CS5284 : Graph Machine Learning

Deep Graph Library (DGL)

Semester 1 2024/25

Xavier Bresson

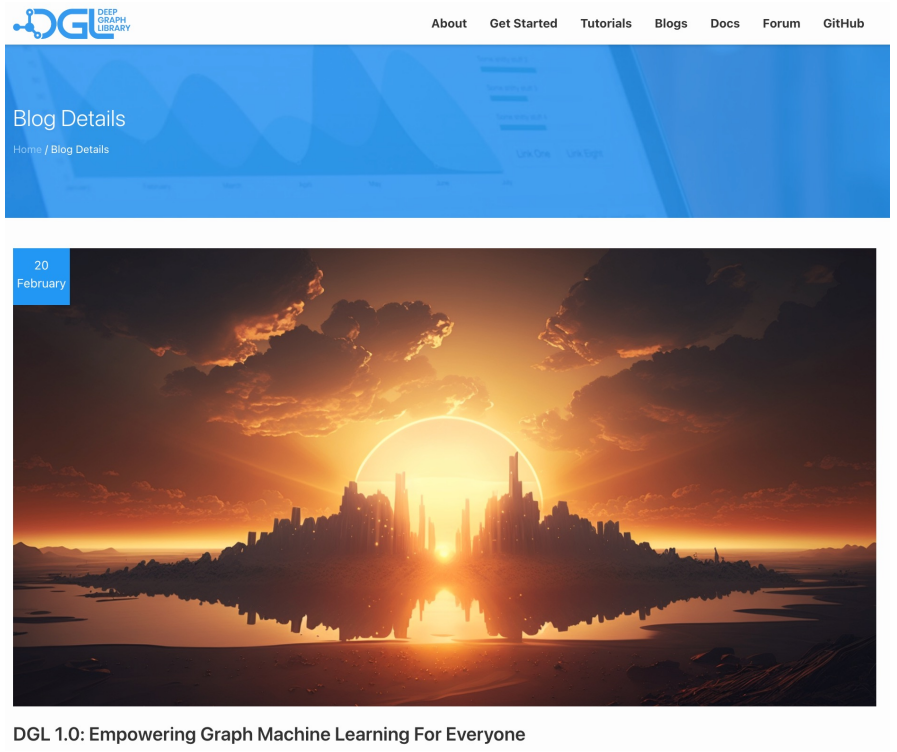
https://x.com/xbresson

Department of Computer Science

National University of Singapore (NUS)

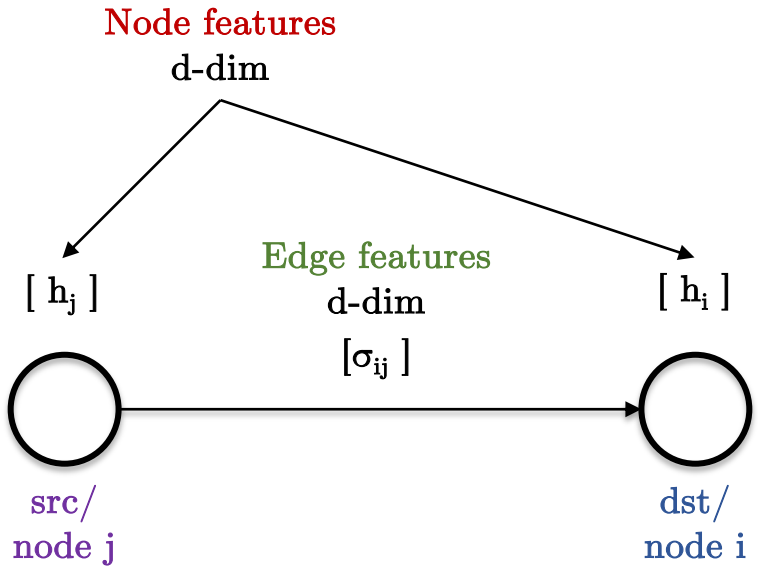
DGL library

- Developed by Prof. Zhang Zheng, AWS AI Lab
- Website : https://www.dgl.ai, https://docs.dgl.ai
- DGL 1.0 : https://www.dgl.ai/release/2023/02/20/release.html



Understanding DGL

- Basic structure of an edge in DGL :



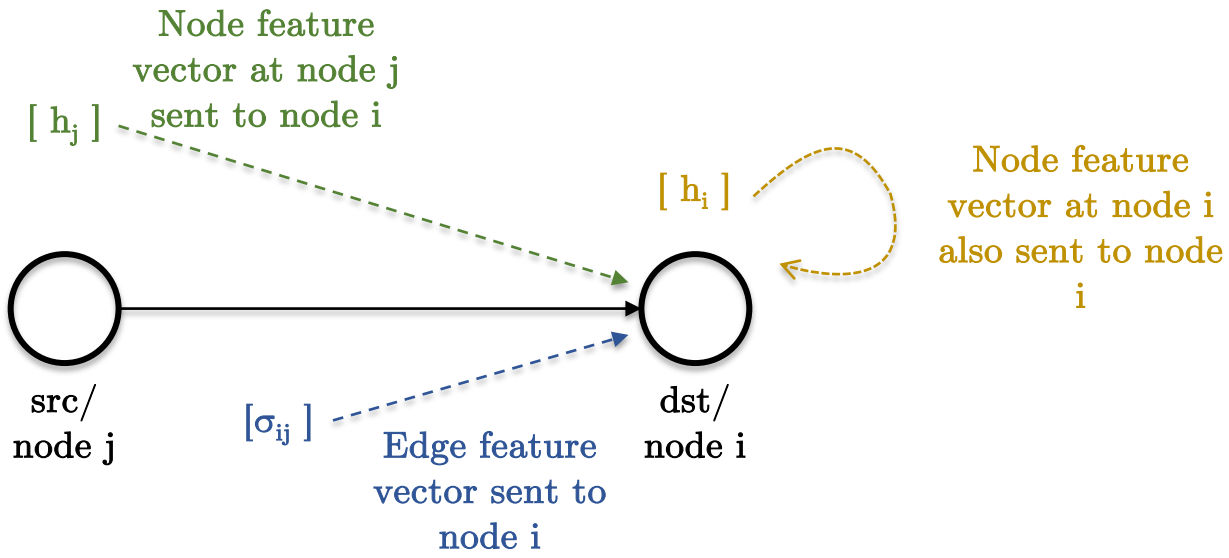
- Goal is to compute efficiently message-passing functions of the form :

$$f_i = h_i + \Sigma_{j \to i}\, \sigma_{ij} \circ h_j$$

could be mean or max

Understanding DGL

- Step 1 : Message passing function defined on edges
  - Node feature and edge feature are passed along all edges connecting a node.



h_j : edges.src['h'] . size() = E x d

h_i : edges.dst['h'] . size() = E x d

σ_{ij} : edges.data['σ'] . size() = E x d

#edges

All message passing operations can be done in parallel !

Understanding DGL

- Step 2 : Reduce function defined on nodes
  - Reduce functions of the form : $f_i = h_i + \Sigma_{j \to i}\ \sigma_{ij} \circ h_j$
  - Reduce function collects all messages passed in Step 1.
  - Code : f = h_i + torch.sum (h_j x σ_{ij} , dim=1) . size() = V x d

#nodes

Sum over neighbors

- GPU acceleration :
  - DGL batches the nodes with the same number of neighbors.

need to be permutation invariant

#nodes in $batch_1$

$$h_j = \text{nodes.mailbox}[\text{`}h_j\text{'}] = \begin{bmatrix} batch_1 .size() = 11 \text{ x } 12 \text{ x } d \\ \vdots \\ batch_{34} .size() = 14 \text{ x } 9 \text{ x } d \end{bmatrix}$$

#neighbors

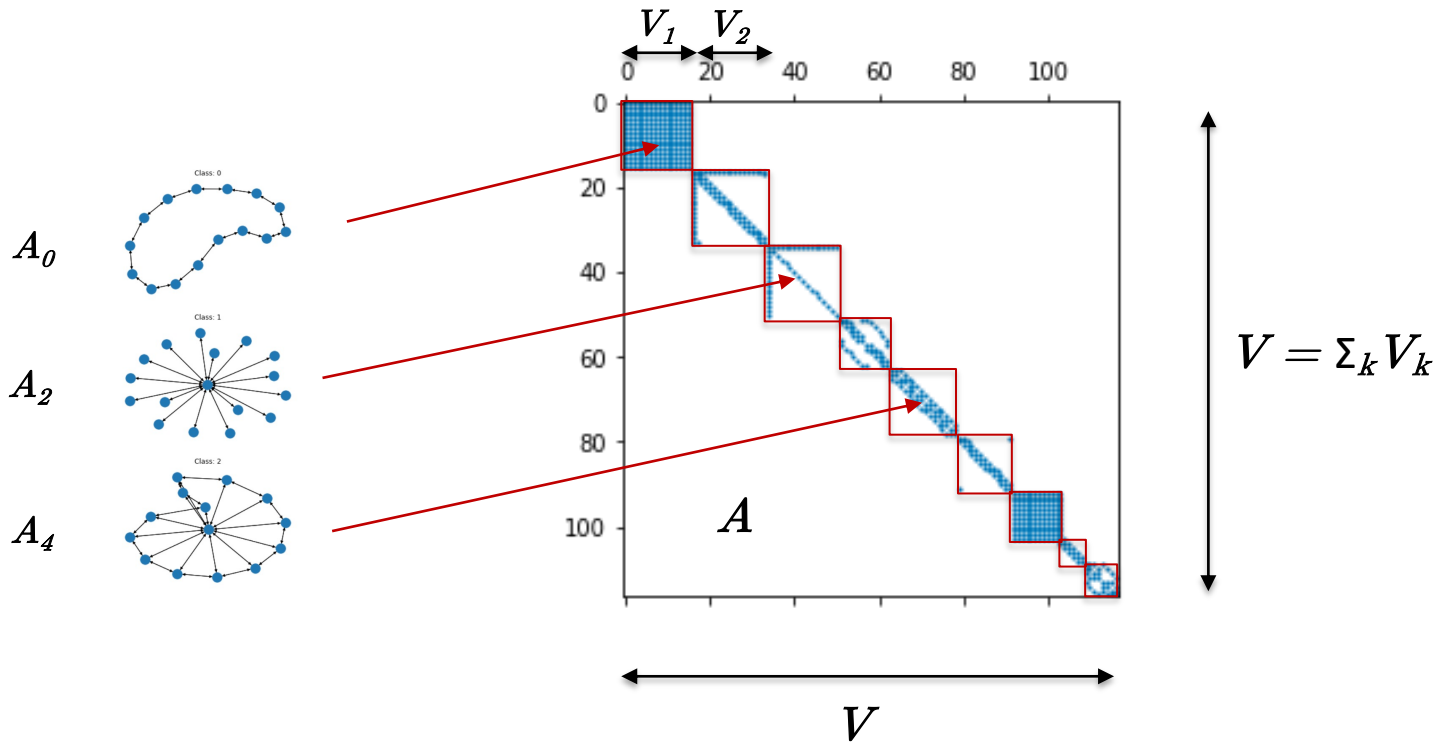
σ_{ij} = nodes.mailbox[‘σ_{ij}’] = same structure than h_j

h_i = nodes.data[‘h’] . size() = V x d

Understanding DGL

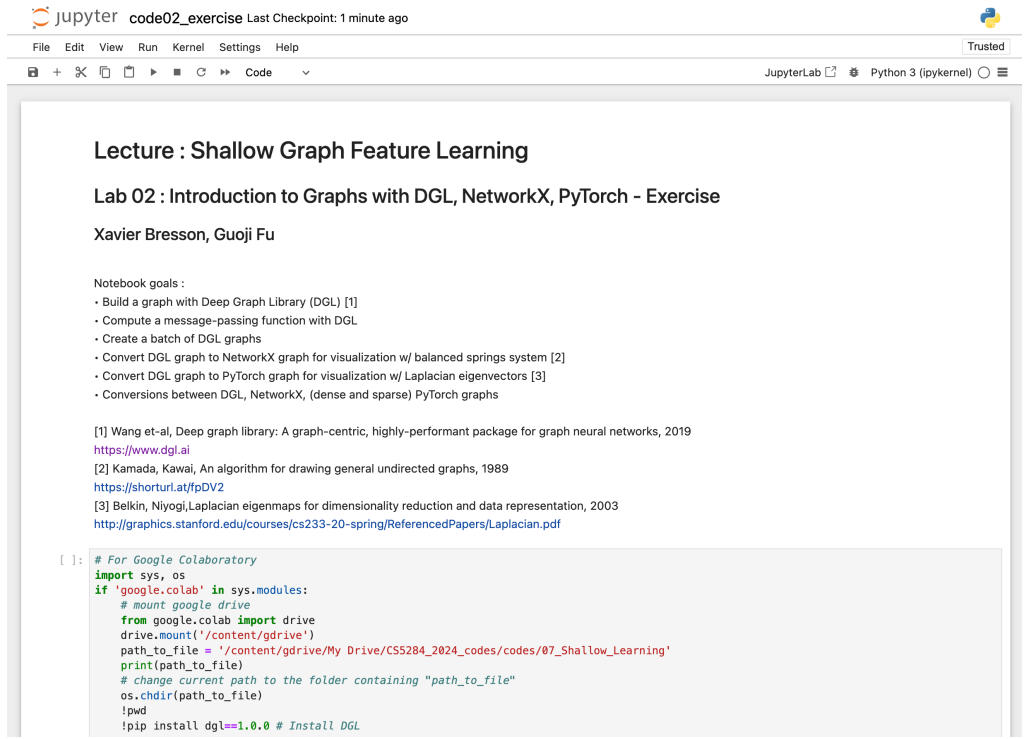
- How to process K graphs of different sizes ?

  Form a (big) sparse block diagonal matrix A with K adjacency matrices A_k .



Lab 2 : Warm-up

- Introduction to DGL, along with NetworkX and sparse/dense PyTorch.



jupyter code02_exercise Last Checkpoint: 1 minute ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

Lecture : Shallow Graph Feature Learning

Lab 02 : Introduction to Graphs with DGL, NetworkX, PyTorch - Exercise

Xavier Bresson, Guoji Fu

Notebook goals :
- Build a graph with Deep Graph Library (DGL) [1]
- Compute a message-passing function with DGL
- Create a batch of DGL graphs
- Convert DGL graph to NetworkX graph for visualization w/ balanced springs system [2]
- Convert DGL graph to PyTorch graph for visualization w/ Laplacian eigenvectors [3]
- Conversions between DGL, NetworkX, (dense and sparse) PyTorch graphs

[1] Wang et-al, Deep graph library: A graph-centric, highly-performant package for graph neural networks, 2019
https://www.dgl.ai
[2] Kamada, Kawai, An algorithm for drawing general undirected graphs, 1989
https://shorturl.at/fpDV2
[3] Belkin, Niyogi,Laplacian eigenmaps for dimensionality reduction and data representation, 2003
http://graphics.stanford.edu/courses/cs233-20-spring/ReferencedPapers/Laplacian.pdf

[ ]:
```
# For Google Colaboratory
import sys, os
if 'google.colab' in sys.modules:
    # mount google drive
    from google.colab import drive
    drive.mount('/content/gdrive')
    path_to_file = '/content/gdrive/My Drive/CS5284_2024_codes/codes/07_Shallow_Learning'
    print(path_to_file)
    # change current path to the folder containing "path_to_file"
    os.chdir(path_to_file)
    !pwd
    !pip install dgl==1.0.0 # Install DGL
```

Exercise 2 : Compute a message-passing function with DGL

Question 2.1 : Implement Step 1 of a message-passing function with DGL

Define the **Message** function to pass node and edge features along **edges**, i.e. from src/j to dst/i.

Hints:

- To access the features of destination nodes `dst`, use `edges.dst['feat_name']`, where `'feat_name'` is the name of the feature for the destination nodes.
- To access the features of source nodes `src`, use `edges.src['feat_name']`, where `'feat_name'` is the name of the feature for the source nodes.
- To access the features of edges, use `edges.data['feat_name']`, where `'feat_name'` is the name of the feature for the edges themselves.

```
# Step 1 of message-passing with DGL
#   Node feature and edge features are passed along edges (src/j => dst/i)
def message_func(edges):
    ################################################
    # YOUR CODE START
    ################################################
    # hi with i/dst, size=(E,d=1), E=num_edges
    hi = edges.dst['feat']

    # hj with j/src, size=(E,d=1)
    hj = edges.src['feat']

    # eji from src/j to dst/i, size=(E,d=100)
    eji = edges.data['feat']

    # update edge feature value
    edges.data['feat'] = 2 * edges.data['feat']

    ################################################
    # YOUR CODE END
    ################################################
    print('hi',hi.size())
    print('hj',hj.size())
    print('eji',eji.size())
```

Questions?