理解深度学习

许志钦

2023 年 9 月 13 日

目录

Chapter 1

多尺度神经网络

1.1 引言

根据前面提到的频率原则，我们认识到神经网络在训练中低频收敛的很快，而高频往往会收敛的比较慢。在实际应用中，当数据中有高频成分时，一般结构的神经网络会变得较难训练。但在很多科学问题或者工程应用中，多尺度现象是常见的，比如求解微分方程和拟合图像。因此，当数据中存在多种频率时，我们有必要设计一些合适的结构来提升神经网络的收敛速度。

既然神经网络在低频收敛快，高频收敛慢，那为了提升神经网络训练时的收敛速度，一种很自然的想法就是把高频数据转换成低频数据，加快训练速度。那怎样转换呢？观察图1.1中的两张子图，左边明显比右边震荡剧烈。那我们怎样能把左边变成右边呢？只需要对坐标做一个简单的拉伸即可。但看到这儿肯定有人要说，这不具有通用性，一般问题中我不知道原来数据的频率怎么办呢，这个问题问的非常好，既然我们不知道频率，那我们就提供多种通道，让神经网络自己选。具体网络结构的设计我们可以看下一节。

图 1.1: 通过拉伸将高频变为低频.

1.2 Mscale DNN

1.2.1 频率拉伸网络的设计想法

在这一部分，我们先来介绍一个直观的想法，为了解决高频收敛慢的问题，同时根据神经网络低频训练快的性质，我们从频率角度考虑做一个拉伸来将高频数据的学习转化成低频数据的学习。同时，对于这样一个直观想法，我们也在后面将讨论其具体算法实现过程中可能会遇到的问题。

考虑一个有限带宽 (band-limited) 的函数 $f(\mathbf{r}), \mathbf{r} \in \mathbb{R}^d$, 其傅里叶变换的形式 $\widehat{f}(\mathbf{k})(k \in \mathbb{R}^d)$ 仅有在频率满足 $\|\boldsymbol{k}\|_2 \leq K_{\max}$ 时不为零，在数学上，我们也称其为具有紧支撑。一个函数 $\widehat{f}(\mathbf{k})$ 的支撑集记为 $\mathrm{supp}\widehat{f}(\mathbf{k})$，指的是函数不为零的自变量的集合。假设我们关心的函数在频率空间的支撑集在一个球 $B(K_{\max})$ 内，

$$\mathrm{supp}\widehat{f}(\mathbf{k}) \subset B(K_{\max}) = \{\mathbf{k} \in \mathbb{R}^d, \|\boldsymbol{k}\|_2 \leq K_{\max}\}. \tag{1.1}$$

我们首先将区域 $B(K_{\max})$ 划分为具有均匀或不均匀宽度的 M 个同心环形的并集，比如

$$A_i = \{\mathbf{k} \in R^d, (i-1)K_0 \leq \|\boldsymbol{k}\|_2 \leq iK_0\}, K_0 = K_{\max}/M, 1 \leq i \leq M \tag{1.2}$$

从而

$$B(K_{\max}) = \bigcup_{i=1}^{M} A_i. \tag{1.3}$$

有了这个划分，我们可以将函数 $\widehat{f}(\mathbf{k})$ 做如下划分，

$$\widehat{f}(\mathbf{k}) = \sum_{i=1}^{M} \chi_{A_i}(\mathbf{k})\widehat{f}(\mathbf{k}) \triangleq \sum_{i=1}^{M} \widehat{f}_i(\mathbf{k}), \tag{1.4}$$

这里有

$$\chi_{A_i}(\mathbf{k}) = \begin{cases} 1, & \boldsymbol{k} \in A_i \\ 0, & \boldsymbol{k} \notin A_i \end{cases} \tag{1.5}$$

$$\mathrm{sup}\, \widehat{f}_i(\mathbf{k}) \subset A_i. \tag{1.6}$$

我们可以应用一个简单的缩放，将高频区域 A_i 转换为低频区域。也就是说，我们定义一个等比例版本的 $\widehat{f}_i(\mathbf{k})$ 如下，

$$\widehat{f}_i^{(\mathrm{scale})}(\mathbf{k}) = \widehat{f}_i(\alpha_i \mathbf{k}), \alpha_i > 1, \tag{1.7}$$

其实域空间的函数对应有

$$f_i(\mathbf{r}) = f_i^{(\mathrm{scale})}(\alpha_i \mathbf{r}), \tag{1.8}$$

注意到如果 α_i 选择的足够大的话, 等比例函数的频谱是低频的, 即,

$$\mathrm{supp}\widehat{f_i^{(\mathrm{scale})}}(\mathbf{k}) \subset \{\mathbf{k} \in R^d, \frac{(i-1)K_0}{\alpha_i} \leq |\mathbf{k}| \leq \frac{iK_0}{\alpha_i}\}. \tag{1.9}$$

通过神经网络的频率原则 [2], 我们知道，当 iK_0/α_i 是小的, 我们可以快速地训练神经网络 $h_i(\mathbf{r}, \theta^{n_i})$ 来学习 $f_i^{(\mathrm{scale})}(\mathbf{r})$。

$$f_i^{(\mathrm{scale})}(\mathbf{r}) \sim h_i(\mathbf{r}, \theta^{n_i}), \tag{1.10}$$

这可以立即给出对 $f_i(\mathbf{r})$ 的近似

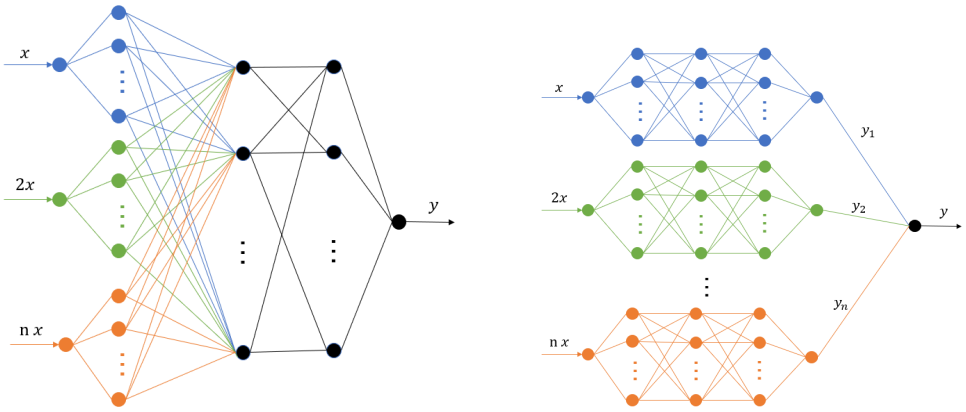
$$f_i(\mathbf{r}) \sim h_i(\alpha_i \mathbf{r}, \theta^{n_i}) \tag{1.11}$$

同样地，也可以立即得到对 $f(\mathbf{r})$ 的近似

$$f(\mathbf{r}) \sim \sum_{i=1}^{M} h_i(\alpha_i \mathbf{r}, \theta^{n_i}). \tag{1.12}$$

尽管上面这一过程看起来很直观，但在处理高维问题时会遇到困难。在高维中逼近函数时，上述过程的困难在于需要计算傅里叶空间的乘积（或者实域空间的卷积），这里有维数诅咒的问题。

1.2.2 MscaleDNN 网络结构



(a) MscaleDNN-1　　(b) MscaleDNN-2

图 1.2: Illustration of two MscaleDNN structures. Reprinted from [1].

即使我们在上一节中提出的导出对 $f(\mathbf{r})$ 的近似 (1.12) 的这一过程，在高维情形下是不切实际的。然而，它确实提出了一种看起来有道理的的函数空间形式，用于使 DNN 函数更快地找到解。我们可以使用一系列 α_i，从 1 到一个大的数字来生成 MscaleDNN，这可以实现我们更快拟合具有多个频率的函数的目标。

为了实现 MscaleDNN，我们将第一个隐含层的神经元分解为 A 个部分。第 i 部分的神经元接受的输入是 $i\boldsymbol{x}$, 也就是说它对应的输出是 $\sigma(i\boldsymbol{w}\cdot\boldsymbol{x}+b)$, 这里 $\boldsymbol{w}$, $\boldsymbol{x}$, b 分别对应着权重，输入和偏差项。完整的 MscaleDNNs 是如下定义的，

$$h(\boldsymbol{x}) = \boldsymbol{W}_L\sigma\circ(\boldsymbol{W}_{L-1}\sigma\circ(\cdots\sigma\circ(\boldsymbol{W}_1\sigma\circ(\boldsymbol{K}\odot\boldsymbol{W}_0\boldsymbol{x}+\boldsymbol{b}_0)+\boldsymbol{b}_1)\cdots)+\boldsymbol{b}_{L-1}) \tag{1.13}$$

这里 $\boldsymbol{x}\in\mathbb{R}^d$, $\boldsymbol{W}_l\in\mathbb{R}^{n_{l+1}\times n_l}$, n_l 是第 l 个隐藏层的神经元数目, $n_0=d$, $b_l\in\mathbb{R}^{l+1}$, σ 是一个标量函数，“$\circ$” 代表着 entry-wise 操作, $\odot$ 是 Hadamard 乘积，

$$\boldsymbol{K} = (\underbrace{1,1,\cdots,1}_{\text{1st part}},\underbrace{1,1,\cdots,1}_{\text{2nd part}},2,\cdots,i-1,\underbrace{i,i,\cdots,i}_{\text{(i+1)th part}},i+1,\cdots,\underbrace{A-1,A-1,\cdots,A-1}_{\text{Ath part}})^T. \tag{1.14}$$

如果一个神经网络的结构遵循 Eq. (1.13)，我们称这样的结构是多尺度神经网络（Multi-scale DNN (MscaleDNN)）。图1.2展示了两种常见的多尺度网络结构。第一种结构只是对正常的全连接网络的第一个隐藏层的每一个神经元加一个系数。第二种结构由不同尺度的小网络组成，由于子网络之间没有相互连接的连接，因此，参数数目远比第一种网络要少。

1.2.3 激活函数的选择

第一个隐藏层的激活函数一般会特殊设定，并且在实验上发现其对实际效果有显著影响。最开始的时候，我们考虑第一个隐藏层的激活函数是用小波函数的母函数，模仿小波能够抓住各种频率的想法。后来发现，用正余弦函数效果会更好。其它层的激活函数不是很敏感，可以用常用的函数。第一个隐藏层的激活函数还可以有特殊的含义。比如用正余弦函数，可以比作是做傅里叶展开，而关键要学的是系数。一般而言，系数对输入的依赖会比较简单。

【这一块可以详细展开，包括 fourier feature let, nerf, 从刘子旗到李西安的尝试，也可以说一下 mscalednn 的一些不足，包括泛化性】

1.3 应用

需要多跑一些例子，然后讨论一下，当尺度太大时，泛化会不好。另外，如果所有尺度都一样，且都很大，学习仍然会很慢，因为虽然频率都小了，但存在不同频率之间的竞争（只是一种直观理解）。

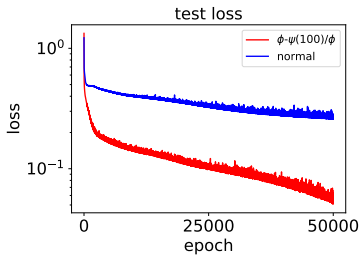


图 1.3: Test loss function vs. training epoch. Fitting high-frequency 1d function. We use a network 1-3000-100-100-100-1. The learning rate is 10^{-4} with a decay rate 10^{-7} for each training step with batch size 30000. The training and test dataset are 10000 and 2000 random samples, respectively.

1.3.1 拟合高频函数

在物理问题/实际问题中，近似高频函数是很重要的。为了说明 MscaleDNN 的效果，我们使用神经网络来拟合一维问题中的高频函数，损失函数用的是常用的均方误差函数。

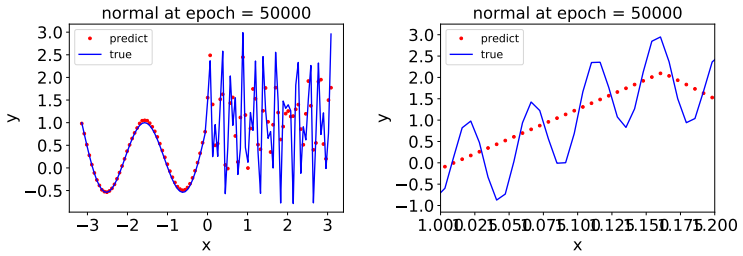
一维问题. 考虑

$$f(x) = \begin{cases} \sin(x) + \sin(3x) + 1 & x \in [-\pi, 0] \\ \sin(23x) + \sin(137x) + \sin(203x) + 1 & x \in [0, \pi] \end{cases}$$

结果如图. 1.3所示, 使用单一尺度的神经网络（即普通/传统的神经网络）拟合目标函数 $f(x)$, 在训练过程中损失函数下降的非常慢，而使用多尺度神经网络的损失函数下降很快。另一方面，为了更直观的对比两种网络结构学到的曲线结果，我们将学到的曲线在测试数据的函数值标记出来，见图 1.4。对于单一尺度的网络结构 (见图. 1.4第一行), 神经网络学到了一个低频函数，这个函数不能捕获/抓住目标函数的特征。与此相反，多尺度网络 (见图. 1.4第二行) 准确地抓住了目标函数的高度震荡的特征。

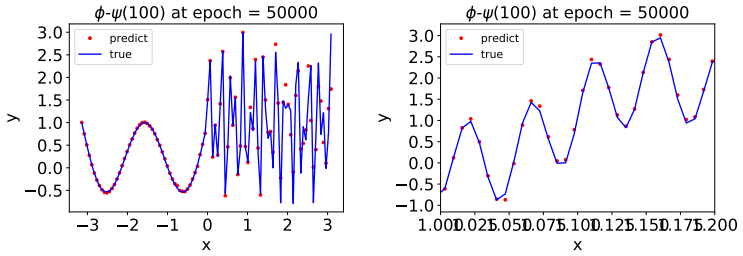
[TBA? 二维需要加吗]

除了拟合问题之外，多尺度神经网络还可以帮助加快求解传统数值问题，比如求解高维偏微分函数。更多关于解偏微分方程的讨论将在下一章展开。



(a) full curve

(b) zoom in



(c) full curve

(d) zoom in

图 1.4: The learning curves in fitting high-frequency 1d function on test data points.

参考文献

[1] Ziqi Liu, Wei Cai, and Zhi-Qin John Xu. Multi-scale deep neural network (mscalednn) for solving poisson-boltzmann equation in complex domains. *Communications in Computational Physics*, 28(5):1970–2001, 2020.

[2] Zhi-Qin John Xu, Yaoyu Zhang, Tao Luo, Yanyang Xiao, and Zheng Ma. Frequency principle: Fourier analysis sheds light on deep neural networks. *arXiv preprint arXiv:1901.06523*, 2019.