torch.squeeze（a,N）

先看torch.squeeze() 这个函数主要对数据的维度进行压缩，去掉维数为1的的维度，比如是一行或者一列这种，一个一行三列（1,3）的数去掉第一个维数为一的维度之后就变成（3）行。

1.squeeze(a)就是将a中所有为1的维度删掉。不为1的维度没有影响。

2.a.squeeze(N) 就是去掉a中指定的维数为一的维度。

  还有一种形式就是b=torch.squeeze(a，N) a中去掉指定的维数N为一的维度。

先看torch.squeeze() 这个函数主要对数据的维度进行压缩，去掉维数为1的的维度，比如是一行或者一列这种，一个一行三列（1,3）的数去掉第一个维数为一的维度之后就变成（3）行。

1.squeeze(a)就是将a中所有为1的维度删掉。不为1的维度没有影响。

2.a.squeeze(N) 就是去掉a中指定的维数为一的维度。

  还有一种形式就是b=torch.squeeze(a，N) a中去掉指定的维数N为一的维度。

torch.unsqueeze（a,N)和上面的相反，这个函数主要对数据的维度进行扩充

线性间距向量：

torch.linspace(start, end, steps=100, out=None, dtype=None, layout=torch.strided, device=None, requires\_grad=False)

start：开始值

end：结束值

steps：分割的点数，默认是100

dtype：返回值（张量）的数据类型

Torch.pow()

函数是计算x的y次方，如果z在存在，则再对结果进行取模，其结果等效于pow(x,y) %z

均布分布：torch.rand(\*sizes, out=None) → Tensor

返回一个张量，包含了从区间[0, 1)的均匀分布中抽取的一组随机数。张量的形状由参数sizes定义

torch.rand(2, 3)  
0.0836 0.6151 0.6958  
0.6998 0.2560 0.0139  
[torch.FloatTensor of size 2x3]

标准正态分布：torch.randn(\*sizes, out=None) → Tensor

返回一个张量，包含了从标准正态分布（均值为0，方差为1，即高斯白噪声）中抽取的一组随机数。张量的形状由参数sizes定义。

torch.randn(2, 3)  
0.5419 0.1594 -0.0413  
-2.7937 0.9534 0.4561  
[torch.FloatTensor of size 2x3]

matplotlib.pyplot.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, verts=None, edgecolors=None, \*, data=None, \*\*kwargs)

参数的解释：

x，y：表示的是大小为(n,)的数组，也就是我们即将绘制散点图的数据点

s:是一个实数或者是一个数组大小为(n,)，这个是一个可选的参数。

c:表示的是颜色，也是一个可选项。默认是蓝色'b',表示的是标记的颜色，或者可以是一个表示颜色的字符，或者是一个长度为n的表示颜色的序列等等，感觉还没用到过现在不解释了。但是c不可以是一个单独的RGB数字，也不可以是一个RGBA的序列。可以是他们的2维数组（只有一行）。

marker:表示的是标记的样式，默认的是'o'。

cmap:Colormap实体或者是一个colormap的名字，cmap仅仅当c是一个浮点数数组的时候才使用。如果没有申明就是image.cmap

norm:Normalize实体来将数据亮度转化到0-1之间，也是只有c是一个浮点数的数组的时候才使用。如果没有申明，就是默认为colors.Normalize。

vmin,vmax:实数，当norm存在的时候忽略。用来进行亮度数据的归一化。

alpha：实数，0-1之间。

linewidths:也就是标记点的长度。

class torch.nn.Linear（in\_features，out\_features，bias = True ）

对传入数据应用线性变换：y = A x+ b

参数：

in\_features - 每个输入样本的大小

out\_features - 每个输出样本的大小

bias - 如果设置为False，则图层不会学习附加偏差。默认值：True

plt.scatter 绘制散点图 具体去上网查

网址：https://blog.csdn.net/qiu931110/article/details/68130199

Plt.show 显示散点图

plt.ion() 打开交互模式

requires\_grad=True    要求梯度

requires\_grad=False   不要求梯度

nn.ReLU(inplace=True)

inplace为True，将会改变输入的数据 ，否则不会改变原输入，只会产生新的输出。

nn.Conv2d(self, in\_channels, out\_channels, kernel\_size, stride=1, padding=0, dilation=1, groups=1, bias=True))

参数：

  in\_channel:　输入数据的通道数，例RGB图片通道数为3；

  out\_channel: 输出数据的通道数，这个根据模型调整；

  kennel\_size: 卷积核大小，可以是int，或tuple；kennel\_size=2,意味着卷积大小2， kennel\_size=（2,3），意味着卷积在第一维度大小为2，在第二维度大小为3；

  stride：步长，默认为1，与kennel\_size类似，stride=2,意味在所有维度步长为2， stride=（2,3），意味着在第一维度步长为2，意味着在第二维度步长为3；

  padding=N：为图片周围填充N圈0的像素点

class torch.nn.MaxPool2d(kernel\_size, stride=None, padding=0, dilation=1, return\_indices=False, ceil\_mode=False)

如果padding不是0，会在输入的每一边添加相应数目0  比如padding=1，则在每一边分别补0 ，其实最后的结果补出来是bias

参数：

kernel\_size(int or tuple) - max pooling的窗口大小，可以为tuple，在nlp中tuple用更多，（n,1）

stride(int or tuple, optional) - max pooling的窗口移动的步长。默认值是kernel\_size

padding(int or tuple, optional) - 输入的每一条边补充0的层数

dilation(int or tuple, optional) – 一个控制窗口中元素步幅的参数

return\_indices - 如果等于True，会返回输出最大值的序号，对于上采样操作会有帮助

ceil\_mode - 如果等于True，计算输出信号大小的时候，会使用向上取整，代替默认的向下取整的操作

Torch.manual\_seed()

为CPU设置种子用于生成随机数，以使得结果是确定的

Import torch

Torch.manual\_seed(2)

Print(torch.rand(2))

要是没有Torch.manual\_seed这个代码的话，那么打印出来的随机数是会变化的，要是用了Torch.manual\_seed这个代码的话，那么每次打印出来的代码是固定的