论文阅读：Batch Normalization——加速网络训练

标签： 神经网络 机器学习 BN 深度学习 训练方法

2016-11-18 22:26 741人阅读 评论(0) 收藏 举报

分类：

机器学习（7）

目录(?) [+]

之前有一些事情，博客停更了一段时间。期间也学了不少东西，打算挑一些慢慢地整理出来。今天就先聊一聊Batch Normalization，这是一种能够大大提高深度神经网络训练速度的方法。虽然是15年发表的，也不是很久，但已经被大家广泛使用了，其作用和重要性可见一斑。这里就记录一下我阅读这篇文章*Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*的理解吧，小弟也是刚接触深度学习不久，理解难免有偏差，写出来也是希望和大家相互交流，还望不吝指正。^_^

摘要

本文要解决的问题：

深度神经网络训练起来比较困难的原因是在训练的过程中，每层的输入的分布会随着前一层的参数的改变而发生变化。

这种变化就需要降低学习速率和小心地进行参数的初始化，从而减慢了训练过程。变化的输入还会产生一个众所周知的问题，那就是输入有可能进入到激活函数的饱和区，造成饱和问题。

关闭

方法概述

本文把这种现象称为*internal covariate shift*。解决的方法就是对每一层的输入进行normalization。本文把normalization作为网络模型结构的一部分嵌入进去，对每一组training mini-batch进行normalization。

这样做的好处：

- 可以让我们使用更高的学习速率；

- 不再那么小心翼翼地对待初始化的问题；
- 它还可以作为正则项，使我们不再依赖Dropout。

简介：

网络训练方法SGD遇到的问题

SGD（stochastic gradient descent，可参考 梯度下降法（Gradient Desce
深度网络常用的方法，它使用一个mini-batch来估计损失函数对参数的梯度。
个样本比起来，这样做有很多好处。比如：

1. 在一个mini-batch上的损失函数的梯度是全体训练集的一个较好的估计
2. 对于现在的计算平台来说，由于并行运算技术，一次计算一组由m个样本组成的batch，效率远远高于进行m次单样本的计算。

SGD的一个问题就是需要小心调整学习率和初始化参数。训练过程复杂的原因在于，每一层的输入受它前面所有层的影响，随着网络深度的增加，前面网络参数微小的变
大。

为什么当每一层输入的分布发生变化时会影响训练速度呢？因为这一层要不断地调整自己的参数，去适应新的分布，因此就会在训练过程中出现反反复复不停地更新参数，从而降低了收敛的速度。

解决思路

其实这个问题很早就有人提出过了，它被称为协方差偏移(*covariate shift*)。*covariate shift*原来是指一个学习系统输入分布的变化，这里把它扩展到了系统的一部分，比如一个子网络或者一个层，因此叫*internal covariate shift*。这也很容易理解，我们可以把每个层看做是一个独立的系统，它的输入就是上一层的输出，系统的一部分的工作模式和整个系统的工作模式是相同的。因此，对于一个神经网络，如果它的输入的某种分布特性可以使训练更有效，那么对于这个网络的每个part，也就是每个层，这种分布也应该是很高效的。这是什么意思呢？就比如说，我们常常会将网络的输入做一些预处理，比如零均值和单位化方差，这样有利于网络的训练，那么对于网络每层的输入，也就是网络的中间数据，如果也做这样的处理，应该对网络的训练也有帮助。这样的话，如果输入x的分布在时间上是不变的，那么这一层的参数也不需要为了适应x而重新调整。

刚才说了分布如果能够固定下来，对子网络或者是一个层的好处，其实它也对子网之外的部分有好处，那就是减小了饱和问题。

考虑sigmoid激活函数$g(x)=\frac{1}{1+exp(-x)}$，$x=Wu+b$，随着$|x|$的增大，向量$\mathbf{X}$的大部分维度都会进入$g(x)$的饱和区，饱和区的导数很小，在反向传播中流给u的梯度就很小，造成梯度小时的问题，减慢收敛速度。因此有人在实践中提出使用ReLU激活函数和小心地初始化来处理这个问题。但如果我们能把上一层的输出稳定下来，那么它就不太可能陷在饱和区域中了。

本文在这样的想法上，提出了Batch Normalization，来减小*internal covariate shift*。除了上面的好处，它还有利于梯度的传播，因为它稳定了每一层输入的分布在非饱和区，因此减小了反向传播中参数大小规模对梯度的影响，以及初始化的影响，可以让我们使用更高的学习

关闭

速率。

减小Internal Covariate Shift

为了使输入数据得分布保持稳定，常用的处理的方法有白化（利用线性变换使数据具有零均值和单位方差）与去相关。根据前面的分析，这些技术应该对于网络中间的数据也有效。那么能不能只在在前向过程应用这些normalization，而在反向传播忽视它呢，答案是否定的。论文中给出了这样一个例子：

假如某一层的输入为u，输出为x，$x = u + b, \mathcal{X} = \{x_{1...N}\}$是整个训

$$E[x] = \frac{1}{N}\sum_{i=1}^{N} x_i$$

如果在反向过程忽略normalization，即忽略$E[x]$对b的依赖，那么更新：

$$b \leftarrow b + \Delta b$$

而：

$$\Delta b \propto -\frac{\partial \ell}{\partial \hat{x}} \cdot \frac{\partial \hat{x}}{\partial x} \cdot \frac{\partial x}{\partial b} = -\frac{\partial \ell}{\partial \hat{x}} \cdot 1 \cdot 1$$

因此更新后：

$$u + (b + \Delta b) - E[u + (b + \Delta b)] = u + b - E[u + b] = x - E[x]$$

我们发现，它和更新前的输出一样！那这样进行下去的话，b会不断更新，而输出或者损失都没有变，最终b会大到爆炸(blow up)！

因此在反向过程中还是要考虑normalization的。更

关闭

$$\hat{x} = Norm(x, \mathcal{X})$$

反向传播过程中，我们需要计算：

$$\frac{\partial Norm(x,\mathcal{X})}{\partial x}, \frac{\partial Norm(x,\mathcal{X})}{\partial \mathcal{X}}$$

也就是把训练集中的每个样本的影响都考虑进去。然而，这个计算量是非常庞大的。因此，我们需要寻找另外的不需要计算整个数据集样本的方法。

Normalization via Mini-Batch Statistics

Identity Transform

由于在整个训练集上同时对所有输入求梯度有困难，本文就提出对每维特征单独做normalization。对于d维输入$x = (x^{(1)} \cdots x^{(d)})$，我们对每维做如下normalization：

$$\hat{x} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}}$$

然而这样的normalization可能会改变这一层网络的表征能力，比如对于一个sigmoid激活函数，可能就把限制在了它的线性区域，失去了非线性激活的能力。因此我们要赋予normalization能够进行单位变换(identity transform)的能力。这可以通过在后面再加一层缩放和平移实现：

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}$$

上式中的参数是在训练过程中学习得到的，可以发现，如果$\gamma^{(k)} = \sqrt{Var[x^{(k)}]}, \beta^{(k)} = E[x^{(k)}]$，那么$y^{(k)} = x^{(k)}$。这样我们就赋予了normalization能够进行单位变换(identity transform)的能力，至于学到的参数到底是不是identity transform，那就看训练的结果了。

关闭

为了降低运算量，我们上述的normalization过程在一个mini-batch中进行。能够这么做的根据在于，我们认为在统计意义上mini-batch可以对整个训练集的均值和方差进行估计。于是整个训练过程就如下算法所述（省略了上标k）：

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

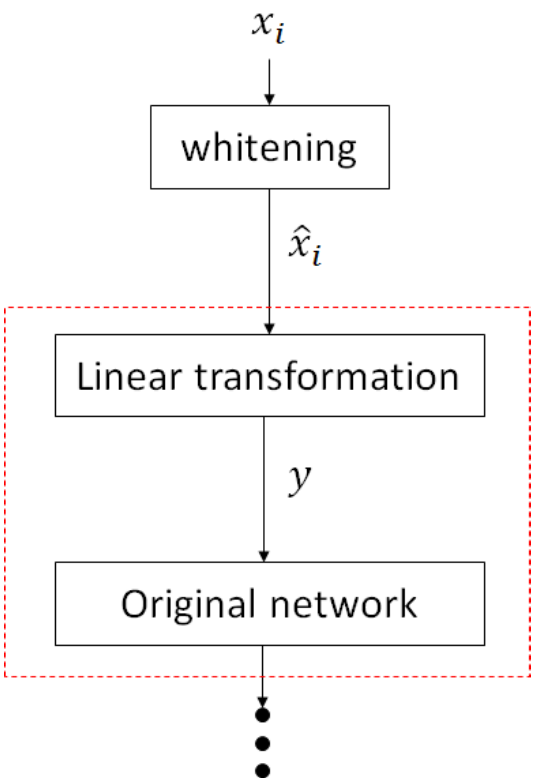
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

为了便于理解数据变换的过程，我画了一个图：



关闭

对于上图红色虚线框以及之后的网络来说，它的输入$\hat{x}_i$就具有固定的均值和方差，从而加速后面子网络的训练，进而加速整个网络。

梯度的反向传播

利用链式法则，不难求出反向传播的梯度：

$$\frac{\partial \ell}{\partial \widehat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$
$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$
$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m}$$
$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$
$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i$$
$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

这里需要注意的是，与普通的层不同，BN层中一个样本的前向传播的计算
个mini-batch中其它的样本的，而不是相互独立的，所以$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2}$是将所有样本传
而不是样本独立计算情况下的取平均。其它参数同理。

网络训练与推断（Training and Inference）

有了前面的BN transformation，我们就可以把它嵌入到一般的网络中，需要修改的就是原来将x作为输入的，现在输入变成了$y = BN(x)$。在inference阶段，由于不需要反向传播，我们就不需要用mini-batch的方差和均值了，我们可以利用整个训练集的统计特性做inference，即：

$$\hat{x} = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}}$$

其中：

$$Var[x] = \frac{m}{m-1} \cdot E_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

是方差的无偏估计，其中的期望是训练集所有mini-batch的采样方差的均值。

在inference阶段，γ和β的值不会发生变化，所以BN(x)的操作实际上就变成了一个参数固定的线性变换。

将BN嵌入到网络中训练batch-normalized networks的过程如下：

关闭

Input: Network N with trainable parameters Θ;
subset of activations $\{x^{(k)}\}_{k=1}^{K}$
Output: Batch-normalized network for inference, $N_{\text{BN}}^{\text{inf}}$
1: $N_{\text{BN}}^{\text{tr}} \leftarrow N$ // Training BN network
2: **for** $k = 1 \ldots K$ **do**
3: Add transformation $y^{(k)} = \text{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\text{BN}}^{\text{tr}}$ (Alg. 1)
4: Modify each layer in $N_{\text{BN}}^{\text{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead
5: **end for**
6: Train $N_{\text{BN}}^{\text{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
7: $N_{\text{BN}}^{\text{inf}} \leftarrow N_{\text{BN}}^{\text{tr}}$ // Inference BN network with frozen // parameters
8: **for** $k = 1 \ldots K$ **do**
9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.
10: Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$\text{E}[x] \leftarrow \text{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\text{Var}[x] \leftarrow \frac{m}{m-1}\text{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

11: In $N_{\text{BN}}^{\text{inf}}$, replace the transform $y = \text{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\text{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\text{E}[x]}{\sqrt{\text{Var}[x]+\epsilon}}\right)$
12: **end for**

Algorithm 2: Training a Batch-Normalized Network

第一个循环中修改输入网络的结构，插入BN；接着训练网络，寻找最优参数；最后的inference阶段将BN的参数固定，变为线性变换，进行样本的推断。

卷积网络的Batch Normalization

对于卷积神经网络，作者也希望BN有一个“卷积”的工作方式。就是对于同一个feature map的不同位置，使用相同的参数$\gamma^{(k)}$和$\beta^{(k)}$，而不是对每一维都用一对不同的参数。

Batch-Normalized的好处主要就是可以让我们使用更高的学习速率而不必担心模型爆炸（model explosion）的问题，加速了训练过程。同时它处理每个训练样本时会利用到整个mini-batch中所有样本之间的联系，起到正则化的作用，网络的泛华能力，从而不再依赖于Dropout。

关闭

实验

这里就略去了，有兴趣的同学可以去看看论文。

Reference

Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[J]. *arXiv preprint arXiv:1502.03167*, 2015.

- 上一篇 生成学习算法（Generative Learning Algorithms）
- 下一篇 神经网络的可解释性——Network Dissection: Quantifying Interpretability of De Representations

相关文章推荐

- 深度学习中的数学与技巧(2):《Batch Normalizatio...
- Batch Normalization —— 加速深度
- MySQL在微信支付下的高可用运营--莫晓东
- 腾讯云容器服务架构实现介绍--董晓
- Batch Normalization 的原理解读
- Batch Normalization 神经网络加速算法
- 容器技术在58同城的实践--姚远
- 微博热点事件背后的数据库运维心得--张冬洪
- Batch Normalization & Layer Normalization整理...
- Batch Normalization论文翻译——中英文对照
- SDCC 2017之容器技术实战线上峰会
- Batch Normalization论文翻译——中文版
- 深度学习（二十九）Batch Normalization 学习笔记
- 论文笔记——《Batch Normalization Accelerating...
- SDCC 2017之数据库技术实战线上峰会
- 论文笔记——Batch Normalization

我的更多文章

梯度下降法（Gradient Descent）(2016-04-08 18:20:09)

关闭