**nn.Module.children()**

返回包括模块所有子模块的迭代器

class ModelD(nn.Module):

def \_\_init\_\_(self):

super(ModelD,self).\_\_init\_\_()

self.model=nn.Sequential() *#序列化模块构造的神经网络*

self.model.add\_module('conv1',nn.Conv2d(num\_channels,num\_features,5,2,0,bias=False))

self.model.add\_module('relu1',nn.ReLU())

self.model.add\_module('conv2',nn.Conv2d(num\_features,num\_features\*2,5,2,0,bias=False))

self.model.add\_module('bnorm2',nn.BatchNorm2d(num\_features\*2))

self.model.add\_module('linear1',nn.Linear(num\_features\*2\*4\*4,num\_features))

self.model.add\_module('linear2',nn.Linear(num\_features,1))

self.model.add\_module('sigmoid',nn.Sigmoid())

netD = ModelD()

for module in netD.model.children():

print(module)

>>>

Conv2d(1, 64, kernel\_size=(5, 5), stride=(2, 2), bias=False)

ReLU()

Conv2d(64, 128, kernel\_size=(5, 5), stride=(2, 2), bias=False)

BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

Linear(in\_features=2048, out\_features=64, bias=True)

Linear(in\_features=64, out\_features=1, bias=True)

Sigmoid()

**nn.Module.named\_children()**

返回包括所有(子模块名称,子模块)二元组的迭代器

for name,module in netD.model.named\_children():

print(name+':',module)

>>>

conv1: Conv2d(1, 64, kernel\_size=(5, 5), stride=(2, 2), bias=False)

relu1: ReLU()

conv2: Conv2d(64, 128, kernel\_size=(5, 5), stride=(2, 2), bias=False)

bnorm2: BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track\_running\_stats=True)

linear1: Linear(in\_features=2048, out\_features=64, bias=True)

linear2: Linear(in\_features=64, out\_features=1, bias=True)

sigmoid: Sigmoid()