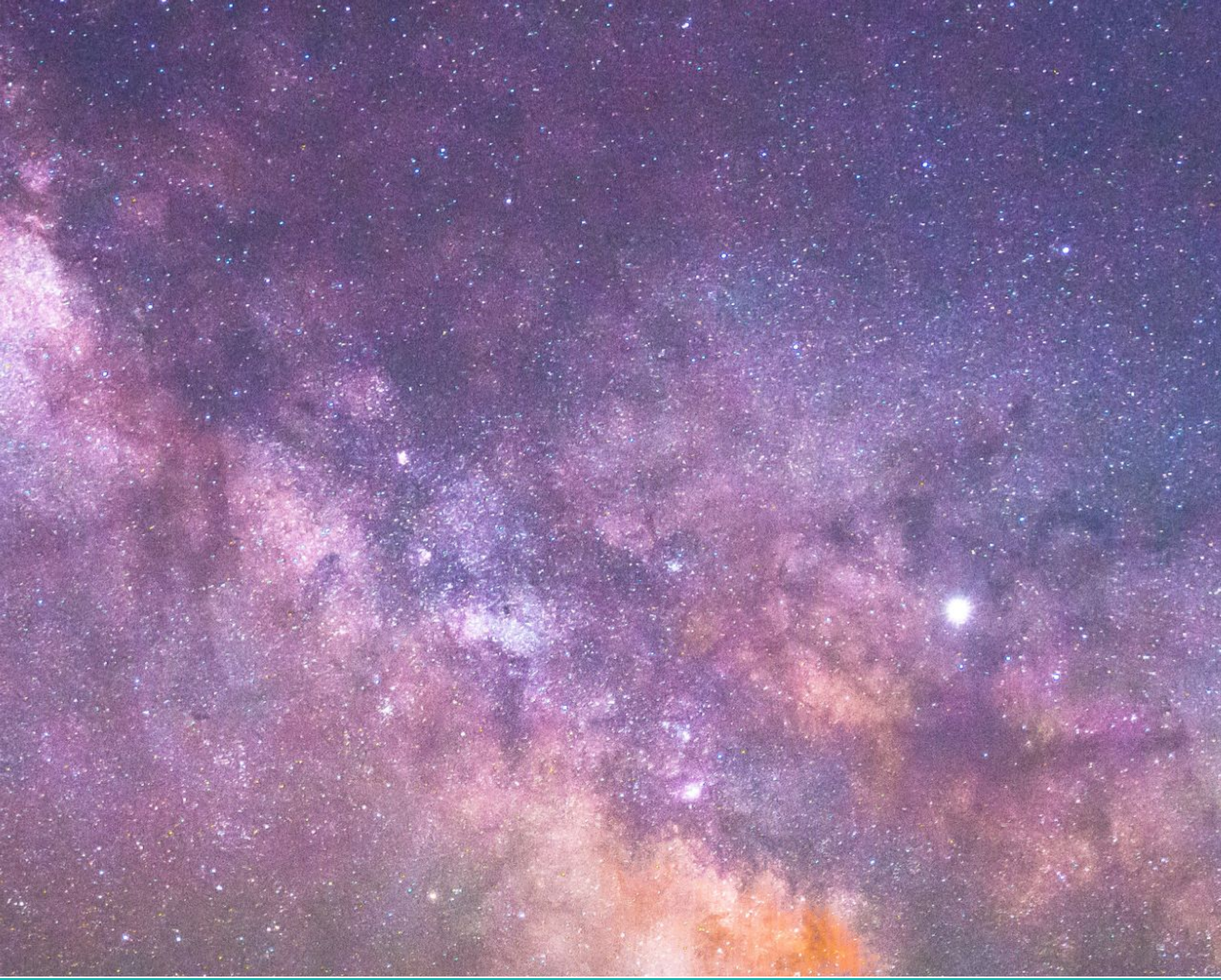


模型可解释性

作者：台运鹏

时间：May 4, 2021

版本：1.0

What I can’t create, I don’t understand. ——Richard Feynman

特别声明

深度学习其实是一种实验型科学，很多时候都是有了实验结果之后，然后人们再去寻找理论去证明实验中出现的现象，但其实很多所谓的证明都很牵强，关于深度学习模型的可解释性还有很长一段路要走。根据深度学习发展的趋势，调参终将会成为过去时，只是早晚的区别罢了。以下是我的观点：

1、深度学习的应用十分广泛，未来大抵也会如此，对于一个应用极广的领域，不同的应用场景会迫使工程师们了解模型，否则无法适应大规模应用。

2、工程师们一直期望做出真正的人工智能，对于技术的极致追求必然会遭遇到瓶颈，只有真正从底层了解出一个模型，才真正有可能提升它的性能，靠一次两次的调参与其说是提升，不如说是自我安慰。

因此，我打算用这本书来记录自己对于深度学习模型可解释性的研究，会有自己的看法，讨论，以及问题。也希望能多了解模型一点，将自己的想法传播出去，让更多人看到，就算文章写的不好，传播一下可解释性的必然趋势也是值得的。

本人自认才疏学浅，仅略知皮毛，更兼时间和精力有限，书中错谬之处在所难免，若蒙读者诸君不吝告知，将不胜感激。

另外，本书采用的是 ElegantBook 的开源 LATEX 模版。

The purpose of computing is insight, not numbers. ——Richard Hamming

台运鹏

May 4, 2021

目录

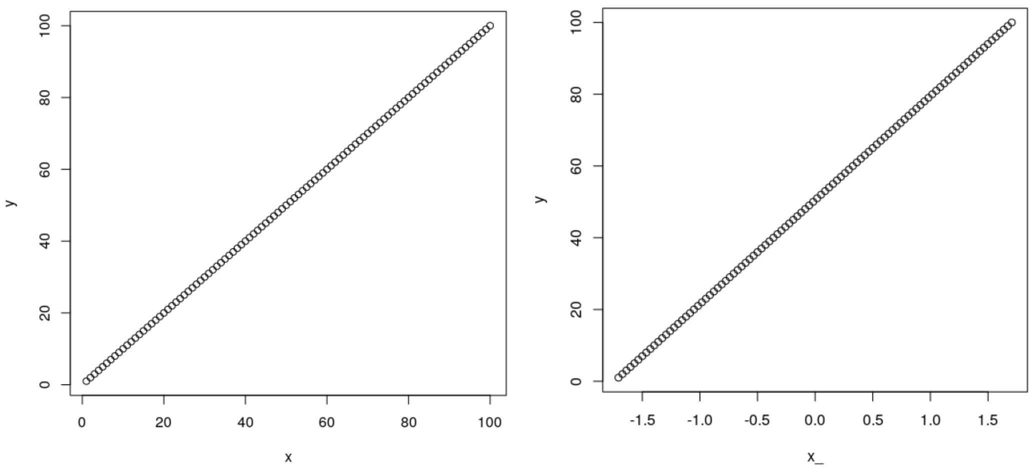
第 1 章 Normalization

做神经网络训练的时候，Normalization 已经成了基操，近些年大火的 Transformer 模型也用 Layer Normalization 来提升性能，在最具有代表性的 MNIST 数据集上的手写数字识别中需要用到 Batch Normalization，似乎深度学习和 Normalization 是标配一样，但它真的有说的这么厉害吗?

1.1 Background

机器学习有一个重要的假设：训练样本符合一个分布，而每一个样本都是独立地从这个分布中采样出来，也就是独立同分布（independent and identically distributed，简称 $i.i.d.$），而我们训练的样本越多，就越能够获得这个分布的信息，如果想要使我们的模型具有不错的泛化能力，那么，模型就得较为准确地预测出这个分布。虽然并不是必备要求，但是能满足这个假设的数据会提升模型的泛化能力已经成为共识。

深度学习由多层网络组成，输入进去之后，经过若干层的处理之后来到高层(这里的高层指的是最后一层)，然后输出，那么，每一层输出的结果所符合的分布跟这一层输入前的是一致的吗？假设我们一开始的输入的数据符合 $\mu = p, \sigma = q$ 的分布，第一层是一个线性变换，形如 $y = wx+b$，那么第一层输出的数据就符合 $\mu = p+b, \sigma = pw$ 的分布，这还只是一个普通的线性变换，我们的数据分布已经发生了改变，那么，当层数不断增加，经过很多种复杂的函数变换之后，到达高层的数据分布已经面目全非了，换言之，当底层的参数更新时，到达高层的数据分布将会发生剧烈的震荡（虽然不排除不同网络层之间可以抵消的可能性，不过费尽心思叠网络应该不是为了相互抵消），高层的输出每一次都会跟 label 计算损失，然后更新参数，问题是，每一次高层的数据分布都不一样，换言之，每一次的计算损失是只跟这一次相关，因为下一次就换了不同的数据，那岂不是白费功夫了吗？于是就有了 Normalization 这一类操作。



(a) 未操作　　(b) 操作后

图 1.1: 标准化操作示例

1.2 Introduction

网络的输入常为一组向量 $X = (x_1, x_2, \ldots, x_n)$，首先我们会将 X 的均值变为 0，标准差变为 1（具体操作见公式 1.1，其中 μ, σ 是均值和方差，这里仅仅泛指，后面会分不同的方法具体讨论），注意，这里并不是将数据的分布变为正态分布，数据本身的分布并没有改变，改变的只是 X 的均值和标准差，举个例子，我将 x, y 都初始化为 $[1, 100]$，所以图像是一条直线，x 的均值和标准差分别为 50.5，29.01，接下来我会对 x 进行操作，让其

符合 $\mu = 0, \sigma = 1$ 的标准正态分布，图像仍为直线（见图 1.1）。确实 x 的分布变成了标准正态分布，可是整个数据的分布并不会因此改变。

$$\hat{x} = \frac{x-\mu}{\sigma} \tag{1.1}$$

然后我们会对 $\hat{X}$ 进行线性变换，最后送入网络中，其实网络可以被近似为一个函数变换，$h = f(\gamma\hat{X} + \beta)$，这里可能有点费解，为什么已经做好标准化之后还要再加一个线性变换？基于两点原因：

1、其实标准化操作可以看做为一个线性变换，$\hat{x} = \frac{1}{\sigma}x - \frac{\mu}{\sigma}$，我们以两层神经元为例，经过第一层后给它加上一个 Normalization 层然后送入第二层，那么，第一层不是白学了吗？因为你无论如何都会给 X 标准化，虽然可以规范输入，但是会有损失上一层信息的风险，再加一次线性变换应该是两者的折中处理，其实再加一次线性变换是可以还原的，$\gamma = \sigma, \beta = \mu$ 即可，这是最极端的情况，信息完全保留，其他情况就是既要保留上一层输出的信息，还要尽可能将下一层的输入规范到一个区间内，这样不至于每一次给下一层的输入相差太大，从而一定程度减少高层所做的“无用功”。

2、在标准正态分布的曲线中，几乎全部的 x 都在 $[-3, 3]$ 里面，联系常见的激活函数，如 sigmoid 可以发现，x 所在的一整个区间都在 sigmoid 的非饱和区（线性区），那么模型就无从获得非线性的表达能力，再加一次线性变换，可以使一部分数据分布在饱和区，从而增加模型的鲁棒性[1]。

下面具体介绍两种常见的 Normalization:Batch Normalization & Layer Normalization。

1.3 Batch Normalization

网络的输入为向量形式，如 $X = (x_1, x_2, \ldots, x_n)$，$x_t$ 代表一个独立的样本，训练时每一个神经元对应一个样本（图 1.2）。举个例子，在 CNN 中，input shape 常为 $[N, C, H, W]$，分别代表样本数量（这里即为一个 mini-batch 的大小），通道数，高度和宽度。Batch Normalization 作用在 Batch 上，我们可以类比一个样本就是一本书，通道就是书的页码，高度和宽度类比为行数和每行的字数，BN 就是对一个 mini-batch 里单独的每一个通道做操作，举个例子，可以是第一本书的第一页，第二本书的第一页到第 N 本书的第一页（N 是 mini-batch 的大小），将它们的 μ, σ 统计出来，被这一个 mini-batch 中所有的第一个通道共享。所以 BN 是对 $N * H * W$ 个值进行求平均和标准差。回到公式（1.1），BN 里面 μ, σ 都是针对每一个样本进行统计的结果，见公式（1.2），（1.3），其中 N 指的是一个 mini-batch 的大小，另外，ϵ 是任意小量，只是为了保证分母不为 0，从而让计算稳定罢了。这种操作是以 mini-batch 为单位做标准化，因而被叫做 Batch Normalization 吧。

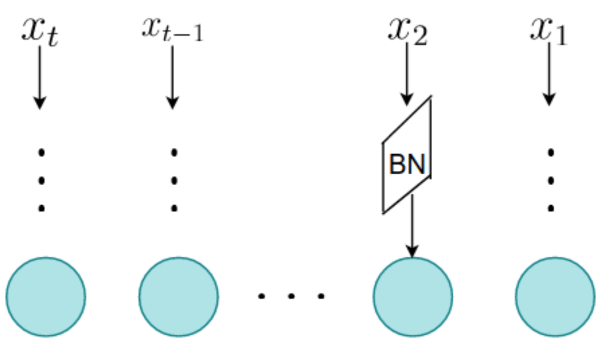


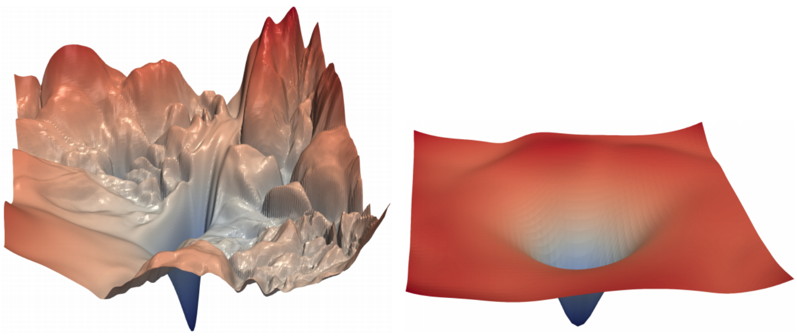
图 1.2: 广义神经网络输入示意图

$$\mu = \frac{1}{N}\sum_i x_i \tag{1.2}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_i (x_i - \mu)^2 + \epsilon} \tag{1.3}$$

那么，新的问题来了：万一 mini-batch 之间数据分布差异很大怎么办？再按照每一个 mini-batch 做标准化，岂不是雪上加霜？其实 BN 有一个默认前提：每一个 mini-batch 的数据应该和其他的 mini-batch 以及和整体都是近似同分布的，换言之，不同的 mini-batch 不过是从整体数据中伴随着噪声随机取样出来的，所以，训练的时候做好 shuffle 是会一定程度提高模型整体水平的。

除了 Introduction 中介绍的可以保留信息，使数据分布更稳定和获得非线性能力之外，还有另一种思路解释 BN 的作用，在一篇论文[2] 中提到 BN 的作用或许是使得目标函数的分布更加平滑（注意论文是建立在 DNN 的线性分析上，并没有添加非线性的激活函数，但可以提供一定的 insight），图上分布的局部最优点较少，SGD 能较快找到最优参数，进而加速训练。图 1.3 是出自论文[3]，他们选取了一两个方向将高维空间投影到三维，当时是探讨 Residual Net 加和没加 residual link 的区别，这里可以类比一下。



(a) without skip connections　　(b) with skip connections

图 1.3: 左右两侧分别为和有无残差相连的目标函数分布图，有残差相连会没有大部分的局部最优点，从而加速训练

另外，从图 1.4 可以看出（同样来自论文[2]），BN 的主要作用其实是加速收敛过程，如上所述，BN 可使目标函数分布更加平滑，可以将学习率一定程度调大，因而 SGD 优化的时候会更快。在大多数情况下，BN 其实不大可能提高准确率，就算有，其实也是比较微小的。同时可以使得数据分布更加稳定，有利于高层的学习。

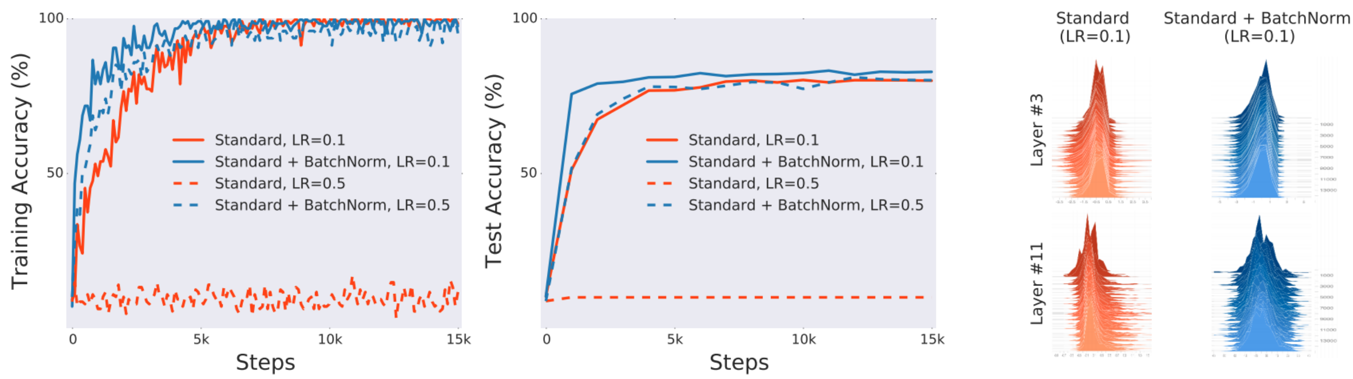


图 1.4: 左 1，左 2 分别对比了是否加 BN 对训练和测试的准确度的影响，左 3 是加了 BN 之后对数据分布的影响

注意，每一次的 BN 都需要计算每一个 mini-batch 中的均值和标准差，对于动态的网络和 RNN 来说，每一个实例的长度不一，这就为计算 BN 所需要的 μ, σ 带来了麻烦，以及，如果每一个 mini-batch 很小，都不建议使用 BN，因而实际训练使用 BN 时需要将 Batch Size 调大使用。因为 BN 实际运用中需要保留两个统计量，导致它也比较耗显存。BN 的作者[4] 同时论证了加了 BN 可以将 Dropout 去掉效果会更好，因为 BN 可以提供一定的

正则化效果。

1.4 Layer Normalization

类比 BN，Layer Normalization 其实是在 channel 方向上操作，继续 BN 的例子，LN 是在每一个样本内部做标准化，换言之，对 $C*H*W$ 值进行取平均和标准差。因为 LN 是在单独一个样本内进行，因而不需要担心 batch size 较小的情况[5]，同时 LN 在 RNN 中也获得不错的效果，近些年大火的 transformer 也采用了 LN。另外，因为 LN 是针对单个样本的，每一次只需要对一个样本进行处理即可，不需要保存统计好的结果，相比于 BN 来说，省了一部分显存。BN 中的 γ, β 是可以通过学习得到的，LN 是固定的，那么，如果一个样本中不同的特征之间相差过大，通过 LN 有一定可能降低模型的表达能力（如性别和年龄）。

$$\mu = \sum_i x_i \tag{1.4}$$

$$\sigma = \sqrt{\sum_i (x_i - \mu)^2 + \epsilon} \tag{1.5}$$

1.5 Theoretical Analysis

这里以 Normalization 中的反向传播来说明为什么 Normalization 会有效，如公式（1.6，1.7）所见，普通的矩阵点乘，这是未加 Normalization 的表示，随着乘上的权重矩阵越来越多，会导致梯度弥散和梯度爆炸发生。

$$H_l = W_l^T H_{l-1} \tag{1.6}$$

$$\frac{\partial H_l}{\partial H_{l-1}} = W_l \tag{1.7}$$

$$\frac{\partial H_l}{\partial H_k} = \prod_{i=k+1}^{l} W_i \tag{1.8}$$

对比一下加了 Normalization 的情况，可以发现在前面会多了 $\frac{g_i}{\sigma_i}$，其中 g_i 是可以被网络学到的，当整体梯度值变大的时候，g_i 就会相对变小，变小的情况也如此。通过一个可学习的参数从而一定程度上缓解两个梯度问题是个不错的角度。在 LSTM 中，也是通过可学习的机制使得一条通道上最终出来的梯度值靠近 1。联系 Residual Net 是通过在后面加 1 来减少这两个问题的频率。其实可以从这三类方法获得一定的灵感，设计网络的时候可以加上两个参数，一个是乘上原来的，一个是加上一个参数，让网络去学习，说不定是一个不错的尝试。

$$H_l = Norm(W_l^T H_{l-1}) = \frac{g_l}{\sigma_l}(W_l^T H_{l-1} - \mu_l) + \beta \tag{1.9}$$

$$\frac{\partial H_l}{\partial H_{l-1}} = \frac{g_l}{\sigma_l} W_l \tag{1.10}$$

$$\frac{\partial H_l}{\partial H_k} = \prod_{i=k+1}^{l} \frac{g_i}{\sigma_i} W_i \tag{1.11}$$

参考文献

[1] Juliuszh. https://zhuanlan.zhihu.com/p/33173246.

[2] SANTURKAR S, TSIPRAS D, ILYAS A, et al. How does batch normalization help optimization?[J]. ArXiv preprint arXiv:1805.11604, 2018.

[3] LI H, XU Z, TAYLOR G, et al. Visualizing the loss landscape of neural nets[J]. ArXiv preprint arXiv:1712.09913, 2017.

[4] IOFFE S, SZEGEDY C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[C]//International conference on machine learning. [S.l. : s.n.], 2015: 448-456.

[5] BA J L, KIROS J R, HINTON G E. Layer normalization[J]. ArXiv preprint arXiv:1607.06450, 2016.

第 2 章 备用

2.1 过早饱和

尽管底层输入的数据会不断发生变换（相当于自变量，分布没有变），可是经过激活函数之后发现到达高层的值几乎没有发生变换，换言之，底层数据变换的讯号无法传播到高层导致训练过早饱和，激活函数是为了使得模型获得非线性的表达能力，可是会一定程度上导致梯度消失，梯度爆炸和训练过早饱和等问题，有没有不用激活函数的更好选择呢?