pytorch实现BN, LN, GN

BN

```
import torch
from torch import nn

'''
tracnk_running_stats= False: 求当前batch真实平均值和标准差
affine=False: 只做归一化，不乘以gamma，beta(通过训练才能确定
num_features: 为feature map的channel数目
eps: 设为0，让官方代码和自己的结果尽量接近
'''
bn = nn.BatchNorm2d(num_features=3, eps=0,
affine=False,track_running_stats=False)

#乘10000为了扩大数值，如果出现不一致，差别更明显
x = torch.rand(10,3,5,5)*10000
official_bn = bn(x)

#及那个channel维度单独提出来，而把其他需要平均值和标准差的维度融合到一起.
x1 = x.permute(1,0,2,3).contiguous().view(3,-1)

mu = x1.mean(dim=1).view(1,3,1,1)
#unbiased=False，求方差时不做无偏估计(除以N-1，而不是N)，和原始论文一致
std = x1.std(dim=1,unbiased=False).view(1,3,1,1)#对每个通道求均值

my_bn = (x-mu)/std

diff = (official_bn - my_bn).sum()
print('diff={}'.format(diff))
```

GN计算均值和标准差时, 把每个样本feature map的channel分成G组, 每组将有C/G个channel, 然后将这些channel中的元素求均值和标准差. 各组channel用其对应的归一化参数独立地归一化.

Group Normalization (GN)用于显存比较大的任务, 如图像分割. 对这类任务, 可能batchsize只能是个位数, 再打显存就不够用了. 而当batchsize是个位数时, BN的表现很差, 因为没办法通过几个样本的数据量, 来近似总体的均值和标准差.

```
import torch
from torch import nn

x = torch.rand(10,20,5,5)*10000

#分成4个group，其余设定与之前相同
gn = nn.GroupNorm(num_groups=4, num_channels=20,eps=0,affine=False)
official_gn = gn(x)

#把同一group的元素融合到一起
x1 = x.view(10,4,-1)
mu = x1.mean(dim=-1).reshape(10,4,-1)
std = x1.std(dim=-1).reshape(10,4,-1)
```

```
x1_norm = (x1-mu)/std
my_gn = x1_norm.reshape(10,20,5,5)

diff = (my_gn - official_gn).sum()

print("diff={}".format(diff))
```

BN的一个缺点时需要较大的batchsize才能合理估计训练数据的均值和方差, 这导致内存很可能不够用, 同时它也很难应用再训练数据长度不同的RNN模型上.

Layer Normalization (LN)的一个优势是不需要批训练, 再单条数据内部就能归一化. 把一个batch的feature类比为一摞书. LN求均值时, 相当于把每一本书的所有字加起来,再除以这本书的字符总数: CxHxW, 即求整本书的'平均字'.

```
import torch
from torch import nn

x=torch.rand(10,3,5,5)*10000
'''
normalization_shape: 相当于告诉程序这本书有多少页, 每页多少行多少列
eps=0: 排除干扰
elementwise_affine=False: 不作映射
这里的映射和BN以及下文的IN有区别, 它是elementwise的affine,
即gamma和beta不是channel维的向量, 而是维度等于normalized_shape的矩阵
'''
ln = nn.LayerNorm(normalized_shape=[3,5,5],eps=0, elementwise_affine=Fasle)

official_ln = ln(x)

x1 = x.view(10,-1)
mu = x1.mean(dim=1).view(10,1,1,1)
std = x1.std(dim=1, unbiased=False).view(10,1,1,1)

my_ln = (x-mu)/std
diff = (my_ln-official_ln).sum()
print("diff={}".)
```