3.3 异构图上的GraphConv模块

(English Version)

DGL提供了 `HeteroGraphConv`，用于定义异构图上GNN模块。 实现逻辑与消息传递级别的API `multi_update_all()` 相同，它包括：

- 每个关系上的DGL NN模块。
- 聚合来自不同关系上的结果。

其数学定义为：

$$h_{dst}^{(l+1)} = \underset{r\in\mathcal{R},r_{dst}=dst}{AGG}(f_r(g_r, h_{r_{src}}^l, h_{r_{dst}}^l))$$

其中 f_r 是对应每个关系 r 的NN模块，AGG 是聚合函数。

HeteroGraphConv的实现逻辑

```
import torch.nn as nn

class HeteroGraphConv(nn.Module):
    def __init__(self, mods, aggregate='sum'):
        super(HeteroGraphConv, self).__init__()
        self.mods = nn.ModuleDict(mods)
        if isinstance(aggregate, str):
            # 获取聚合函数的内部函数
            self.agg_fn = get_aggregate_fn(aggregate)
        else:
            self.agg_fn = aggregate
```

异构图的卷积操作接受一个字典类型参数 `mods`。这个字典的键为关系名，值为作用在该关系上NN模块对象。参数 `aggregate` 则指定了如何聚合来自不同关系的结果。

```
def forward(self, g, inputs, mod_args=None, mod_kwargs=None):
    if mod_args is None:
        mod_args = {}
    if mod_kwargs is None:
        mod_kwargs = {}
    outputs = {nty : [] for nty in g.dsttypes}
```

除了输入图和输入张量，`forward()` 函数还使用2个额外的字典参数 `mod_args` 和 `mod_kwargs` 。这2个字典与 `self.mods` 具有相同的键，值则为对应NN模块的自定义参数。

`forward()` 函数的输出结果也是一个字典类型的对象。其键为 `nty` ，其值为每个目标节点类型 `nty` 的输出张量的列表，表示来自不同关系的计算结果。`HeteroGraphConv` 会对这个列表进一步聚合，并将结果返回给用户。

```
if g.is_block:
    src_inputs = inputs
    dst_inputs = {k: v[:g.number_of_dst_nodes(k)] for k, v in inputs.items()}
else:
    src_inputs = dst_inputs = inputs

for stype, etype, dtype in g.canonical_etypes:
    rel_graph = g[stype, etype, dtype]
    if rel_graph.num_edges() == 0:
        continue
    if stype not in src_inputs or dtype not in dst_inputs:
        continue
    dstdata = self.mods[etype](
        rel_graph,
        (src_inputs[stype], dst_inputs[dtype]),
        *mod_args.get(etype, ()),
        **mod_kwargs.get(etype, {}))
    outputs[dtype].append(dstdata)
```

输入 `g` 可以是异构图或来自异构图的子图区块。和普通的NN模块一样，`forward()` 函数需要分别处理不同的输入图类型。

上述代码中的for循环为处理异构图计算的主要逻辑。首先我们遍历图中所有的关系(通过调用 `canonical_etypes`)。通过关系名，我们可以使用g[`stype, etype, dtype`]的语法将只包含该关系的子图(`rel_graph`)抽取出来。对于二分图，输入特征将被组织为元组 `(src_inputs[stype], dst_inputs[dtype])` 。接着调用用户预先注册在该关系上的NN模块，并将结果保存在outputs字典中。

```
rsts = {}
for nty, alist in outputs.items():
    if len(alist) != 0:
        rsts[nty] = self.agg_fn(alist, nty)
```

最后，`HeteroGraphConv` 会调用用户注册的 `self.agg_fn` 函数聚合来自多个关系的结果。 读者可以在API文档中找到 :class:~dgl.nn.pytorch.HeteroGraphConv 的示例。