**nn.Module.apply(fn)**

将函数fn递归地应用到模块的每个子模块和自身，常用于初始化模型参数

内部执行：

for module in self.children():

module.apply(fn)

fn(self)

return self

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())

def weight\_init(m):

*#使用如下初始化方式可以，可以让方差更小，使得收敛更快*

class\_name=m.\_\_class\_\_.\_\_name\_\_

if class\_name.find('conv')!=-1:

m.weight.data.normal\_(0,0.02)

if class\_name.find('norm')!=-1:

m.weight.data.normal\_(1.0,0.02)

netD = ModelD()

netD.apply(weight\_init)*#若ModelD中有实例属性weight，则此语句也会修改weight*