**Tensor属性**

data：张量的值

grad：张量的梯度值

grad\_fn：得到张量的运算

requires\_grad：是否需要为张量计算梯度

is\_leaf：是否为叶子节点

**叶子节点**

叶子节点有两种产生方式：

①由用户的创建的Tensor，无论是否设置requires\_grad=True，都为叶子节点

②仅由requires\_grad=False的叶子节点(requires\_grad=False的节点只能为叶子节点)运算得到的节点为叶子节点

**性质：**叶子节点的的requires\_grad=True/False，即不确定是否需要计算梯度；只有requires=True的叶子节点才有grad；rad\_fn=None

**非叶子节点**

非叶子节点只有一种产生方式：

①由requires\_grad=True的节点运算得到的Tensor为非叶子节点

**性质：**非叶子节点的requires\_grad=True，实际上，PyTorch中无法将非叶子节点requires\_grad设置为False，因为这将影响叶节点梯度的计算；无grad值；grad\_fn存在

看一个具体实例的计算图

Y4

Y3

Loss

s

+

s

+

XX

X

P2

P1

Y1

Y2

p1=torch.tensor([1.0,2.0,3.0],requires\_grad=True)

p2=torch.tensor([1.1,2.2,3.3],requires\_grad=True)

x=torch.tensor([4.0,5.0,6.0],requires\_grad=False)

xx=x\*\*2

y1=p1+xx

y2=torch.sigmoid(y1)

y3=y2+p2

y4=torch.sigmoid(y3)

loss=y4.mean()

print(x.is\_leaf,xx.is\_leaf,p1.is\_leaf,p2.is\_leaf)

print(y1.is\_leaf,y2.is\_leaf,y3.is\_leaf,y4.is\_leaf)

>>>

True True True True

False False False False False