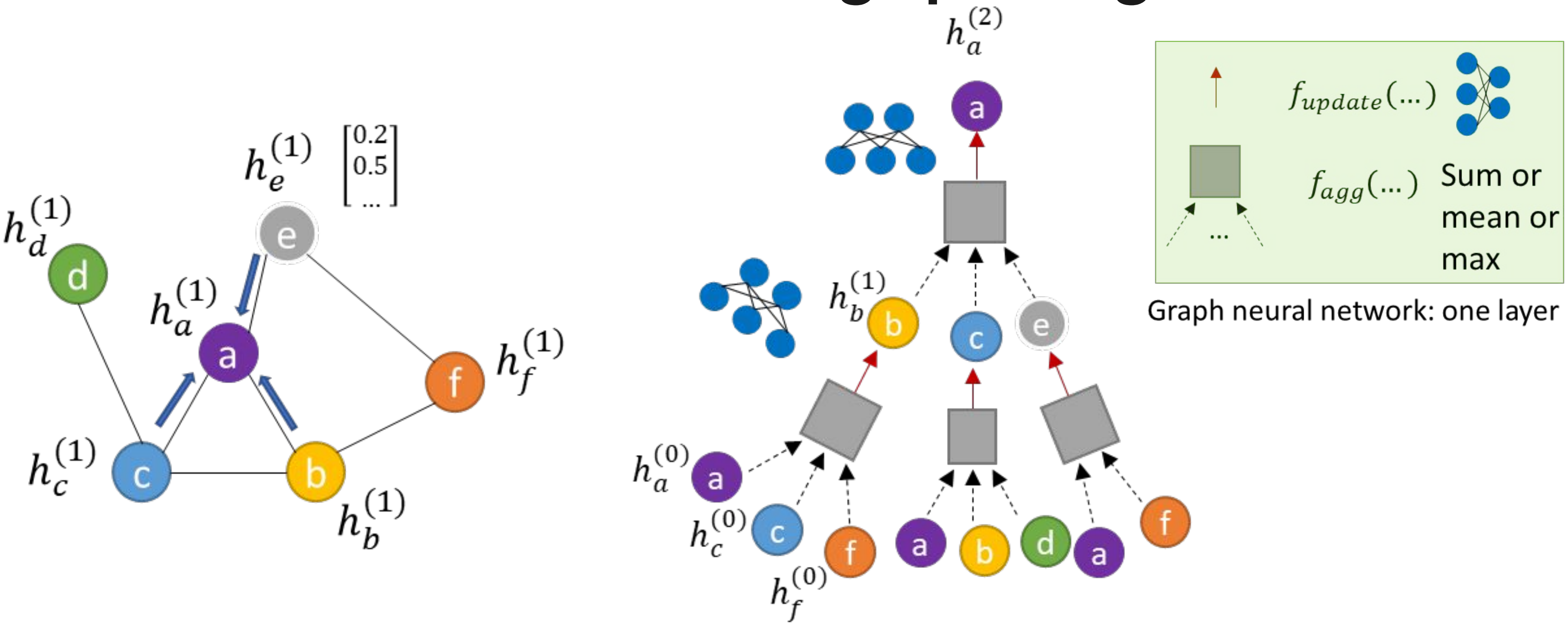
How to store graph data?

- How about a batch of ***B*** graphs:
    - What should be included as features?
        - Node features $x \in \mathbb{R}^?$
        - Edge features $\text{edge\_attr} \in \mathbb{R}^?$
        - Adjacency matrix…?
            - Edge list! $\text{edge\_index} \in \mathbb{Z}^{+?}$


$\mathcal{G}_1 = (\mathbf{X}_1, \mathbf{A}_1)$
$\mathcal{G}_2 = (\mathbf{X}_2, \mathbf{A}_2)$
$\text{GNN}\left(\begin{bmatrix}\mathbf{A}_1 & \\ & \mathbf{A}_2\end{bmatrix}, \begin{bmatrix}\mathbf{X}_1 \\ \mathbf{X}_2\end{bmatrix}\right) = \begin{bmatrix}\mathbf{X}'_1 \\ \mathbf{X}'_2\end{bmatrix}$

How to implement GNNs?

Most GNNs use a message passing scheme



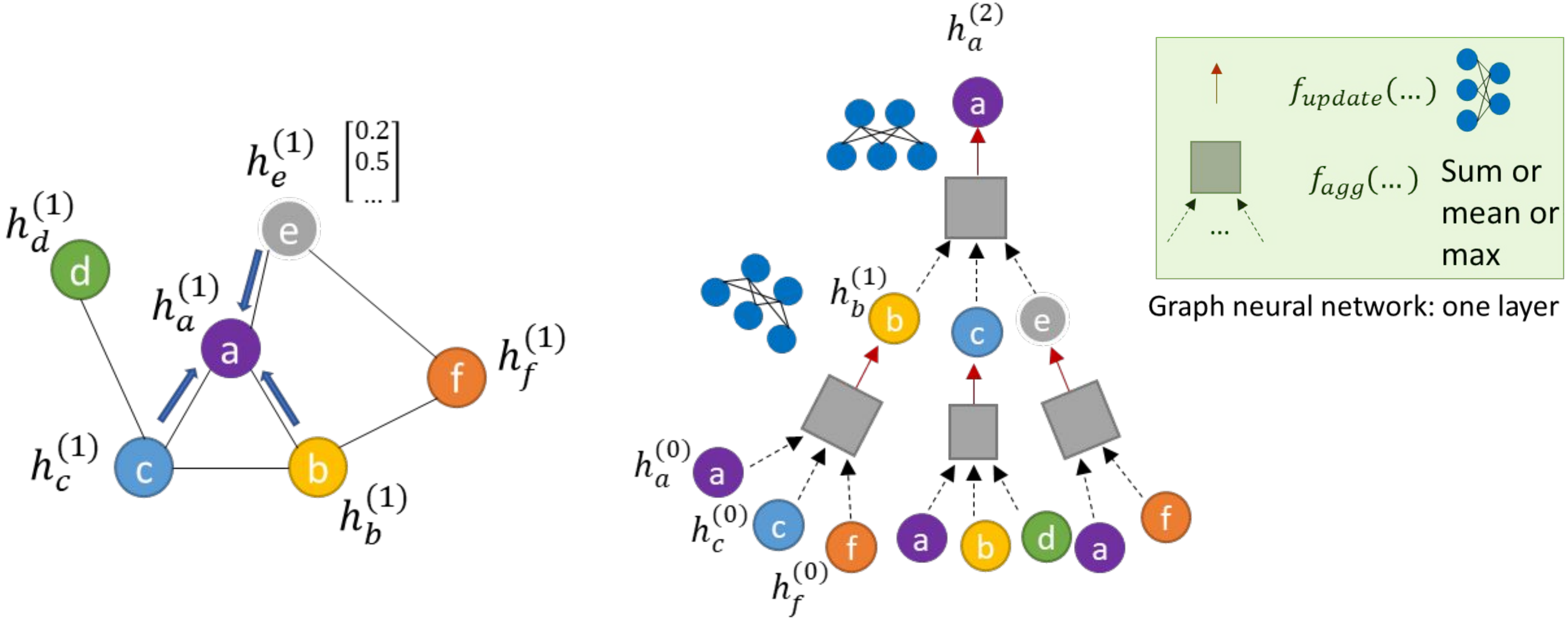
Sum or mean or max

Graph neural network: one layer

$$h_v^{(t+1)} = f_{update}\left(h_v^{(t)}, f_{agg}\left(\left\{h_u^{(t)} \mid u \in N_v\right\}\right)\right)$$

How to implement GNNs?

How about node classification tasks?

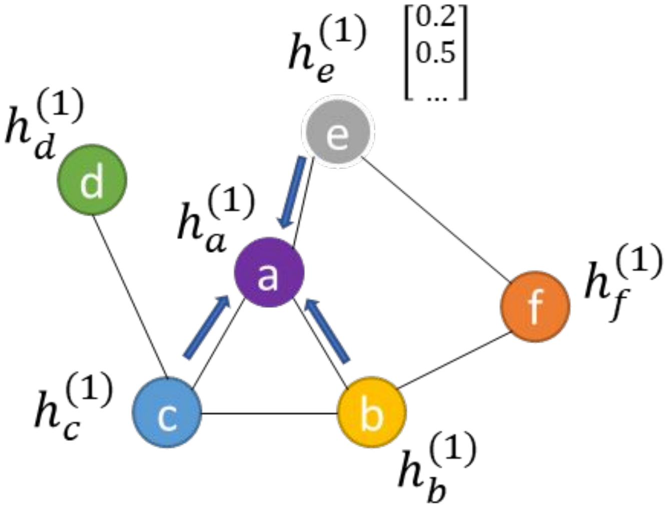


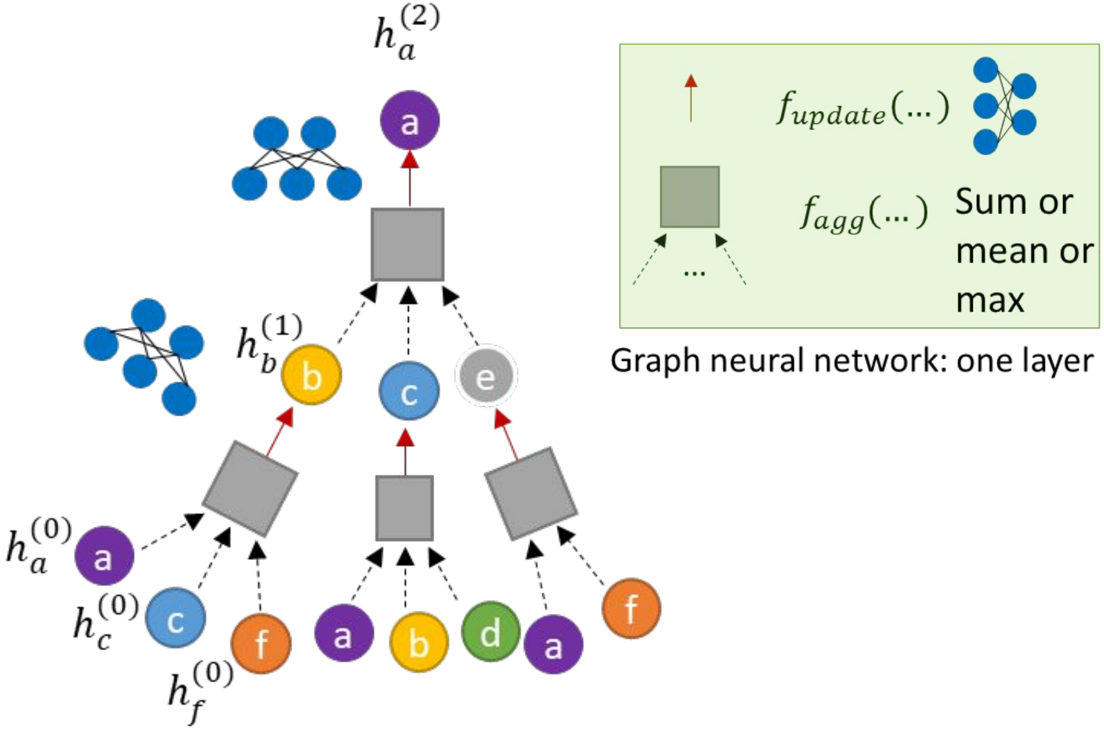
Graph neural network: one layer

$$y_v = Softmax(MLP(h_v^{(L)}))$$

How to implement GNNs?

How about graph classification tasks?





Sum or mean or max

Graph neural network: one layer

$$h_G = \mathrm{POOL}\left(\left\{h_v^{(L)} \mid v \in V\right\}\right)$$

How to implement GNNs?

Create your own GNNs!

$$h_v^{(t+1)} = f_{update}\left(h_v^{(t)}, f_{agg}\left(\left\{h_u^{(t)} \mid u \in N_v\right\}\right)\right)$$

Q & A