一 bn的原理

* 1. 为了解决什么问题.

首先，此部分也即是讲为什么深度网络会需要batchnorm batchnormbatchnorm，我们都知道，深度学习的话尤其是在CV上都需要对数据做归一化，**因为深度神经网络主要就是为了学习训练数据的分布，并在测试集上达到很好的泛化效果**，但是，**如果我们每一个batch输入的数据都具有不同的分布，显然会给网络的训练带来困难**。另一方面，**数据经过一层层网络计算后，其数据分布也在发生着变化**，此现象称为***Internal Covariate Shift***，接下来会详细解释，会给下一层的网络学习带来困难。batchnorm直译过来就是批规范化，就是**为了解决这个分布变化问题。**

何为internal covarate shift?

其主要描述的是：训练深度网络的时候经常发生训练困难的问题，因为，每一次参数迭代更新后，上一层网络的输出数据经过这一层网络计算后，数据的分布会发生变化，为下一层网络的学习带来困难（神经网络本来就是要学习数据的分布，要是分布一直在变，学习就很难了）

和covarite shift区别

**Internal Covariate Shift和Covariate Shift具有相似性，但并不是一个东西**，前者发生在神经网络的内部，所以是Internal，后者发生在输入数据上。

**Covariate Shift主要描述的是由于训练数据和测试数据存在分布的差异性**，给网络的泛化性和训练速度带来了影响，我们经常使用的方法是做归一化或者白化。

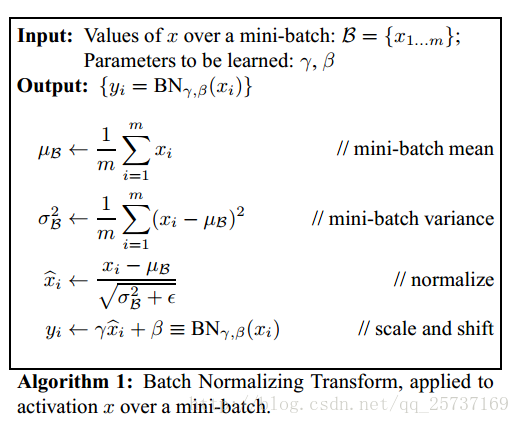
1. Covariate shift是训练集和测试集的分布不同.
2. Internal covariate shift是每一个layer的输出的分布是不同的(一直在变化的). 给收敛造成困难.

二 bn原理

为减小internal covariate shift,引入batch normalization.

(思考) 假设将每一层输出后的数据都归一化到0均值，1方差，满足正太分布，但是，此时有一个问题，**每一层的数据分布都是标准正太分布，导致其完全学习不到输入数据的特征**，因为，费劲心思学习到的特征分布被归一化了，因此，直接对每一层做归一化显然是不合理的。

但是如果稍作修改，加入可训练的参数做归一化，那就是BatchNorm实现的了，接下来结合下图的伪代码做详细的分析：



1. 输入是mini batch的(样本). X1到Xm
2. 超参数是γ,β.
3. 先求minibatch样本的均值μB.
4. 求对应的方差σB
5. 利用均值求更新后的xi.(正则化)
6. 然后对xi做缩放和平移(保留该layer的特征属性).

训练时候的BN代码段:

|  |
| --- |
| **def** Batchnorm\_simple\_for\_train**(**x**,** gamma**,** beta**,** bn\_param**):**  """  param:x : 输入数据，设shape(B,L)  param:gama : 缩放因子 γ  param:beta : 平移因子 β  param:bn\_param : batchnorm所需要的一些参数  eps : 接近0的数，防止分母出现0  momentum : 动量参数，一般为0.9， 0.99， 0.999  running\_mean ：滑动平均的方式计算新的均值，训练时计算，为测试数据做准备  running\_var : 滑动平均的方式计算新的方差，训练时计算，为测试数据做准备  """  running\_mean **=** bn\_param**[**'running\_mean'**]** #shape = [B]  running\_var **=** bn\_param**[**'running\_var'**]** #shape = [B]  results **=** 0. # 建立一个新的变量    **## x是[B F]的.**   1. **每个B下是一组F, 每个F是={F1,..Ff}.** 2. **让所有B对应的Fi相加求均值.** 3. **最后的x\_mean是[1 F]的结构. F是{Fmean1, Fmean2, …}**   x\_mean**=**x**.**mean**(**axis**=**0**)** # 计算x的均值  x\_var**=**x**.**var**(**axis**=**0**)** # 计算方差  x\_normalized**=(**x**-**x\_mean**)/**np**.**sqrt**(**x\_var**+**eps**)** # 归一化  results **=** gamma **\*** x\_normalized **+** beta # 缩放平移  **## 求mean和var的指数平均. 这个会在test的时候使用.**  running\_mean **=** momentum **\*** running\_mean **+** **(**1 **-** momentum**)** **\*** x\_mean  running\_var **=** momentum **\*** running\_var **+** **(**1 **-** momentum**)** **\*** x\_var    #记录新的值  bn\_param**[**'running\_mean'**]** **=** running\_mean  bn\_param**[**'running\_var'**]** **=** running\_var    **return** results **,** bn\_param |

推断时候的代码段:

|  |
| --- |
| **def** Batchnorm\_simple\_for\_test**(**x**,** gamma**,** beta**,** bn\_param**):**  """  param:x : 输入数据，设shape(B,L)  param:gama : 缩放因子 γ  param:beta : 平移因子 β  param:bn\_param : batchnorm所需要的一些参数  eps : 接近0的数，防止分母出现0  momentum : 动量参数，一般为0.9， 0.99， 0.999  running\_mean ：滑动平均的方式计算新的均值，训练时计算，为测试数据做准备  running\_var : 滑动平均的方式计算新的方差，训练时计算，为测试数据做准备  """  running\_mean **=** bn\_param**[**'running\_mean'**]** #shape = [B]  running\_var **=** bn\_param**[**'running\_var'**]** #shape = [B]  results **=** 0. # 建立一个新的变量  **## 使用train时候的参数**  x\_normalized**=(**x**-**running\_mean **)/**np**.**sqrt**(**running\_var **+**eps**)** # 归一化  results **=** gamma **\*** x\_normalized **+** beta # 缩放平移    **return** results **,** bn\_param |

二 tensorflow的Bn的使用

|  |
| --- |
| **def** batch\_norm\_layer**(**x**,** train\_phase**,** scope\_bn**):**  **with** tf**.**variable\_scope**(**scope\_bn**):**  # 新建两个变量，平移、缩放因子  beta **=** tf**.**Variable**(**tf**.**constant**(**0.0**,** shape**=[**x**.**shape**[-**1**]]),** name**=**'beta'**,** trainable**=True)**  gamma **=** tf**.**Variable**(**tf**.**constant**(**1.0**,** shape**=[**x**.**shape**[-**1**]]),** name**=**'gamma'**,** trainable**=True)**    # 计算此次批量的均值和方差  axises **=** np**.**arange**(**len**(**x**.**shape**)** **-** 1**)**  batch\_mean**,** batch\_var **=** tf**.**nn**.**moments**(**x**,** axises**,** name**=**'moments'**)**  # 滑动平均做衰减  ema **=** tf**.**train**.**ExponentialMovingAverage**(**decay**=**0.5**)**  **def** mean\_var\_with\_update**():**  ema\_apply\_op **=** ema**.**apply**([**batch\_mean**,** batch\_var**])**  **with** tf**.**control\_dependencies**([**ema\_apply\_op**]):**  **return** tf**.**identity**(**batch\_mean**),** tf**.**identity**(**batch\_var**)**  # train\_phase 训练还是测试的flag  # 训练阶段计算runing\_mean和runing\_var，使用mean\_var\_with\_update()函数  # 测试的时候直接把之前计算的拿去用 ema.average(batch\_mean)  mean**,** var **=** tf**.**cond**(**train\_phase**,** mean\_var\_with\_update**,**  **lambda:** **(**ema**.**average**(**batch\_mean**),** ema**.**average**(**batch\_var**)))**  **## 用tf.nn.batch\_normalization实现如下:**    normed **=** tf**.**nn**.**batch\_normalization**(**x**,** mean**,** var**,** beta**,** gamma**,** 1e-3**)**  **return** normed |

|  |
| --- |
| **def** batch\_normalization**(**x**,**  mean**,**  variance**,**  offset**,**  scale**,**  variance\_epsilon**,**  name**=None):**  **## 实现公式**    **with** ops**.**name\_scope**(**name**,** "batchnorm"**,** **[**x**,** mean**,** variance**,** scale**,** offset**]):**  inv **=** math\_ops**.**rsqrt**(**variance **+** variance\_epsilon**)**  **if** scale **is** **not** **None:**  inv **\*=** scale  **return** x **\*** inv **+** **(**offset **-** mean **\*** inv  **if** offset **is** **not** **None** **else** **-**mean **\*** inv**)** |

三 BN的优点

**没有它之前，需要小心的调整学习率和权重初始化，但是有了BN可以放心的使用大学习率**，但是使用了BN，就不用小心的调参了，较大的学习率极大的提高了学习速度，

Batchnorm本身上也是一种正则的方式，可以代替其他正则方式如dropout等

另外，个人认为，batchnorm降低了数据之间的绝对差异，有一个去相关的性质，更多的考虑相对差异性，因此在分类任务上具有更好的效果。