Project 1 of NNDL

Linyang He (15307130240)

2018 年 10 月 30 日

1 问题

理论和实验证明, 一个两层的 ReLU 网络可以模拟任何函数. 请自行定义一个函数, 并使用基于 ReLU 的神经网络来拟合此函数.

2 证明

问题中的描述并不准确. 首先, “模拟”一词含义也不清晰, 笔者默认为问题的意思是“拟合”. 此外, 一个两层的 ReLU 网络并不能“模拟”任何函数. 根据 Hornik et al., 1989, 我们注意到原文中提到：“This paper rigorously establishes that standard multilayer feedforward networks with as few as one hidden layer using using arbitrary squashing functions are capable of approximating any **Borel measurable** function from one finite dimensional space to another to any desired degree of accuracy, provided sufficiently many hidden units are available.”可见并不是所有的函数都是可以 approximate 的, 而只是对博雷尔可测（Borel measurable）函数. 于是我们将问题修改为：**一个两层的 ReLU 网络可以拟合任何博雷尔可测函数**.

根据 Universal Approximator Theorem, 我们有：

定理 $\varphi(\cdot)$ 是一个有界且连续的单调上升函数, 且 $\varphi(\cdot)$ 不恒为常数, 令 I_m 是 R^m 的一个紧子集, 那么 $\forall \varepsilon > 0, f \in C(I_m), \exists N \in N, v_i, b_i \in R, w_i \in R^m, (i = 1, 2, ..., N)$, 使得可以定义 $F(x) = \sum_{i=1}^{N} v_i \varphi(w_i^T x + b_i)$, 满足 $|F(x) - f(x)| < \varepsilon$ 对 $\forall x \in I_m$ 成立.

注意到对于单独的一层 ReLU 网络,ReLU 函数并不满足 $\varphi(\cdot)$ 是一个有界且连续的单调上升函数的条件. 于是, 对于 ReLU 网络, 我们可以构造：

$$\varphi'(x) = ReLU(x-a) - ReLU(x-b), (a < b)$$

可以发现 $\varphi'(x) = 0, \varphi'(x) = \lim_{x\to+\infty}[(x-a)-(x-b)] = b-a$. 此时, φ' 有界、连续且单调, 满足 Universal Approximator Theorem. 于是我们可以定义：$F(x) = \sum_{i=1}^{N} v_i\varphi'(w_i^T x + b_i)$, 满足 $|F(x) - f(x)| < \varepsilon$ 对 $\forall x \in I_m$ 成立. φ' 是两个 ReLU 函数的线性组合, 可以通过两层的 ReLU 网络构造, 这样就证明了**一个两层的 ReLU 网络可以拟合任何博雷尔可测函数**.

3 实现

我们先以一个简单的函数为例说明实现过程，之后我们会比较不同的函数拟合之间的比较。

3.1 函数定义

我们以一个实际的一元二次函数为例子.

$$y = 2x^2 + 3x, x \in R$$

在构建函数的时候, 我们可以用 [-10,10] 上的均匀分布的 500 个点来表示. 同时这也是我们的训练集. 对于测试集, 我们另取 [-10,10] 上随机分布的 100 个点.

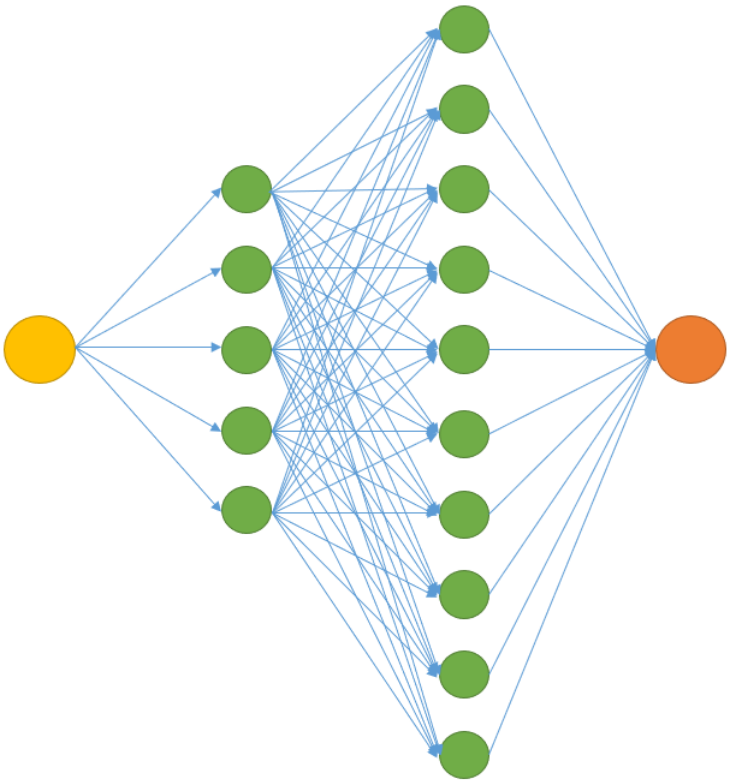
3.2 模型表述

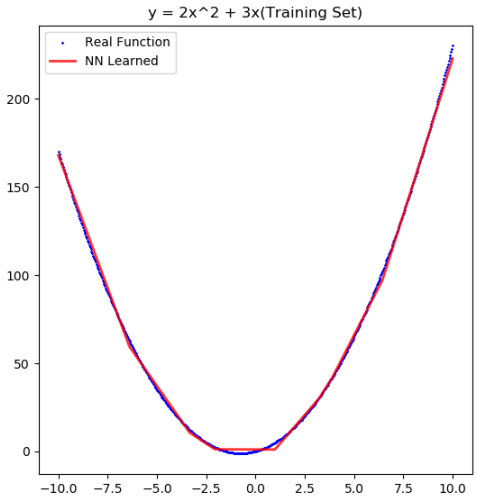
对于神经网络模型, 我们构造了两层 hidden layer 激活函数均是 ReLU 的网络. 输入层 1 个节点（x）, 第一层隐藏层 5 个节点, 第二层隐藏层 10 个节点, 输出层 1 个节点（y）. 此外, 对于网络的 optimizer, 我们选用了随机梯度下降（SGD）,learning rate 为 0.0001. 损失函数选取的是均方差 MSE. 神经网络下图所示.

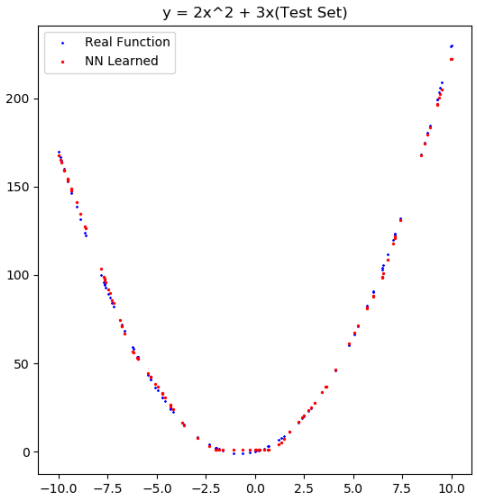
3.3 拟合效果

对于训练好的网络，我们发现，对于测试集相对误差（下文会提到）Relative error 为 9.95e-07. 而对于测试集 Relative error 是 4.79e-06. 神经

网络在测试集表现地非常好，从数据和图中都可以看出, 神经网络确实很好的拟合了 $y = 2x^2 + 3x$ 这个函数.







函数类型	函数举例	迭代次数	测试集相对误差
多项式	$y = 2x^2 + 3x,\ x \in [-10, 10]$	10356	4.79e-06
三角函数	$y = sin(x), x \in [-3, 3]$	500000	3.13e-05
指数函数	$y = e^x, x \in [-1, 6]$	30276	4.54e-06
对数函数	$y = lnx, x \in (0, 1]$	500000	3.89e-05

3.4 实验分析

3.4.1 对于实验细节的讨论

对于隐藏层结点个书的选择，我们根据经验公式：

$$h = \sqrt{m+n} + a$$

这里，h 为隐藏层节点个数，m 为输入层节点个数，n 为输出层节点个数，a 为 1-10 的常数。这样，我们的 10 和 5 的节点数是较为合理的。

在训练中，为了提升训练速度，我们希望在 loss 值变化非常小的时候（意味着几乎到达了最优点），能够提早结束训练。考虑到归一化的原因，我们定义相对误差 (Relative Error) 为：

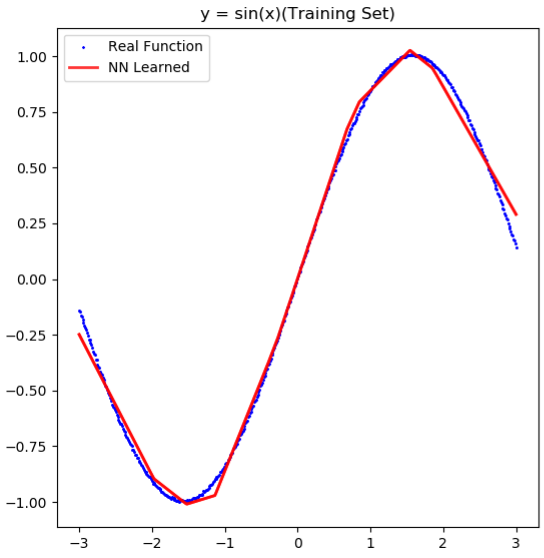
$$RelError = \frac{MSE}{||y||^2} = \frac{\frac{1}{n}\sum_{i=1}(y_i - \hat{y_i})2}{||y||^2}$$

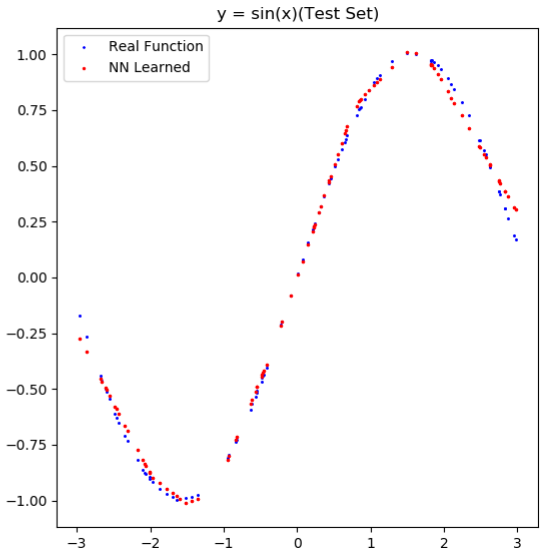
且规定当相对误差小于 1e-6 的时候，训练提早结束。实验发现，对于原本 20000 步 epochs 的训练，可以在第 10356 次迭代时后结束训练，极大地提升了神经网络的效率。

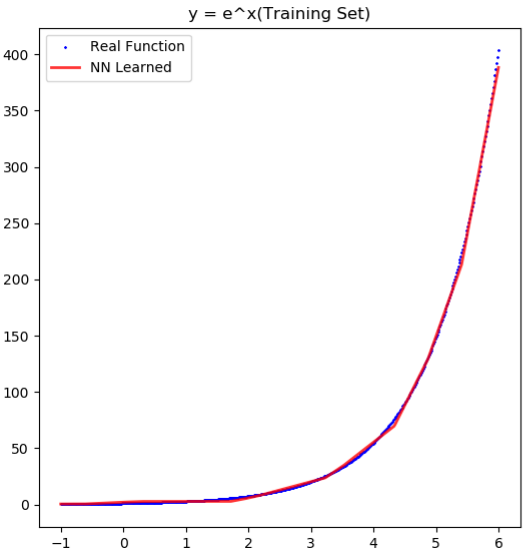
3.4.2 拟合不同类型函数之间的比较

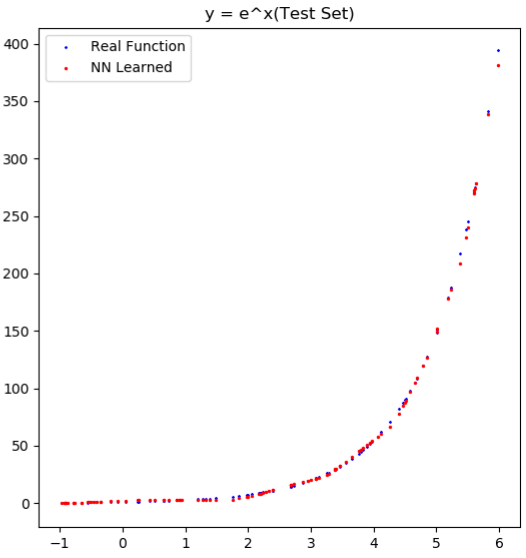
为了便于比较，我们设定训练集都是 500 个点。

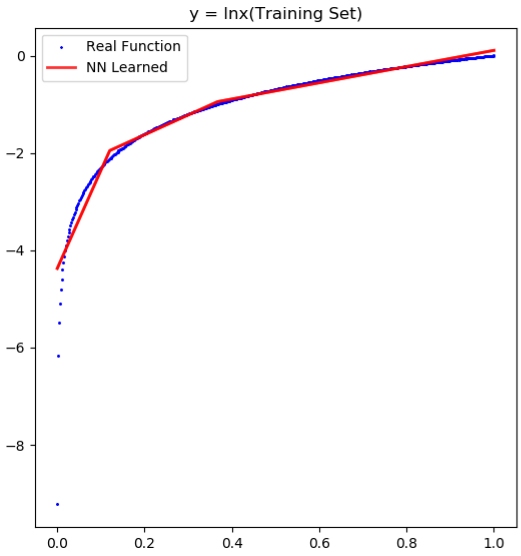
我们发现，三角函数比较难学习的，迭代次数高达 500000 后测试集相对误差仍然高。主要的原因是在所取的定义域上，三角函数形状较为复杂。而对数函数较难以拟合的部分主要是接近在 $x \to 0$ 时，$y \to -\infty$ 的时候，此时 loss 总会很大。当然，我们也可以预见，对于高次、图像复杂的多项式函数，神经网络拟合也较为困难。下图是一些神经网络拟合函数的图示。注意在 Python 代码中，默认以多项式函数为主。

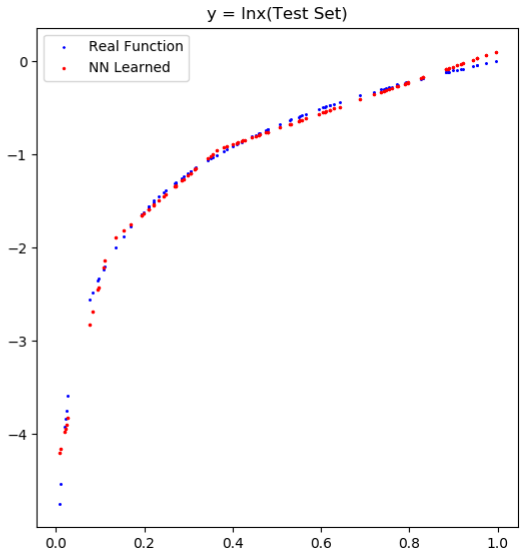

y = sin(x)(Training Set)
Real Function
NN Learned
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−3
−2
−1
0
1
2
3


y = sin(x)(Test Set)
Real Function
NN Learned
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−3
−2
−1
0
1
2
3


y = e^x(Training Set)
Real Function
NN Learned
400
350
300
250
200
150
100
50
0
−1
0
1
2
3
4
5
6


y = e^x(Test Set)
Real Function
NN Learned
400
350
300
250
200
150
100
50
0
−1
0
1
2
3
4
5
6


y = lnx(Training Set)
Real Function
NN Learned
0
−2
−4
−6
−8
0.0
0.2
0.4
0.6
0.8
1.0


y = lnx(Test Set)
Real Function
NN Learned
0
−1
−2
−3
−4
0.0
0.2
0.4
0.6
0.8
1.0