理解深度学习

许志钦

2023 年 9 月 13 日

目录

Chapter 1

线性区域的隐式偏好

1.1 NTK 动力学

考虑一个宽度为 m 的神经网络 $f(\cdot,\boldsymbol{\theta})$，训练数据集 $(\boldsymbol{x}_i,y_i)_{i=1}^n$，其中，$f$ 的参数为 $\boldsymbol{\theta}$。以 L_S 为损失函数的梯度流动力学为

$$\begin{cases} \dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}} L_S(\boldsymbol{\theta}), \\ \boldsymbol{\theta}(0) = \boldsymbol{\theta}_0, \end{cases} \tag{1.1}$$

其中

$$L_S(\boldsymbol{\theta}) = \frac{1}{2}\sum_{i=1}^{n}(f(\boldsymbol{x}_i,\boldsymbol{\theta}) - y_i)^2. \tag{1.2}$$

那么输出函数 $f(\cdot,\boldsymbol{\theta})$ 的训练动力学为

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} f(\boldsymbol{x},\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x},\boldsymbol{\theta}) \cdot \dot{\boldsymbol{\theta}} \\ &= -\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x},\boldsymbol{\theta}) \cdot \nabla_{\boldsymbol{\theta}} L_S(\boldsymbol{\theta}) \\ &= -\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x},\boldsymbol{\theta}) \cdot \sum_{i=1}^{n} \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_i,\boldsymbol{\theta})(f(\boldsymbol{x}_i,\boldsymbol{\theta}) - y_i) \\ &= -\sum_{i=1}^{n} K_m(\boldsymbol{x},\boldsymbol{x}_i)(f(\boldsymbol{x}_i,\boldsymbol{\theta}) - y_i), \end{aligned}$$

其中对时间 t 时刻，在定义在 $\Omega\times\Omega$ 上的神经正切核 (Neural Tangent Kernel,NTK)

$$K_m(\boldsymbol{x},\boldsymbol{x}')(t) := \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x},\boldsymbol{\theta}(t)) \cdot \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}',\boldsymbol{\theta}(t)). \tag{1.3}$$

K_m 也称为 Gram 矩阵。考虑宽度趋于无穷时，我们记

$$K^*(\boldsymbol{x},\boldsymbol{x}')(t)=\lim_{m\to\infty}K_m(\boldsymbol{x},\boldsymbol{x}')(t):=\lim_{m\to\infty}\nabla_{\boldsymbol{\theta}}f(\boldsymbol{X},\boldsymbol{\theta}(t))(\nabla_{\boldsymbol{\theta}}f(\boldsymbol{X},\boldsymbol{\theta}(t)))^T\in\mathbb{R}^{n\times n}. \tag{1.4}$$

显然, K^* 是一个非负定的矩阵。简单的证明如下。做 SVD 分解，可得 $\nabla_{\boldsymbol{\theta}}f(\boldsymbol{X},\boldsymbol{\theta})=U\Sigma V^T$，因此，$K^*=U\Sigma^2U^T\geq 0$。

模型的梯度下降流变成了

$$\frac{\mathrm{d}}{\mathrm{d}t}\Big(f(\boldsymbol{x},\boldsymbol{\theta}(t))-f(\boldsymbol{x})\Big)=-\sum_{i=1}^{n}K^*(\boldsymbol{x},\boldsymbol{x}_i)(t)\Big(f(\boldsymbol{x}_i,\boldsymbol{\theta}(t))-f(\boldsymbol{x}_i)\Big). \tag{1.5}$$

定义模型输出与真实标签的差为 $\boldsymbol{u}(\boldsymbol{x},t)=f(\boldsymbol{x},\boldsymbol{\theta}(t))-f(\boldsymbol{x})$。记 $\boldsymbol{X}\in\mathbb{R}^{n\times d}$ 与 $\boldsymbol{Y}\in\mathbb{R}^n$ 为训练数据，$u(\boldsymbol{X})\in\mathbb{R}^n,\nabla_{\boldsymbol{\theta}}f(\boldsymbol{X},\boldsymbol{\theta}(t))\in\mathbb{R}^{n\times M}$，其中 M 为模型参数的数量，那么，我们可以得到

$$\frac{\mathrm{d}u(\boldsymbol{X})}{\mathrm{d}t}=-K^*(t)u(\boldsymbol{X}). \tag{1.6}$$

在连续形式下，我们可以定义经验密度 $\rho(\boldsymbol{x})=\sum_{i=1}^{n}\delta(\boldsymbol{x}-\boldsymbol{x}_i)/n$ 然后进一步记 $u_\rho(\boldsymbol{x})=u(\boldsymbol{x})\rho(\boldsymbol{x})$。因此 u 的动力学转化为

$$\frac{\mathrm{d}}{\mathrm{d}t}u(\boldsymbol{x},t)=-\int_{\mathbb{R}^d}K^*(\boldsymbol{x},\boldsymbol{x}')(t)u_\rho(\boldsymbol{x}',t)\,\mathrm{d}\boldsymbol{x}'. \tag{1.7}$$

注意到，我们在这里稍微滥用了 K^* 的记号。

考虑一个两层全连接网络，在神经正切核（NTK）区域 (Jacot et al. 2018) 的设定下

$$f(\boldsymbol{x},\boldsymbol{\theta})=\frac{1}{\sqrt{m}}\sum_{j=1}^{m}a_j\sigma(\boldsymbol{w}_j^\intercal\boldsymbol{x}+b_j), \tag{1.8}$$

其中 $\boldsymbol{\theta}$ 是所有参数组成的向量，它具体由每个神经元 $j\in[m]$ 的三元组 $(a_j,\boldsymbol{w}_j^\intercal,b_j)^\intercal\in\mathbb{R}^{d+2}$，并且用标准正态分布初始化。当神经元数目 $m\to\infty$ 时，如(1.12)所示在初始化附近进行线性化

$$f^{\mathrm{lin}}(\boldsymbol{x};\boldsymbol{\theta}(t))=f(\boldsymbol{x};\boldsymbol{\theta}(0))+\nabla_{\boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta}(0))(\boldsymbol{\theta}(t)-\boldsymbol{\theta}(0)) \tag{1.9}$$

是是理论上和实验上对 $f(\boldsymbol{x};\boldsymbol{\theta}(t))$ 的一个有效逼近 (Jacot et al. 2018, Lee et al. 2019), 即对任意 t, $f^{\mathrm{lin}}(\boldsymbol{x};\boldsymbol{\theta}(t))\approx f(\boldsymbol{x};\boldsymbol{\theta}(t))$。请注意,$f^{\mathrm{lin}}(\boldsymbol{x};\boldsymbol{\theta}(t))$ 对 $\boldsymbol{\theta}$ 是线性的, 对 $\boldsymbol{x}$ 是非线性的, 在 $m\to\infty$ 时仍保留了 $f(\boldsymbol{x};\boldsymbol{\theta}(t))$ 的万有逼近能力。在本小节下文中, 我们不再区分 $f(\boldsymbol{x};\boldsymbol{\theta}(t))$ 和 $f^{\mathrm{lin}}(\boldsymbol{x};\boldsymbol{\theta}(t))$。

直观地，我们理解一下为什么 NTK 设定下，一阶线性近似可以成立。利用参数化在初始化时具有独立性，神经网络 (1.8) 的输出的期望 (关于初始化) 显然为 0，其方差

$$E(f(\boldsymbol{x},\boldsymbol{\theta}))^2=E\frac{1}{m}\sum_{j,k=1}^{m}a_j\sigma(\boldsymbol{w}_j^\intercal\boldsymbol{x}+b_j)a_k\sigma(\boldsymbol{w}_k^\intercal\boldsymbol{x}+b_k),$$

由于独立性，上式右端仅能取 $j=k$ 的项，

$$E(f(\boldsymbol{x},\boldsymbol{\theta}))^2=\frac{1}{m}\sum_{j=1}^{m}a_j^2(\sigma(\boldsymbol{w}_j^\intercal\boldsymbol{x}+b_j))^2=Ea_j^2(\sigma(\boldsymbol{w}_j^\intercal\boldsymbol{x}+b_j))^2.$$

因此，模型在初始化的输出是一个均值为 0，方差为 O(1) 的随机变量。目标数据的尺度一般也是 O(1)，因此要学好数据需要改变的总量也在 O(1)，每个神经元需要改变的量在 O(1/m)。由于前置因子是 $1/\sqrt{m}$，因此，每个 $a_j\sigma(\boldsymbol{w}_j^\intercal\boldsymbol{x}+b_j)$ 需要改变的量为 $1/\sqrt{m}$，随着 m 变大，这个量趋于 0。但对于平均场模型，其初始尺度更小，要改变的总量也在 O(1)，每个神经元需要改变的量在 O(1/m)。由于前置因子是 $1/m$，因此，每个 $a_j\sigma(\boldsymbol{w}_j^\intercal\boldsymbol{x}+b_j)$ 需要改变的量为 O(1)，不是一个小量，因此，具有高度非线性。

对于 NTK 模型，Gram 矩阵 $K^*(t)$ 在整个训练过程中，在宽度趋于无穷大时，由于参数改变量很小，所以它可以被初始的 $K^*(0)$ 所近似，为方便，记为 K^*。

关于 Eq. (1.6) 中的动力学的收敛性分析可以通过对 K 进行特征分解来完成。考虑特征分解

$$K=P^T\Sigma P,$$

其中 $P=(\boldsymbol{v}_1,\boldsymbol{v}_2,\cdots,\boldsymbol{v}_n)^T\in\mathbb{R}^{n\times n}$, $\Sigma\in\mathbb{R}^{n\times n}$ 是一个对角矩阵。记 λ_i 为特征方向 $\boldsymbol{v}_i$ 对应的特征值。因此，K 可以写成

$$K=\sum_{i=1}^{n}\lambda_i\boldsymbol{v}_i\boldsymbol{v}_i^T.$$

用 $\{\boldsymbol{v}_i\}$ 构成的基来表示 $u(\boldsymbol{X})$,

$$u(\boldsymbol{X})=\sum_{i=1}^{n}\alpha_i\boldsymbol{v}_i.$$

注意，α_i 在训练过程中是随训练变化的。我们把上式代入 Eq. (1.6)，

$$\sum_{i=1}^{n}\frac{\mathrm{d}\alpha_i}{\mathrm{d}t}\boldsymbol{v}_i=-\sum_{i=1}^{n}\alpha_iK\boldsymbol{v}_i=-\sum_{i=1}^{n}\alpha_i\lambda_i\boldsymbol{v}_i. \tag{1.10}$$

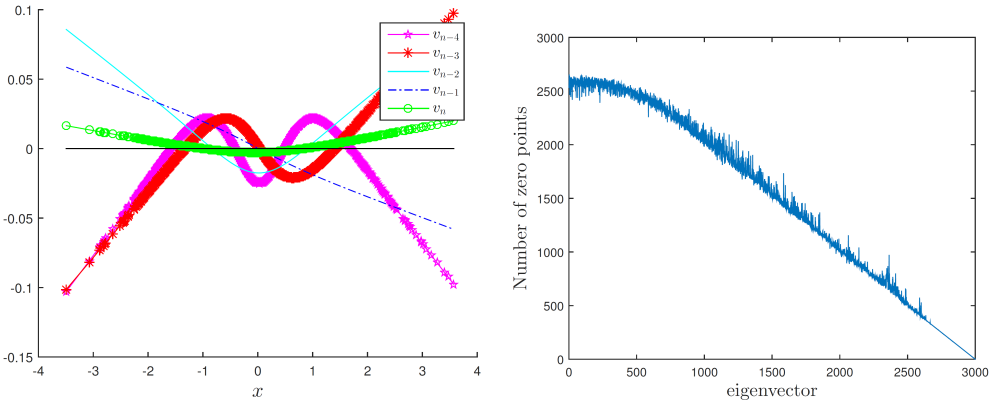
对每个分量可以得到

$$\frac{\mathrm{d}\alpha_i}{\mathrm{d}t}=-\alpha_i\lambda_i. \tag{1.11}$$

这个方程有显示的解 $\alpha_i(t)=\exp(-\lambda_it)$。这些分析可以提供大量的信息。一，在 NTK 设定下，误差是指数衰减的，收敛的快慢最终被最小特征值的大小所控制。一旦假设最小特征值是一个大于某个正数的情况，那误差就显然是指数收敛到全局最小值。有一系列的工作把这些分析做了严格化。二，特征向量的信息提供了哪些信息可以快速收敛或者很慢收敛。任意一个特征向量 $\boldsymbol{v}_i\in\mathbb{R}^n$，数值上，它是一个函数的离散采样，也就是 $\{(\boldsymbol{x}_j,\boldsymbol{v}_{i,j})\}_{j=1}^n$。我们可以把这些点画出来，就可以看到它的形状。一般情况，大特征值对应的特征方向的频率会比较低，这也

说明了低频为什么会先收敛。对于激活函数是 ReLU 的情况，K 的特征值和特征向量可以算出来，也有一系列的工作做了相应的研究，验证了频率原则。这里我们数值上展示一个 ReLU 的例子，看一下不同特征向量的振荡情况。

数值上，关于两层 ReLU 神经网络，在 NTK 设定下，一维输入，样本数 3000，计算其 Gram 矩阵的特征向量。如图 (1.1a) 所示，比如，最大特征值对应的特征方向为绿色圆圈的曲线，随着特征值变小，红色星星曲线明显更加振荡。为进一步定量刻画特征方向的振荡，我们数其穿过零点的次数，如图 (1.1b) 所示，我们设定特征值随着指标 (横坐标) 变大而变大，可以看出特征值越大，其穿过零点的次数越少，也可以理解为振荡越少，频率越低。这也符合我们前面谈到频率原则，即低频先收敛。



(a) (b)

图 1.1: 使用均方损失和梯度下降来找到具有 11 阶和 12 阶等间距点的多项式插值的解。

线性区域的神经网络模型是能够线性收敛到全局最小值的。这种听起来非常好的性质实际上是以牺牲模型的泛化性为代价。在这一章节中，我们将以 NTK 的设定来讨论线性区域神经网络的隐式偏好。

1.2 泛化

首先，我们通过一个一维的例子 Xu et al. (2019)，来看一下不同初始化对训练结果的影响。如图1.2所示，第一行的神经网络用比较大的初始化来学习训练点，在训练集上，神经网络可以拟合得很不错，但在测试集上，它确有明显地振荡。第二行的网络用的初始化比较小，它在训练集和测试集都能有好的拟合效果。为什么大初始化会这样呢？很容易关注到，如图第一列所示的，对于大初始化，神经网络的初始输出是比较混乱的，看起来这种混乱是会保留到训练最后。从前一章的相图分析，我们可以知道，当初始化越大时，神经网络就会越趋于线性化

区域。下面，我们来通过 NTK 设定来理解为什么初始的输出会保留到最后，以及如何克服它。

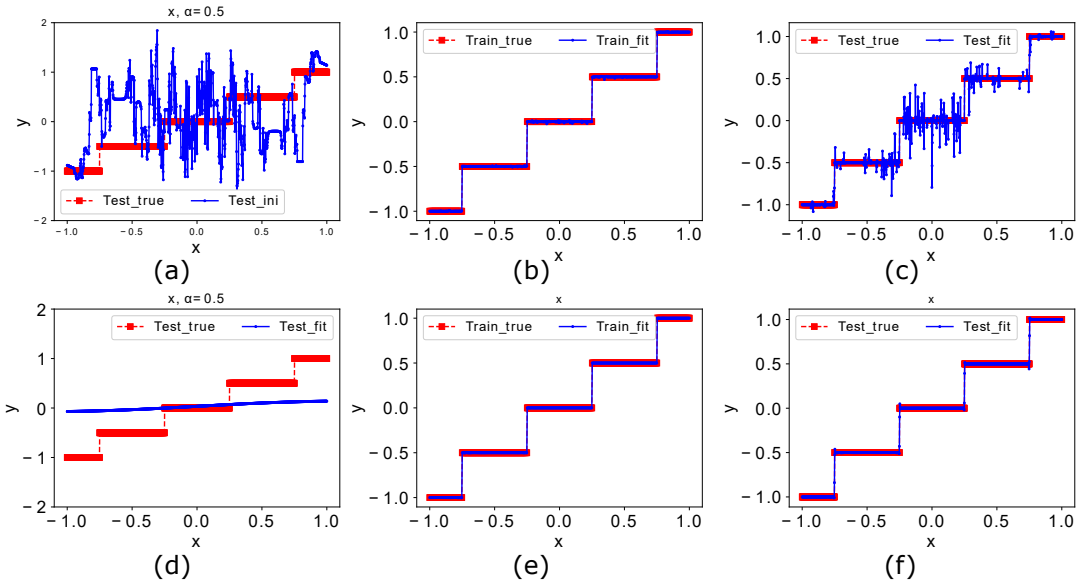


图 1.2: 使用不同初始化的 DNN 拟合式红色点。训练集和测试集在 $[-1,1]$ 上均匀采样, 样本量分别为 600 和 1200。DNN 的参数由均值为 0, 标准差为 10(第一行) 或 0.1(第二行) 的高正态分布初始化。(a,d): 测试集上的 $f(x)$(红色虚线) 和 DNN 的初始输出 (蓝色实线)。(b,e): 训练集上的 $f(x)$(红色虚线) 和训练结束时 DNN 的输出 (蓝色实线)。(c,f): 测试集上的 $f(x)$(红色虚线) 和训练结束时 DNN 的输出 (蓝色实线)。

1.3 NTK 动力学偏向距离初始最近的解

考虑一个两层全连接网络 (1.8), 当网络宽度 $m \to \infty$，如下在初始化附近的线性化展开

$$f\left(\boldsymbol{x};\boldsymbol{\theta}(t)\right) = f\left(\boldsymbol{x};\boldsymbol{\theta}(0)\right) + \nabla_{\boldsymbol{\theta}} f\left(\boldsymbol{x};\boldsymbol{\theta}(0)\right)\left(\boldsymbol{\theta}(t) - \boldsymbol{\theta}(0)\right) \tag{1.12}$$

是对于 $f\left(\boldsymbol{x};\boldsymbol{\theta}(t)\right)$ 的有效估计。

我们考虑任意损失函数 $R_S(\boldsymbol{\theta}) = \mathrm{dist}\left(\boldsymbol{f}(\boldsymbol{X},\boldsymbol{\theta}),\boldsymbol{Y}\right)$ 下的梯度流, 其中 dist 是连续可导的, 并对任意 $\boldsymbol{z} \in \mathbb{R}^n$ 满足:

1. $\mathrm{dist}(\boldsymbol{z},\boldsymbol{z}) = 0$;

2. 当且仅当 $\boldsymbol{z}' = \boldsymbol{z}$ 时,$\mathrm{dist}(\boldsymbol{z}', \boldsymbol{z})$ 取得最小值。

3. 当且仅当 $\nabla_{\boldsymbol{z}'}\mathrm{dist}(\boldsymbol{z}', \boldsymbol{z}) = 0$ 时,$\boldsymbol{z}' = \boldsymbol{z}$。

根据Zhang et al. (2020) 中的定理, 动力学(1.1)的长时间解 $\boldsymbol{\theta}(\infty) = \lim_{t\to\infty}\boldsymbol{\theta}(t)$ 与优化问题

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{\theta} - \boldsymbol{\theta}_0\|_2, \quad s.t., \quad \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\theta}) = \boldsymbol{Y} \tag{1.13}$$

的解等价，其中, $\boldsymbol{X}$: 训练集输入 $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n)^\intercal \in \mathbb{R}^{n\times d}$; $\boldsymbol{Y}$: 训练集输出 $(y_1, \cdots, y_n)^\intercal \in \mathbb{R}^n$; $\boldsymbol{g}(\boldsymbol{X})$: 对任意定义在 Ω 上的函数 g, 有 $(g(\boldsymbol{x}_1), \cdots, g(\boldsymbol{x}_n))^\intercal$。

这个结论体现在线性区域，神经网络在梯度流训练下，具有倾向于找到离初始值最近的解的隐式正则化。

1.4 非零初始输出的影响

为方便起见，我们用 h 表示神经网络 $f(\cdot, \boldsymbol{\theta})$。在下文中, 我们表示 NTK 核函数 k, 初始模型 $h_{\mathrm{ini}}(\boldsymbol{x}) = h(\boldsymbol{x}, \boldsymbol{\theta}_0)$, 数据集 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 的问题(1.13)的解为 $h_k(\boldsymbol{x}; h_{\mathrm{ini}}, \boldsymbol{X}, \boldsymbol{Y})$。

由于线性的可叠加性,对于一个处于 NTK 设定的神经网络,对任意初始函数 h_{ini}, $h_k(\cdot; h_{\mathrm{ini}}, \boldsymbol{X}, \boldsymbol{Y})$ 能够被分解为

$$h_k(\cdot; h_{\mathrm{ini}}, \boldsymbol{X}, \boldsymbol{Y}) = h_k(\cdot; 0, \boldsymbol{X}, \boldsymbol{Y}) + h_{\mathrm{ini}} - h_k(\cdot; 0, \boldsymbol{X}, h_{\mathrm{ini}}(\boldsymbol{X})). \tag{1.14}$$

该结论定量的揭示了非零初始化, 即 $h_{\mathrm{ini}} \neq 0$, 对 NTK 区域下训练充分的 DNN 输出函数的影响。在训练开始时，DNN 输出为 h_{ini}，而在训练结束时, 该初始值变为 $h_{\mathrm{ini}} - h_k(\cdot; 0, \boldsymbol{X}, h_{\mathrm{ini}}(\boldsymbol{X}))$ （后者为用神经网络拟合 h_{ini}，采样是在 $\boldsymbol{X}$ 中获得）。由于初始化输出的任意性，显然，它会使模型的泛化性变差。

1.5 反对称初始化技巧（ASI）

一般来说, 从贝叶斯推断的角度看, 对固定的 NTK 核函数 k, 一个随机的 h_{ini} 引入了一个与目标函数无关的先验分布，通常会降低推断的准确性。特别地，NTK 初始化中的 $1/\sqrt{m_l}$ 尺度导致随机初始函数 $h_{\mathrm{ini}} \sim O(1)$，使得它在分析中不能被忽略。为了消除非零的初始输出带来的负面影响, 一个简单的方法是将初始参数设定足够小。在实践中, 过小的初始参数也会导致 DNN 的初始核函数值趋于零, 这会减缓训练。基于我们上述的理论结果, 我们设计了一个反对称初始化 (antiSymmetrical initialization trick) 技巧, 它可以将初始输出固定为零, 同时保持核函数不变 (图 (1.3))。

假设激活函数 σ 是 $C^1(\mathbb{R})$ 的。记 $h_i^{[l]}$ 为具有 L 层的 DNN 的第 l 层的第 i 个节点的输出。则对于 $i = 1, \cdots, m_l$, 有 $h_i^{[l]}(\boldsymbol{x}) = \sigma_i^{[l]}(\boldsymbol{W}_i^{[l]} \cdot \boldsymbol{h}^{[l-1]}(\boldsymbol{x}) + b_i^{[l]})$。对于 DNN 输出层的第 i 个神经元, 有 $h_i^{[L]}(\boldsymbol{x}) = \boldsymbol{W}_i^{[L]} \cdot \boldsymbol{h}^{[L-1]} + b_i^{[L]}$。在用任意方法初始化 DNN 后, 我们得到 $h^{[L]}(\boldsymbol{x}, \boldsymbol{\theta}(0))$, 其中

$$\boldsymbol{\theta}(0) = (\boldsymbol{W}^{[L]}(0), \boldsymbol{b}^{[L]}(0), \boldsymbol{W}^{[L-1]}(0), \boldsymbol{b}^{[L-1]}(0), \cdots, \boldsymbol{b}^{[1]}(0)).$$

对于一般的损失函数，ASI 是考虑一个新的 DNN, 表达为 $h_{\text{ASI}}(\boldsymbol{x}, \boldsymbol{\Theta}(t)) = \frac{\sqrt{2}}{2} h^{[L]}(\boldsymbol{x}, \boldsymbol{\theta}(t)) - \frac{\sqrt{2}}{2} h^{[L]}(\boldsymbol{x}, \boldsymbol{\theta}'(t))$ 其中 $\boldsymbol{\Theta} = (\boldsymbol{\theta}, \boldsymbol{\theta}')$，在初始时，$\boldsymbol{\theta}'(0) = \boldsymbol{\theta}(0)$。我们可以证明这样一个定理: ASI 技巧在不改变核函数 k 的情况下，消除了非零的随机先验分布Zhang et al. (2020)。

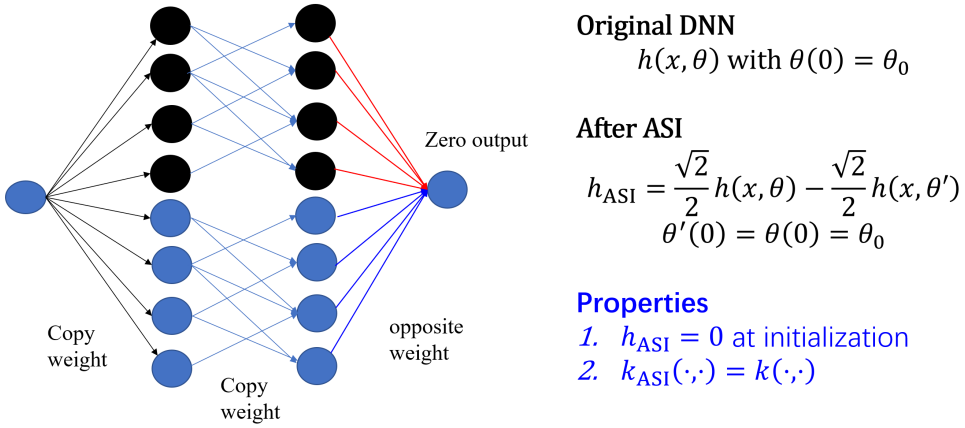


图 1.3: ASI。

注意Chizat & Bach (2018) 提出了一个 “加倍技巧” 来抵消 DNN 的初始输出, 即最后一层的神经元被复制, 新神经元具有相同的输入权重但输出权重相反。通过应用 “加倍技巧”, 得到 $h'(\cdot, 0) = 0$。然而, 第 L-1 层和第 L 层的核函数值加倍, 而第 m L-2 层的核函数值完全消失, 这可能对 DNN 的训练动力学和泛化性能产生很大影响。

1.6 实验

为了可视化方便，我们用一维数据来训练一个宽的全连接网络。如图 1.4(a) 所示，在不应用任何训练技巧的情况下，较大初始化将会以一种比较震荡的方式（蓝色的实线）学习训练数据。而 ASI 技巧（青色虚线）和加倍技巧（绿色虚线）使得全连接网络以平滑的方式学习训练数据。如图中的红色虚线曲线所示，由 Eq. (1.14) 右端计算得到的输出在测试集上准确预测了不加技巧的全连接网络的测试输出。在我们的实验中，$h_k(x; 0, \boldsymbol{X}, h_{\text{ini}}(\boldsymbol{X}))$, $h_k(x; 0, \boldsymbol{X}, \boldsymbol{Y})$,

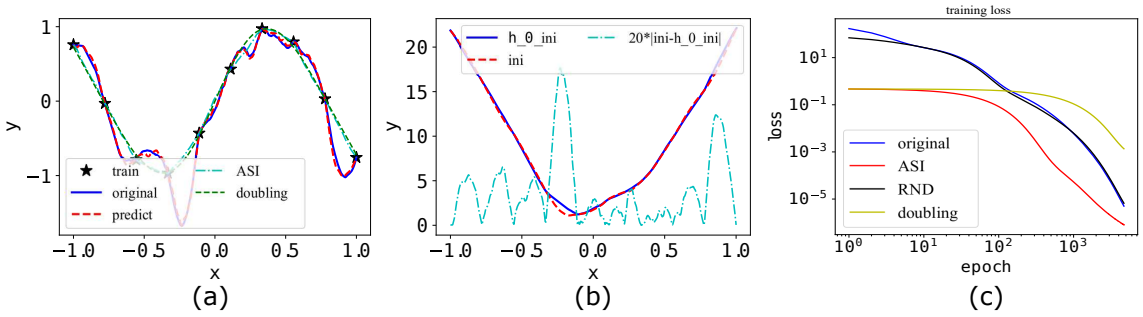


图 1.4: 合成数据。(a) 黑色星号表示训练集。其他曲线表示不同 DNN 在测试点上的最终输出。蓝色实线: 原始 DNN(无技巧); 青色虚线点: 应用 ASI 技巧的 DNN; 绿色虚线: 应用 “加倍技巧” 的 DNN; 红色虚线: 式 (1.14) 的右边项。(b) 蓝色:$h_k(x;0,\boldsymbol{X},h_{\text{ini}}(\boldsymbol{X}))$; 红色:$h_{\text{ini}}$; 青色:$20|h_{\text{ini}}-h_k(x;0,\boldsymbol{X},h_{\text{ini}}(\boldsymbol{X}))|$。(c) 不同 DNN 训练过程中的损失函数演化。蓝色: 原始 DNN; 红色: 应用 ASI 技巧的 DNN; 黑色: 应用 RND 的 DNN; 黄色: 应用 “加倍技巧” 的 DNN。原始 DNN 的宽度为 1-5000-5000-1。$v_{\text{std}}=10$。我们从 $\sin(4x)$ 在 $[-1,1]$ 区间随机采样训练集和测试集, 大小分别为 10 和 500。

$h_k(x;h_{\text{ini}},\boldsymbol{X},\boldsymbol{Y})$ 是从使用或者没有使用 ASI 技巧的非常宽的全连接网络中得到。从式 (1.14) 可见, 非零初始化为最终 DNN 输出添加了一个先验项 $h_{\text{ini}}-h_k(x;0,\boldsymbol{X},h_{\text{ini}}(\boldsymbol{X}))$。如图1.4(b) 中的青色虚线所示, 这个先验项波动很大, 因此导致训练后 DNN 的输出存在振荡。注意到这个实验同样支持频率原则（F-Principle）(Xu et al. 2019, 2020) 因为 h_{ini}（红色虚线）中的高频率部分一直保留到了训练最后的全连接网络输出中。

接下来，我们考虑神经网络的训练速度，如图1.4(c)，运用 ASI 技巧的全连接网络的损失函数比原始神经网络以及使用加倍技巧的神经网络下降的更快。为了进行比较，我们将原始神经网络的宽度加倍使得宽度与应用 ASI 技巧的网络一样宽，然后使用和原先相同的分布进行初始化。我们将这个技巧称作 RND 技巧。如图1.4(c) 中黑色曲线所示，RND 技巧下全连接网络的损失函数同样比应用 ASI 技巧的神经网络慢。

对于一维问题, 线性关系得到了很好的验证,ASI 技巧可以有效去除由 h_{ini} 引入的先验误差, 并加速训练速度。

参考文献

Chizat, L. & Bach, F. (2018), 'A note on lazy training in supervised differentiable programming', *arXiv preprint arXiv:1812.07956* .

Jacot, A., Gabriel, F. & Hongler, C. (2018), Neural tangent kernel: Convergence and generalization in neural networks, *in* 'Advances in neural information processing systems', pp. 8571–8580.

Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Sohl-Dickstein, J. & Pennington, J. (2019), 'Wide neural networks of any depth evolve as linear models under gradient descent', *arXiv preprint arXiv:1902.06720* .

Xu, Z.-Q. J., Zhang, Y., Luo, T., Xiao, Y. & Ma, Z. (2020), 'Frequency principle: Fourier analysis sheds light on deep neural networks', *Communications in Computational Physics* **28**(5), 1746–1767.

Xu, Z.-Q. J., Zhang, Y. & Xiao, Y. (2019), 'Training behavior of deep neural network in frequency domain', *International Conference on Neural Information Processing* pp. 264–274.

Zhang, Y., Xu, Z.-Q. J., Luo, T. & Ma, Z. (2020), A type of generalization error induced by initialization in deep neural networks, *in* 'Mathematical and Scientific Machine Learning', PMLR, pp. 144–164.