1.2 图、节点和边

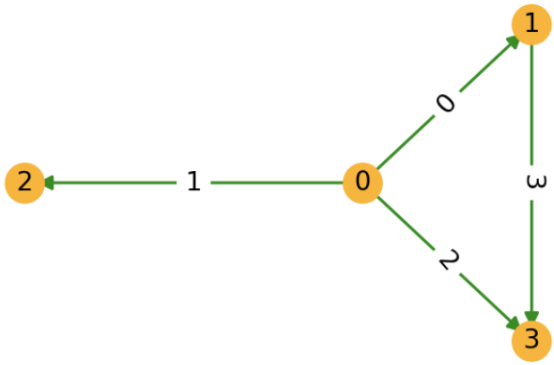
(English Version)

DGL使用一个唯一的整数来表示一个节点，称为点ID；并用对应的两个端点ID表示一条边。同时，DGL也会根据边被添加的顺序， 给每条边分配一个唯一的整数编号，称为边ID。节点和边的ID都是从0开始构建的。在DGL的图里，所有的边都是有方向的， 即边 (u, v) 表示它是从节点 u 指向节点 v 的。

对于多个节点，DGL使用一个一维的整型张量（如，PyTorch的Tensor类，TensorFlow的Tensor类或MXNet的ndarray类）来保存图的点ID， DGL称之为"节点张量"。为了指代多条边，DGL使用一个包含2个节点张量的元组 (U, V) ，其中，用 $(U[i], V[i])$ 指代一条 $U[i]$ 到 $V[i]$ 的边。

创建一个 `DGLGraph` 对象的一种方法是使用 `dgl.graph()` 函数。它接受一个边的集合作为输入。DGL也支持从其他的数据源来创建图对象。 读者可参考 1.4 从外部源创建图。

下面的代码段使用了 `dgl.graph()` 函数来构建一个 `DGLGraph` 对象，对应着下图所示的包含4个节点的图。 其中一些代码演示了查询图结构的部分API的使用方法。



```
>>> import dgl
>>> import torch as th

>>> # 边 0->1, 0->2, 0->3, 1->3
>>> u, v = th.tensor([0, 0, 0, 1]), th.tensor([1, 2, 3, 3])
>>> g = dgl.graph((u, v))
>>> print(g) # 图中节点的数量是DGL通过给定的图的边列表中最大的点ID推断所得出的
Graph(num_nodes=4, num_edges=4,
      ndata_schemes={}
      edata_schemes={})

>>> # 获取节点的ID
>>> print(g.nodes())
tensor([0, 1, 2, 3])
>>> # 获取边的对应端点
>>> print(g.edges())
(tensor([0, 0, 0, 1]), tensor([1, 2, 3, 3]))
>>> # 获取边的对应端点和边ID
>>> print(g.edges(form='all'))
(tensor([0, 0, 0, 1]), tensor([1, 2, 3, 3]), tensor([0, 1, 2, 3]))

>>> # 如果具有最大ID的节点没有边，在创建图的时候，用户需要明确地指明节点的数量。
>>> g = dgl.graph((u, v), num_nodes=8)
```

对于无向的图，用户需要为每条边都创建两个方向的边。可以使用 `dgl.to_bidirected()` 函数来实现这个目的。 如下面的代码段所示，这个函数可以把原图转换成一个包含反向边的图。

```
>>> bg = dgl.to_bidirected(g)
>>> bg.edges()
(tensor([0, 0, 0, 1, 1, 2, 3, 3]), tensor([1, 2, 3, 0, 3, 0, 0, 1]))
```

Note

由于Tensor类内部使用C来存储，且显性定义了数据类型以及存储的设备信息，DGL推荐使用Tensor作为DGL API的输入。 不过大部分的DGL API也支持Python的可迭代类型(比如列表)或numpy.ndarray类型作为API的输入，方便用户快速进行开发验证。

DGL支持使用 32 位或 64 位的整数作为节点ID和边ID。节点和边ID的数据类型必须一致。如果使用 64 位整数， DGL可以处理最多 $2^{63} - 1$ 个节点或边。不过，如果图里的节点或者边的数量小于 $2^{31} - 1$ ，用户最好使用 32 位整数。 这样不仅能提升速度，还能减少内存的使用。DGL提供了进行数据类型转换的方法，如下例所示。

```
>>> edges = th.tensor([2, 5, 3]), th.tensor([3, 5, 0])  # 边：2->3, 5->5, 3->0
>>> g64 = dgl.graph(edges)  # DGL默认使用int64
>>> print(g64.idtype)
torch.int64
>>> g32 = dgl.graph(edges, idtype=th.int32)  # 使用int32构建图
>>> g32.idtype
torch.int32
>>> g64_2 = g32.long()  # 转换成int64
>>> g64_2.idtype
torch.int64
>>> g32_2 = g64.int()  # 转换成int32
>>> g32_2.idtype
torch.int32
```

相关API：`dgl.graph()`、`dgl.DGLGraph.nodes()`、`dgl.DGLGraph.edges()`、`dgl.to_bidirected()`、`dgl.DGLGraph.int()`、`dgl.DGLGraph.long()`和`dgl.DGLGraph.idtype`。