5.5 Use of Edge Weights

(中文版)

In a weighted graph, each edge is associated with a semantically meaningful scalar weight. For example, the edge weights can be connectivity strengths or confidence scores. Naturally, one may want to utilize edge weights in model development.

Message Passing with Edge Weights

Most graph neural networks (GNNs) integrate the graph topology information in forward computation by and only by the message passing mechanism. A message passing operation can be viewed as a function that takes an adjacency matrix and additional input features as input arguments. For an unweighted graph, the entries in the adjacency matrix can be zero or one, where a one-valued entry indicates an edge. If this graph is weighted, the non-zero entries can take arbitrary scalar values. This is equivalent to multiplying each message by its corresponding edge weight as in GAT.

With DGL, one can achieve this by:

- Saving the edge weights as an edge feature
- Multplying the original message by the edge feature in the message function

Consider the message passing example with DGL below.

```
import dgl.function as fn

# Suppose graph.ndata['ft'] stores the input node features
graph.update_all(fn.copy_u('ft', 'm'), fn.sum('m', 'ft'))
```

One can modify it for edge weight support as follows.

```
import dgl.function as fn

# Save edge weights as an edge feature, which is a tensor of shape (E, *)
# E is the number of edges
graph.edata['w'] = eweight

# Suppose graph.ndata['ft'] stores the input node features
graph.update_all(fn.u_mul_e('ft', 'w', 'm'), fn.sum('m', 'ft'))
```

Using NN Modules with Edge Weights

One can modify an NN module for edge weight support by modifying all message passing operations in it. The following code snippet is an example for NN module supporting edge weights.

DGL's built-in NN modules support edge weights if they take an optional `edge_weight` argument in the forward function.

One may need to normalize raw edge weights. In this regard, DGL provides `EdgeWeightNorm()`.