Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

深度学习中的Normalization

苏霞　　硕士

2019年10月20日

内容提要

- 背景简介
- 基本概念
- 算法原理
- 应用总结
- 参考文献

预期收获

- 预期收获
  - 1. 了解深度学习中各Normalization模型的算法原理
  - 2. 探讨Normalization有效的深层原因

背景简介

背景简介

- 为什么需要Normalization
  - 深度学习中的 Internal Covariate Shift现象：由于深度网络由很多隐层构成，在训练过程中由于底层网络参数不断变化，导致上层神经元净激活值的分布逐渐发生很大的变化和偏移。

背景简介

- 为什么需要Normalization
  - Internal Covariate Shift现象带来的后果：

1、上层参数需要不断适应新的输入数据分布，降低学习速度

2、深度网络中微小的参数变动引起梯度上的剧变，导致训练陷入非线性饱和区，使得学习过早停止

3、每层参数的更新都会影响到其它层，因此参数更新策略需要尽可能的谨慎。需要使用更小的学习率，参数初始化也需要更为谨慎的设置

背景简介

- 为什么需要Normalization
  - “白化（whitening）”是一个重要的数据预处理步骤。通过白化每一层的输入，实现固定的输入分布，将消除Internal Covariate Shift的不良影响。
  - 标准的白化操作代价高昂，并且我们希望白化操作是可微的，保证白化操作可以通过反向传播来更新梯度。

基本概念

基本概念

- 基本概念
  - Normalization到底是在做什么

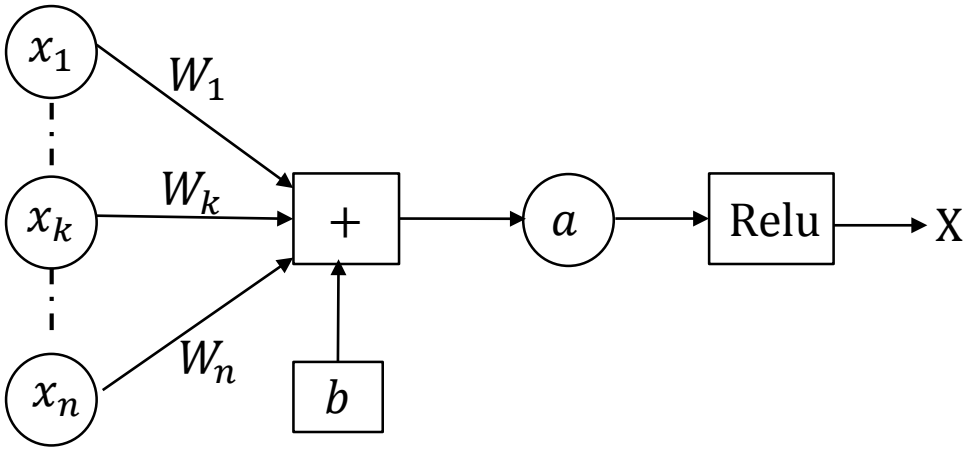
Normalization是一种对数值的特殊函数变换方法，也就是说假设原始的某个数值是x，套上一个起到规范化作用的函数，对x进行转换，形成一个规范化后的数值

$$\hat{x} = f(x)$$

并且我们希望转换后的数值$\hat{x}$满足一定的特性

基本概念

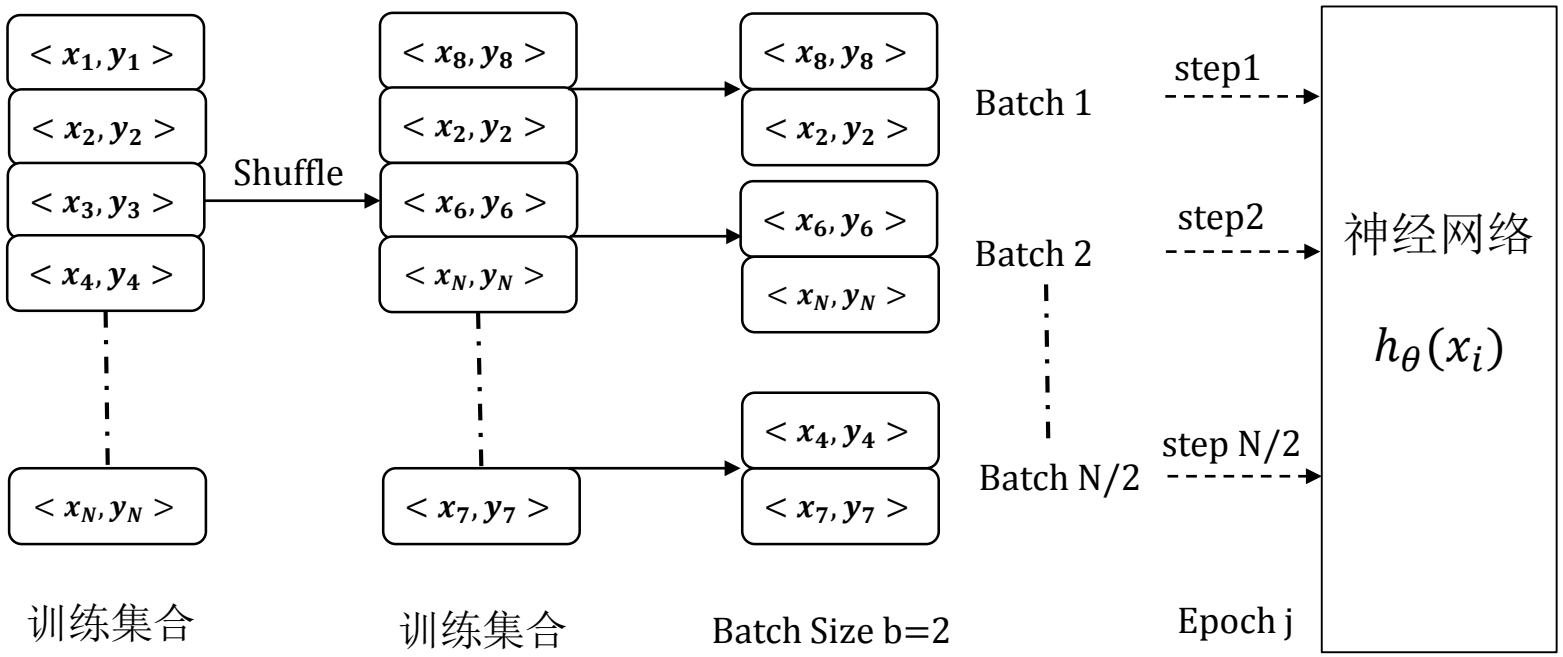
- 基本概念
  - 神经元



步骤一：对输入数据进行线性变换，产生净激活值
步骤二：套上非线性激活函数，得到激活值。神经网络的非线性能力来自于此

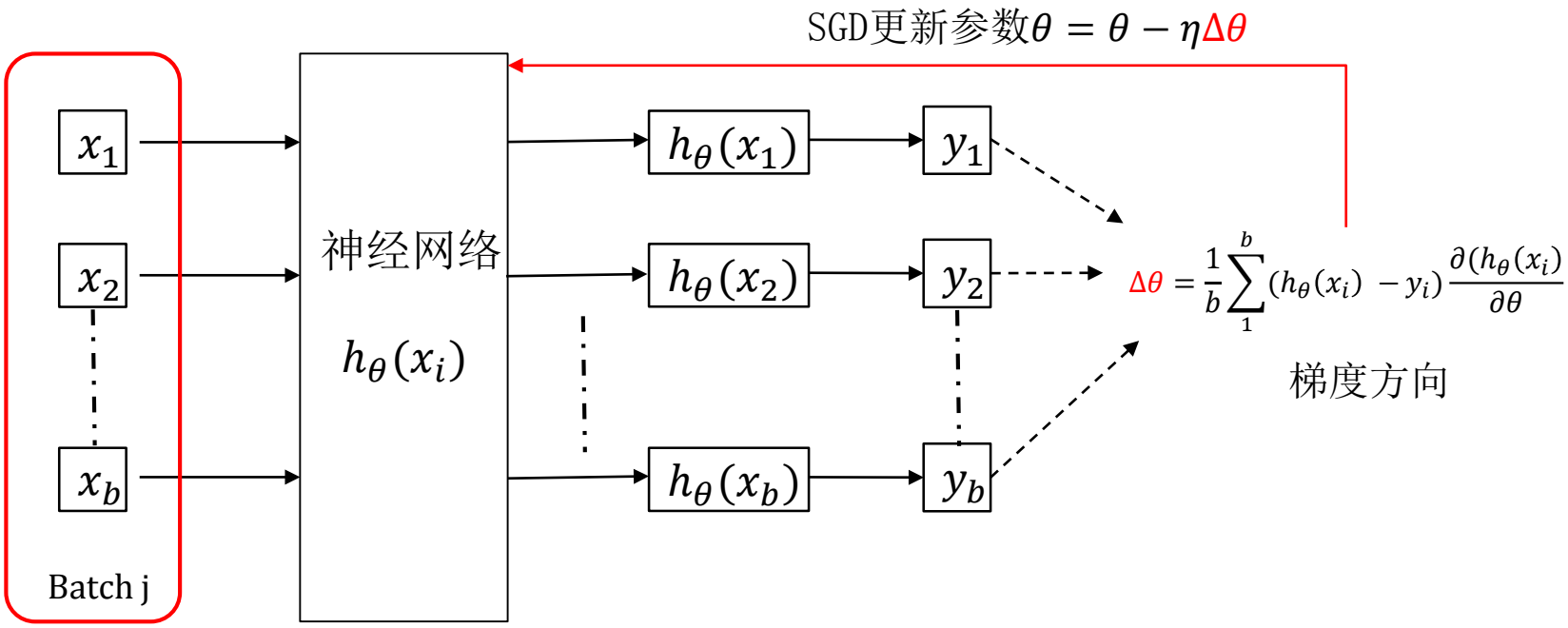
基本概念

- 基本概念
  - Mini-Batch SGD



基本概念

- 基本概念
  - Mini-Batch SGD



Mini-Batch内的每个实例需要走一遍当前的网络，产生当前参数下神经网络的预测值

算法原理

算法原理

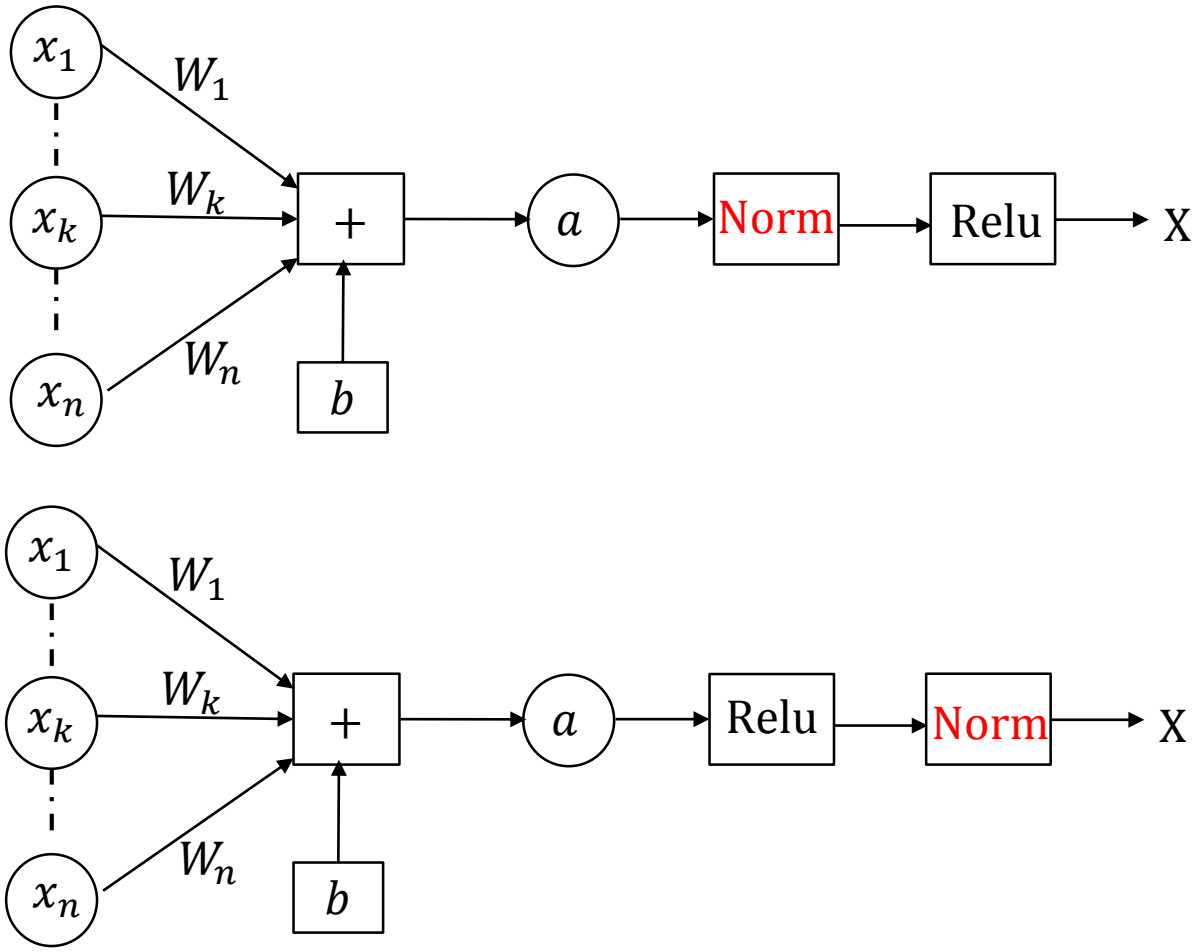
T	加快神经网络训练时的收敛速度，提高准确率
I	神经元接收的一组输入向量X
P	对X进行平移和伸缩变换
0	固定区间范围的标准分布向量$\hat{X}$

算法原理

- 深度学习中的Normalization
  - 对第L层神经元的净激活值或者说对第L+1层神经元的输入值进行Normalization操作，比如BatchNorm/LayerNorm/InstanceNorm/GroupNorm等
  - 对神经网络中连接相邻隐层神经元之间的边上的权重进行规范化操作，比如WeightNorm

算法原理

- 深度学习中的Normalization



算法原理

- 深度学习中的Normalization
  - 对净激活值规整到均值为0，方差为1的正态分布范围内

$$\tau = \frac{a_i - \mu}{\sigma_i}$$

μ是通过神经元集合S中包含的m个神经元各自的净激活值求出的均值，即：

$$\mu = \frac{1}{m}\sum_{k=1}^{m} a_k \quad k \epsilon S \; and \; \| S \| = m$$

σ_i为根据均值和集合S中神经元各自净激活值求出的标准差：

$$\sigma_i = \sqrt{\frac{1}{m}\sum_{k=1}^{m}(a_k - \mu)^2 + \varepsilon} \quad k \epsilon S \; and \; \| S \| = m$$

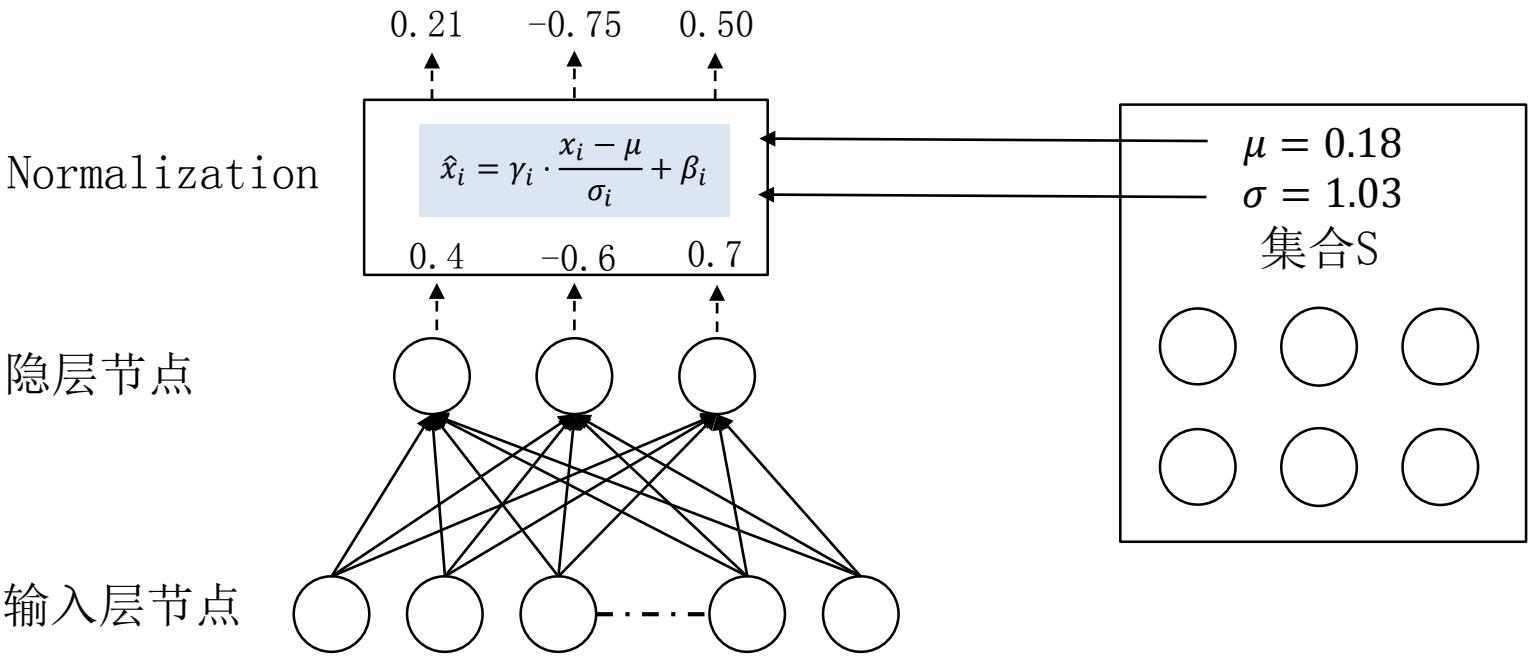
算法原理

- 深度学习中的Normalization
  - 让每个神经元在训练过程中学习到对应的两个调节因子，对规范到0均值，1方差的值进行微调。因为经过第一步操作后，Normalization有可能降低神经网络的非线性表达能力，所以会以此方式来补偿Normalization操作后神经网络的表达能力。

$$a_i^{norm} = \gamma_i \cdot \tau + \beta_i$$

算法原理

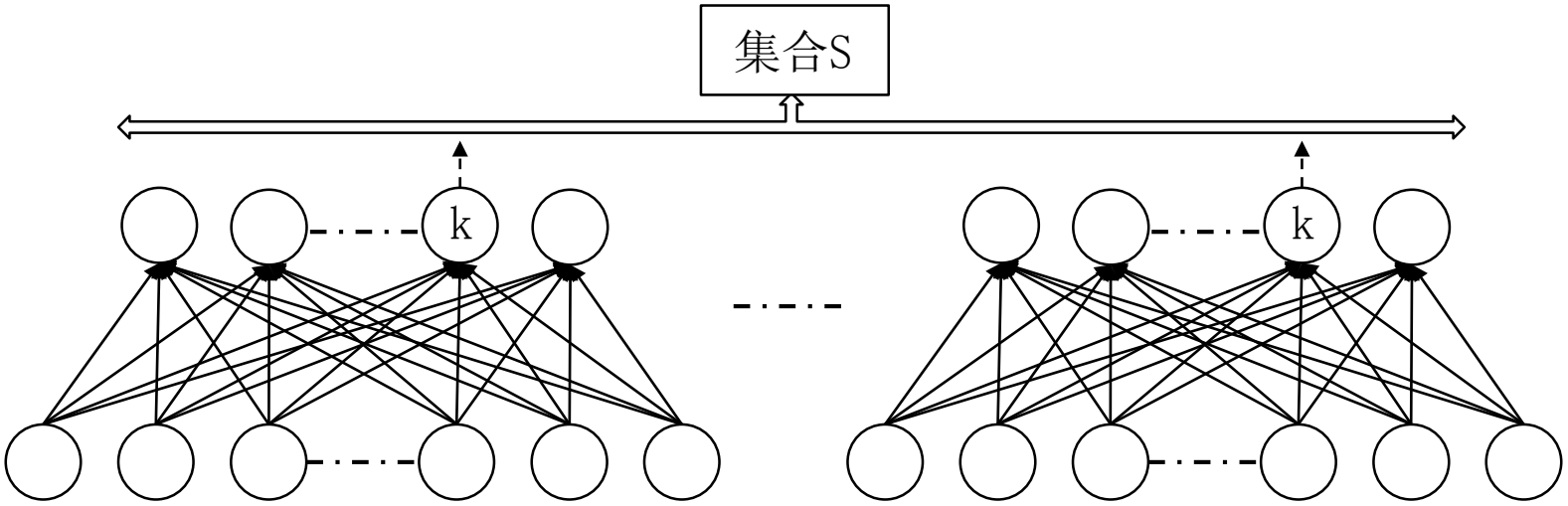
- 深度学习中的Normalization



在实际的计算中，隐层中的神经元可能共用同一个集合S，也可能每个神经元采用不同的神经元集合S

算法原理

- Batch Normalization
  - 前向神经网络中的BN



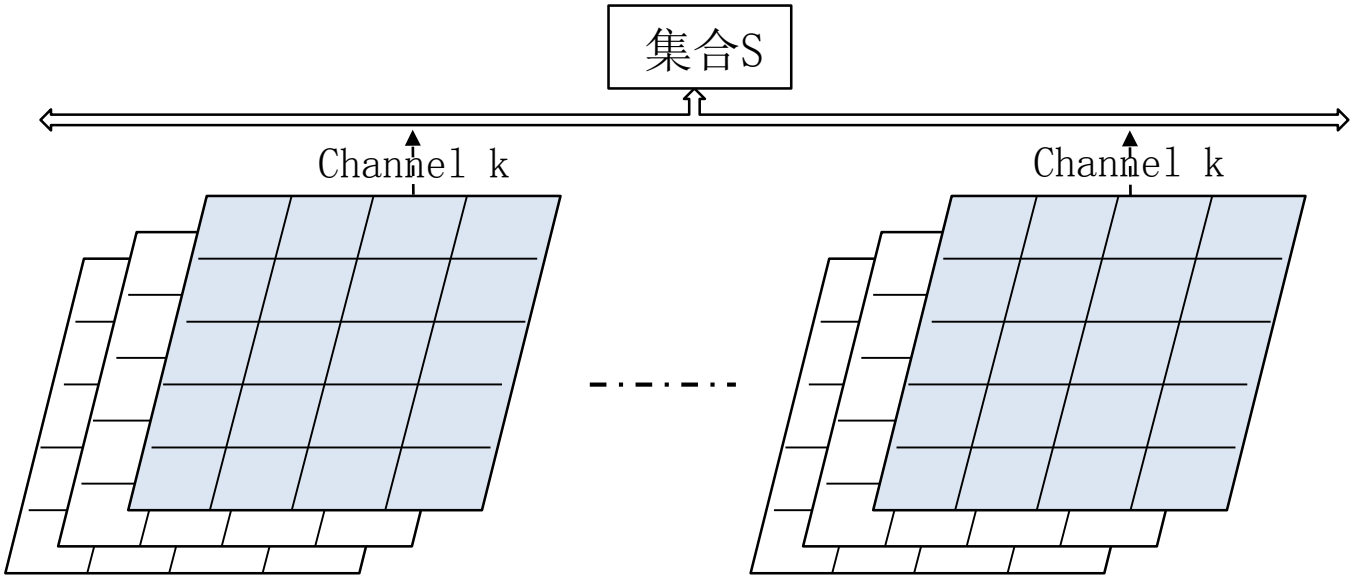
Batch i

实例x_1

实例x_n

算法原理

- Batch Normalization
  - CNN网络中的BN



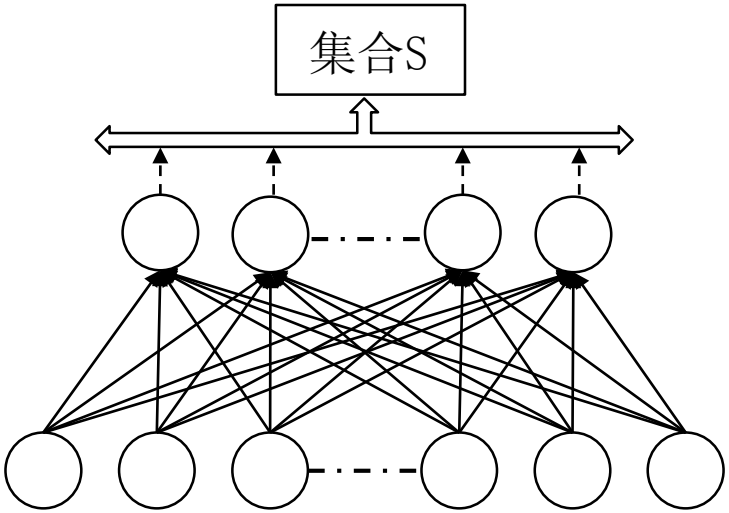
Batch i　　实例x_1　　实例x_n

算法原理

- Batch Normalization
  - 如果Batch Size太小，则BN效果明显下降
  - 对于有些像素级图片生成任务来说，BN效果不佳
  - RNN等动态网络使用BN效果不佳且使用起来不方便
  - 训练时和预测时统计量计算方法不一致

算法原理

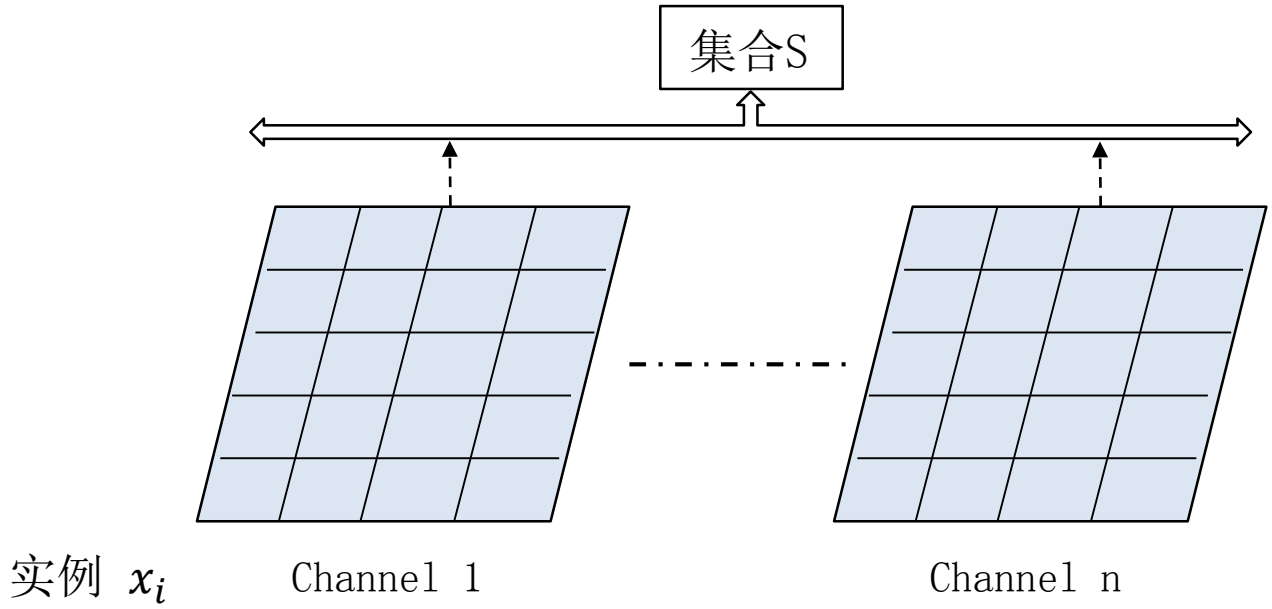
- Layer Normalization
  - 前向神经网络中的LN



MLP的同一隐层自己包含了若干神经元，直接用同层隐层神经元的净激活值作为集合S的范围来求均值和方差

算法原理

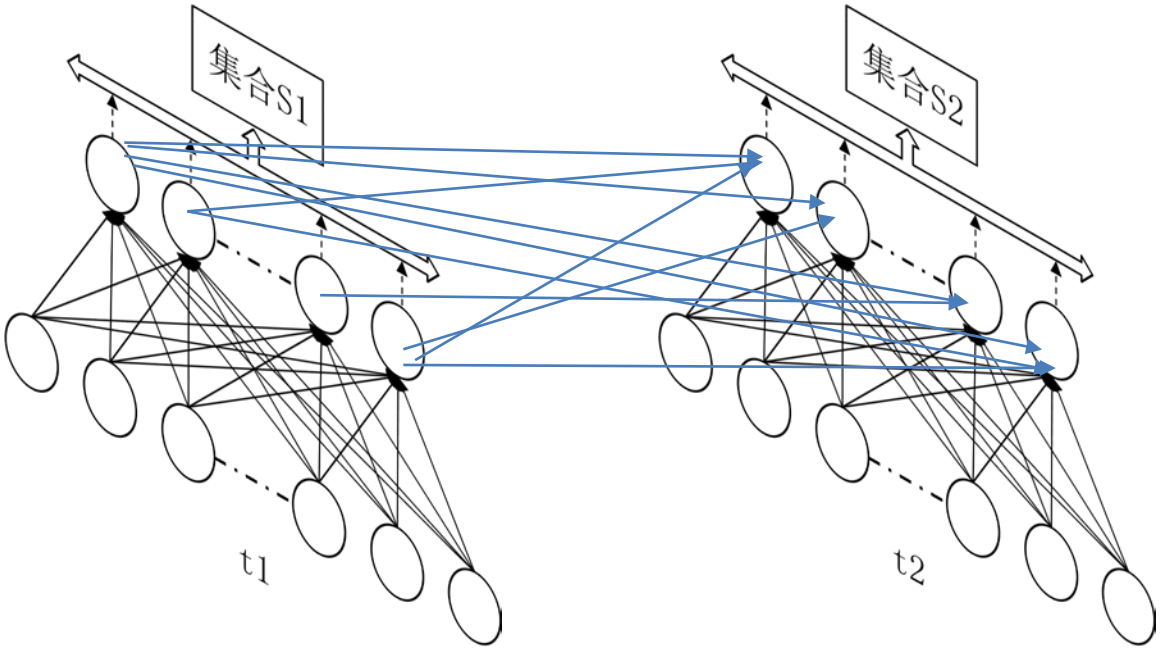
- Layer Normalization
  - CNN网络中的LN



CNN中同一个卷积层包含n个输出通道，每个通道包含m*1个神经元,整个通道包含了n*m*1个神经元

算法原理

- Layer Normalization
  - RNN网络中的LN



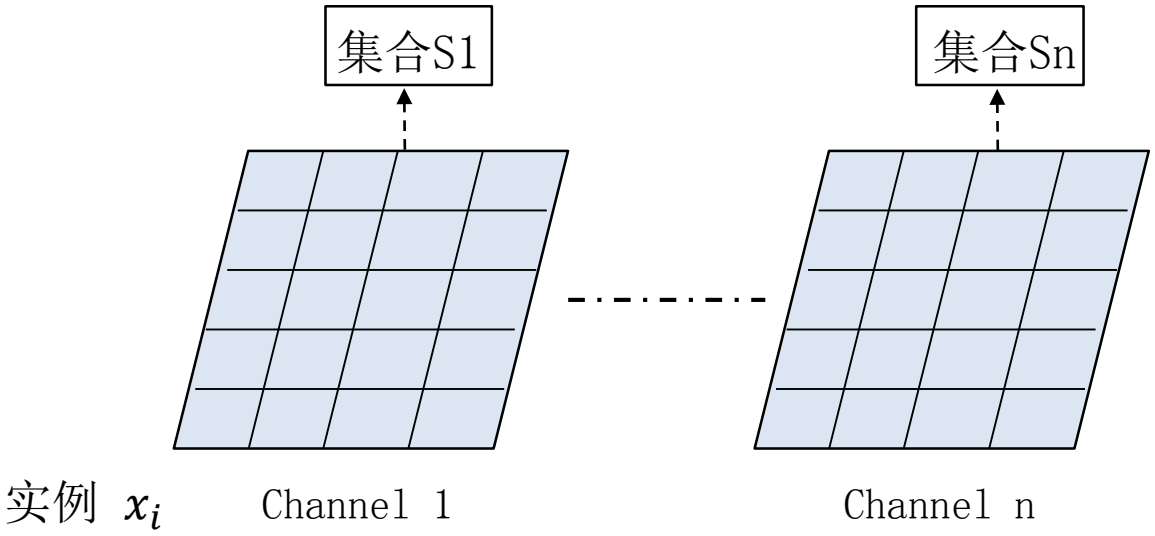
RNN的每个时间步的隐层也包含了若干神经元

算法原理

- Layer Normalization
  - Layer Normalization目前只适合应用在RNN场景下，在CNN等环境下效果是不如BatchNorm或者GroupNorm等模型的

算法原理

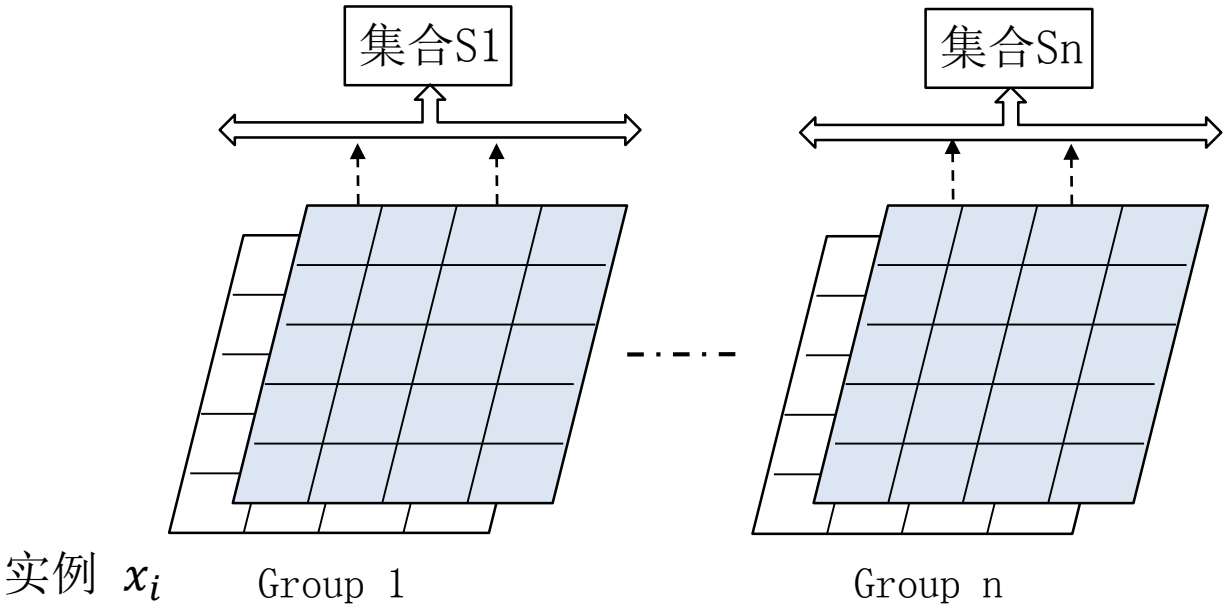
- Instance Normalization
  - CNN网络中的IN



对于某个卷积层来说，每个输出通道内的神经元会作为集合S来统计均值方差。对于RNN或者MLP，如果在同一个隐层类似CNN这样缩小范围，那么就只剩下单独一个神经元，输出也是单值而非CNN的二维平面，这意味着没有形成集合S，所以RNN和MLP是无法进行Instance Normalization操作的

算法原理

- Group Normalization
  - CNN网络中的GN



对CNN中某一层卷积层的输出通道进行分组，在分组范围内进行统计

算法原理

- Normalization为何有效
  - 权重伸缩不变性（weight scale invariance）

  指的是当权重W进行伸缩时，得到的规范后的值保持不变。

  - 效果

  1、权重伸缩不变性避免了反向传播时因为权重过大或过小导致的梯度消失或梯度爆炸问题，从而加速了神经网络的训练。

  2、权重伸缩不变性还具有参数正则化的效果，避免参数的大幅震荡，提高网络的泛化性能。可以使用更高的学习率。

算法原理

- Normalization为何有效
  - 权重伸缩不变性（weight scale invariance）

$$a_i = W_i \cdot X$$

$$a_i' = \varphi W_i \cdot X$$

$$\mu' = \frac{1}{m}\sum_{k=1}^{m} a_k' = \frac{1}{m}\sum_{k=1}^{m} (\varphi W_k \cdot X) = \varphi\mu$$

$$\sigma_i' = \sqrt{\frac{1}{m}\sum_{k=1}^{m}(a_k' - \mu')^2} = \sqrt{\frac{1}{m}\sum_{k=1}^{m}(\varphi a_k - \varphi\mu)^2} = \varphi\sigma_i$$

$$\tau' = \frac{a_i' - \mu'}{\sigma_i'} = \frac{\varphi a_i - \varphi\mu}{\varphi\sigma_i} = \tau$$

算法原理

- Normalization为何有效
  - 数据伸缩不变性（data scale invariance）

  指的是当数据X进行伸缩时，得到的规范后的值保持不变。

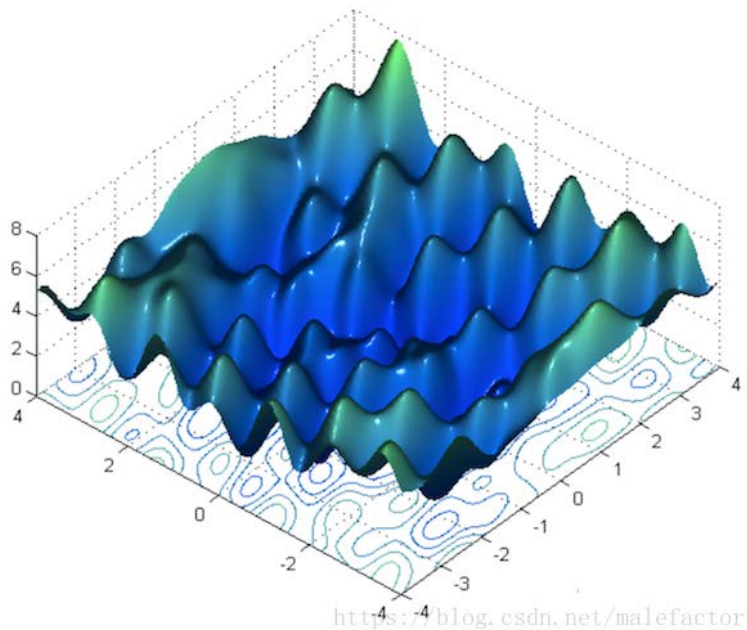
  - 效果

  数据伸缩不变性可以有效地减少梯度弥散，简化对学习率的选择：每一层神经元的输出依赖于底下各层的计算结果。如果没有规范化，当下层输入发生伸缩变化时，经过层层传递，可能会导致数据发生剧烈的膨胀或者弥散，从而也导致了反向计算时的梯度爆炸或梯度弥散；数据的伸缩变化也不会影响到对该层的权重参数更新，使得训练过程更加鲁棒，简化了对学习率的选择。

算法原理

- Normalization为何有效
  - 损失曲面

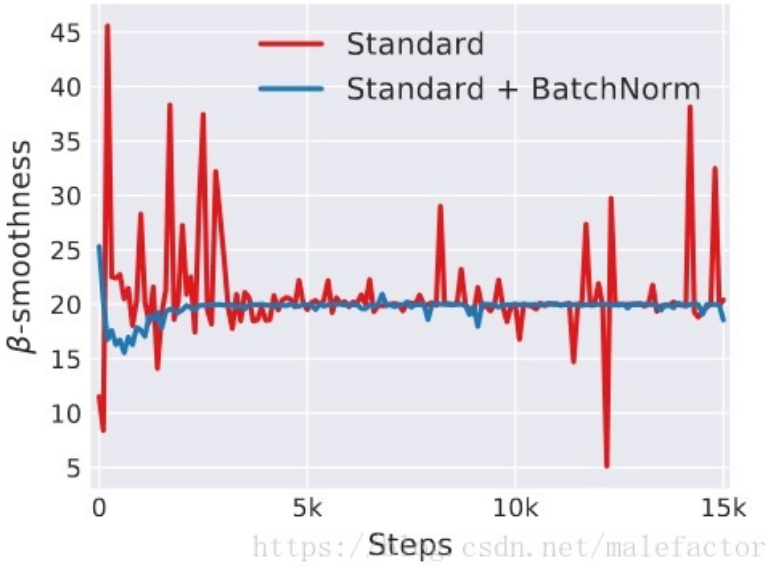
深度网络通过叠加大量非线性函数来解决非凸复杂问题，导致损失曲面形态异常复杂，大量空间坑坑洼洼相当不平整



算法原理

- Normalization为何有效
  - 损失曲面

用L-Lipschitz函数来评估损失曲面的平滑程度，L越小曲面越平滑，L越大，则曲面越凹凸不平。
Normalization操作通过网络参数重整，对非线性非凸问题复杂的损失曲面起到了很好的平滑作用。



$$|f(x_1) - f(x_2)| \leq L \parallel x_1 - x_2 \parallel$$

应用总结

应用总结

- 各种Normalization的适用场景
  - 对于RNN的神经网络结构来说，目前只有LayerNorm是相对有效的
  - 如果是GAN等图片生成或图片内容改写类型的任务，可以优先尝试InstanceNorm
  - 如果使用场景约束BatchSize必须设置很小，考虑使用GroupNorm
  - 而其它任务情形应该优先考虑使用BatchNorm

参考文献

参考文献

- [1] Sergey Ioffe etc. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2015.
- [2] https://blog.csdn.net/malefactor/article/details/82154224.
- [3] https://zhuanlan.zhihu.com/p/33173246.

道德经

五色令人目盲，五音令人耳聋，五味令人口爽，驰骋畋猎令人心发狂，难得之货令人行妨。是以圣人，为腹不为目，故去彼取此。

谢谢！