1. 请总结MessagePassing基类的运行流程。

消息传递图神经网络是一种通过聚合邻接节点信息来更新中心节点信息的过程，用公式可以表示为：

$$\mathbf{x}_i^{(k)} = \gamma^{(k)}\left(\mathbf{x}_i^{(k-1)}, \square_{j\in\mathcal{N}(i)}\,\phi^{(k)}\left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i}\right)\right)$$

其中输入为：

- $\mathbf{x}_i^{(k-1)} \in \mathbb{R}^F$：第(k-1)层中节点i的表征
- $\mathbf{e}_{j,i} \in \mathbb{R}^D$：从节点j到节点i的边的属性

输出为：

- $\mathbf{x}_i^{(k)} \in \mathbb{R}^F$：第(k)层中节点i的表征

根据这一公式，MessagePassing的运行流程为：

1. 定义message函数，对节点输入进行自定义变换，相当于上述公式中的$\phi^{(k)}$，例如线性投影。
2. 定义aggregation函数，对变换后的邻居节点信息进行聚合，相当于上述公式中的$\square_{j\in\mathcal{N}(i)}$，例如sum、max、mean。
3. 定义update函数，将上一层节点信息$\mathbf{x}_i^{(k-1)}$与聚合邻居节点信息$\square_{j\in\mathcal{N}(i)}\,\phi^{(k)}\left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i}\right)$结合起来，更新得到下一层节点信息$\mathbf{x}_i^{(k)}$。

2. 请复现一个一层的图神经网络的构造，总结通过继承MessagePassing基类来构造自己的图神经网络类的规范。

我们尝试复现GraphSAGE，其伪代码如下所示：

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

仿照上一题的思路，将流程进行拆解：

1. message函数：这里没有对节点进行变换，直接输出。
2. Aggregation函数：这边使用mean，直接调用自带的函数，输入参数aggr='mean'
3. update函数：将聚合后的节点表征与原始表征拼接cat，再输入线性变换lin和激活函数act，最后进行归一化。

```
import torch
from torch.nn import Linear, ReLU
from torch.nn import functional as F
from torch_geometric.nn import MessagePassing


```

```
class SAGEConv(MessagePassing):
    def __init__(self, in_channels, out_channels):
        super(SAGEConv, self).__init__(aggr='mean')
        self.lin = torch.nn.Linear(in_channels + out_channels, out_channels,
bias=False)
        self.act = torch.nn.ReLU()

    def forward(self, x, edge_index):
        return self.propagate(edge_index, size=(x.size(0), x.size(0)), x=x)

    def message(self, x_j):
        return x_j

    def update(self, aggr_out, x):
        aggr_out = torch.cat([aggr_out, x], dim=1)

        aggr_out = self.lin(aggr_out)
        aggr_out = self.act(aggr_out)

        aggr_out = F.normalize(aggr_out, p=2, dim=1)
        return aggr_out
```