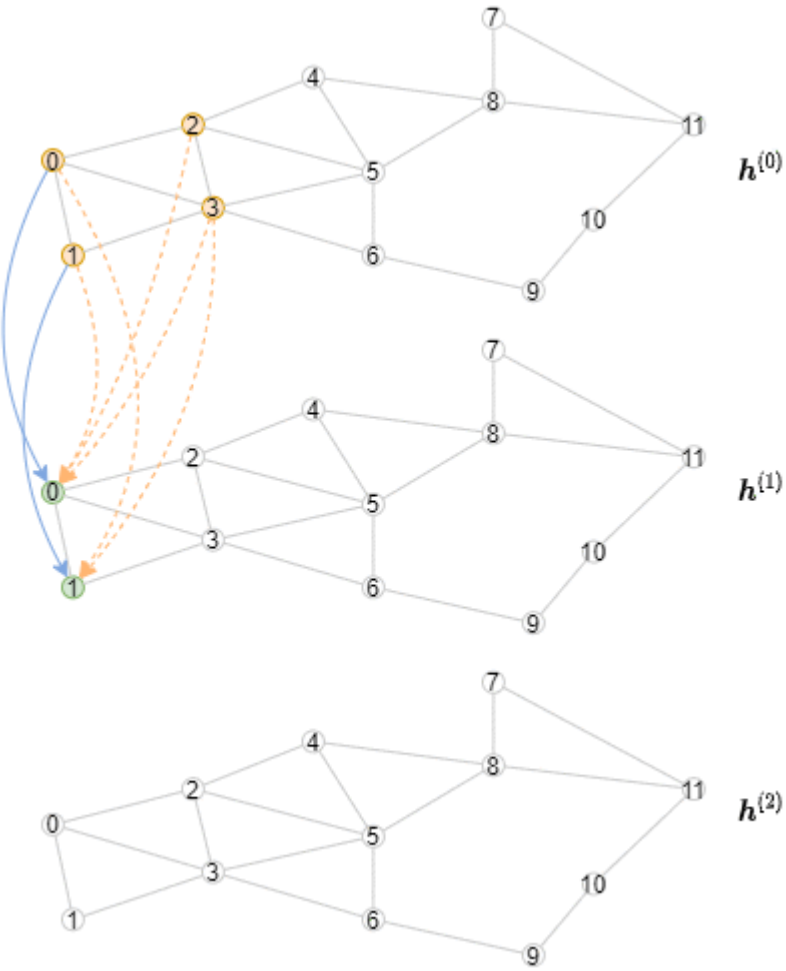
6.6 超大图上的精准离线推断

(English Version)

子图采样和邻居采样都是为了减少用GPU训练GNN模型的内存和时间消耗。在进行推断时，通常更好的方法是将所有邻居进行真正的聚合，以避免采样所带来的随机性。 然而，在GPU上进行全图前向传播通常由于显存大小的限制而不可行，而在CPU上进行则计算速度很慢。本节介绍了在GPU显存有限的情况下通过小批次处理和邻居采样实现全图前向传播的方法。

推断算法不同于训练算法，因为需要从第一层开始对节点表示逐层计算。具体来说，对于一个指定的层， 需要以小批次的方式计算这个GNN层所有节点的输出表示。其结果是，推断算法将包含一个外循环以迭代执行各层， 和一个内循环以迭代处理各个节点小批次。相比之下，训练算法有一个外循环以迭代处理各个节点小批次， 和一个内循环以迭代执行各层（包含邻居采样和消息传递）。

下面的动画展示了计算的过程（注意，每层只展示前3个小批次）：



实现离线推断

这里以6.1节中 调整模型以进行小批次训练 提到的两层GCN为例。实现离线推断的方法依然需要使用 `MultiLayerFullNeighborSampler` ， 但它每次只为一层进行采样。注意，这里的离线推断被实现为GNN模块的一个方法， 这是因为它对一层的计算依赖于消息的聚合和结合。

```
class StochasticTwoLayerGCN(nn.Module):
    def __init__(self, in_features, hidden_features, out_features):
        super().__init__()
        self.hidden_features = hidden_features
        self.out_features = out_features
        self.conv1 = dgl.nn.GraphConv(in_features, hidden_features)
        self.conv2 = dgl.nn.GraphConv(hidden_features, out_features)
        self.n_layers = 2

    def forward(self, blocks, x):
        x_dst = x[:blocks[0].number_of_dst_nodes()]
        x = F.relu(self.conv1(blocks[0], (x, x_dst)))
        x_dst = x[:blocks[1].number_of_dst_nodes()]
        x = F.relu(self.conv2(blocks[1], (x, x_dst)))
        return x

    def inference(self, g, x, batch_size, device):
        """        用该模块进行离线推断        """
        # 逐层计算表示
        for l, layer in enumerate([self.conv1, self.conv2]):
            y = torch.zeros(g.num_nodes(),
                            self.hidden_features
                            if l != self.n_layers - 1
                            else self.out_features)
            sampler = dgl.dataloading.MultiLayerFullNeighborSampler(1)
            dataloader = dgl.dataloading.NodeDataLoader(
                g, torch.arange(g.num_nodes()), sampler,
                batch_size=batch_size,
                shuffle=True,
                drop_last=False)

            # 在一层中，依批次对节点进行迭代
            for input_nodes, output_nodes, blocks in dataloader:
                block = blocks[0]

                # 将必要输入节点的特征复制到GPU上
                h = x[input_nodes].to(device)

                # 计算输出，注意计算方法是一样的，但只对一层进行计算
                h_dst = h[:block.number_of_dst_nodes()]
                h = F.relu(layer(block, (h, h_dst)))

                # 将输出复制回CPU
                y[output_nodes] = h.cpu()

            x = y

        return y
```

注意，如果以模型选择为目的在验证集上计算评价指标，则通常不需要进行计算精确的离线推断。 原因是这需要为每一层上的每个节点计算表示，会非常消耗资源，尤其是在包含大量未标记数据的半监督系统中。 邻居采样在这个时候可以更好地发挥作用。

对于离线推断的示例，用户可以参照 GraphSAGE 和 RGCN。