Deep Neural Networks for Solving High Dimensional PDEs

https://github.com/SyrahT/DNN_fit_high_dim_function

1 工作总结

1. 修改代码，总体非常清晰、简洁（从 430 减至 300 行），可读性强，命名规范，使用了 Pytorch，可用 GPU，评估方法改为自助法 (bootstrap)

2. 增加输出，比如 max gap of train inputs, max gap of test inputs, mean gap of train inputs, mean gap of test inputs.

3. 根据论文完成 DNN 解 Laplace 方程

4. 有一些对比和讨论，拟合的函数也较为全面

个人评分：15

2 一些结果

2.1 低频函数

$$f(x,y) = cos(4x) + cos(4y),\ (x,y) \in [-\pi/2, \pi/2]^2$$

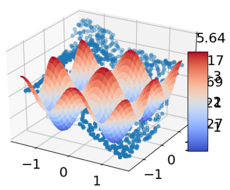

图 1: 0 epoch

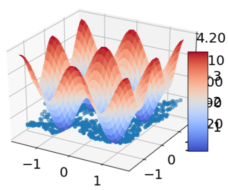

图 2: 1000 epoch

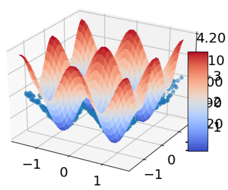

图 3: 2000 epoch

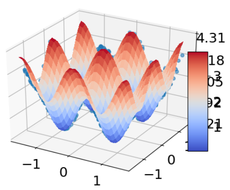

图 4: 3000 epoch

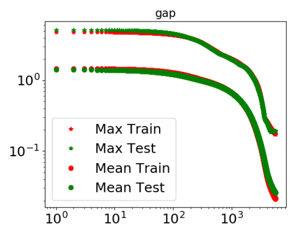

图 5: gap

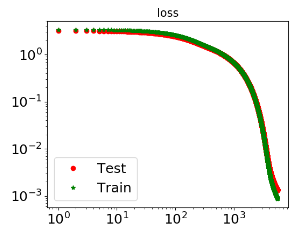

图 6: loss

在网络 2-200-200-200-1，激活函数为 $tanh$，$learning\_rate = 1e-6$，$learning\_rateDecay = 2e-7$ 条件下，用 Adam 算法优化. gap 表示与真值差的绝对值的均值或最大值. Loss 采用 MSE loss. （以下若无说明均为这里的定义）

2.2 低频高频混合

$$f(x) = \begin{cases} sin(4x), \ x \in [-\pi/2, 0) \\ sin(16x), \ x \in [0, \pi/2] \end{cases}$$

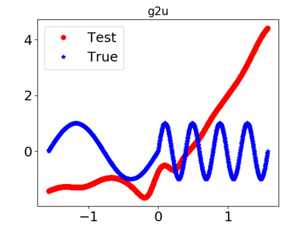

图 7: 0 epoch

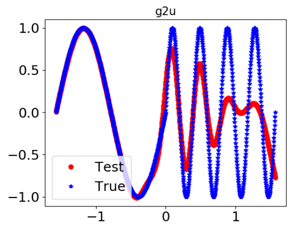

图 8: 500 epoch

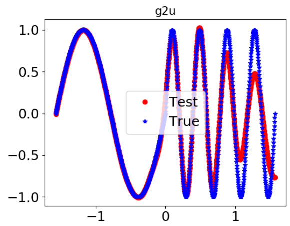

图 9: 2500 epoch

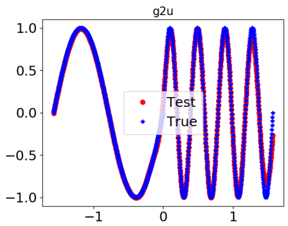

图 10: 5000 epoch

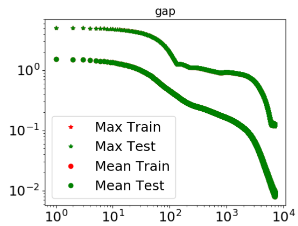

图 11: gap

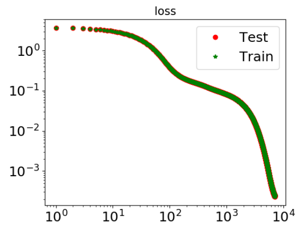

图 12: loss

$$f(x) = sin(4x) + sin(16x), \ x \in [-\pi/2, \pi/2]$$

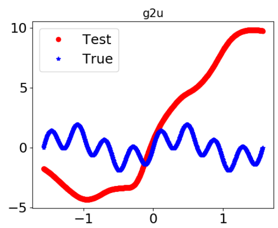

图 13: 0 epoch

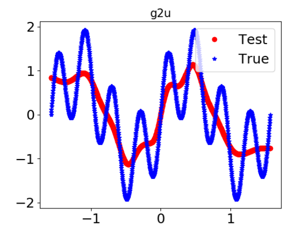

图 14: 500 epoch

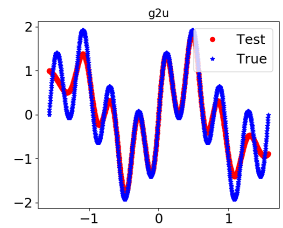

图 15: 2500 epoch

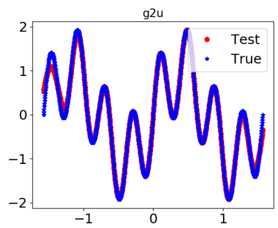

图 16: 5000 epoch

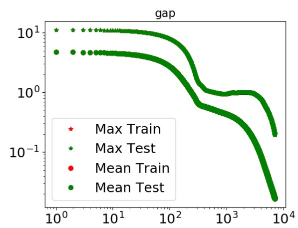

图 17: gap

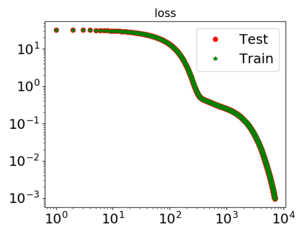

图 18: loss

在网络 1-500-500-500-1，激活函数为 $tanh$，$learning\_rate = 1e-6$，$learning\_rateDecay = 2e-7$ 条件下，用 Adam 算法优化. 可以明显观察到 F-Principle. 在 Loss 上的拐点似乎与低频拟合完毕有关.

2.3 低维到高维

考虑 $f:\mathbf{R}^3\rightarrow\mathbf{R}^{60}$

线形情况：

$$f(\mathbf{x})=(f_0(\mathbf{x}),f_1(\mathbf{x}),\cdots,f_{59}(\mathbf{x})),\ \mathbf{x}=(x_0,x_1,x_2)\in[-\pi/2,\pi/2]^3$$

$$f_i(\mathbf{x})=cos(i)*x_{[i/20]}$$

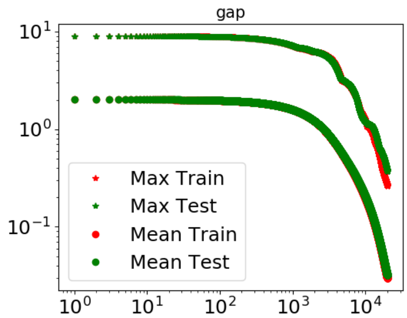

图 19: gap

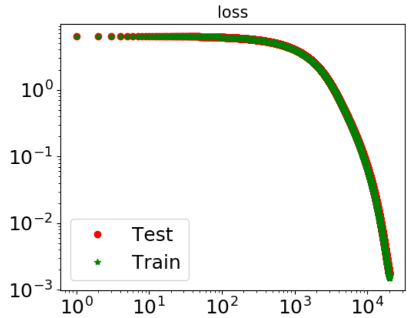

图 20: loss

非线性情况：

$$f_i(\mathbf{x})=cos(cos(i)*x_{[i/20]})$$

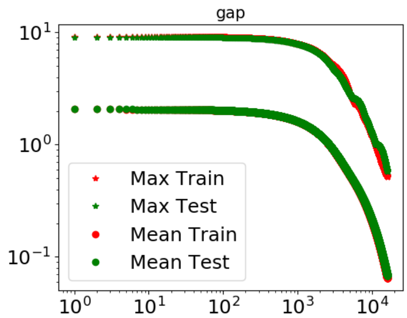

图 21: gap

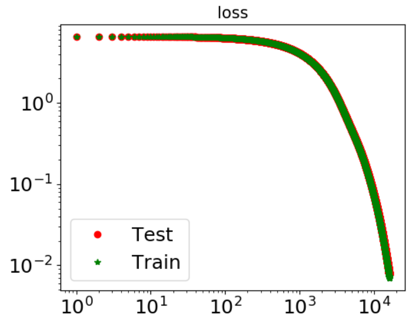

图 22: loss

在网络 3-200-200-200-60，激活函数为 $tanh$，$learning\_rate=1e-6$，$learning\_rateDecay=2e-7$ 条件下，用 Adam 算法优化. 可以看到非线性情况收敛要慢很多.

2.4 低维嵌高维

考虑 $f:\mathbf{R}^{60}\to\mathbf{R}$

$$f(\mathbf{x}(\mathbf{t}))=\sum_{j=0}^{2}cos(10t_j)+sin(5t_j),\ t=(t_0,t_1,t_2)\in[0,1]^3,\ x\in\mathbf{R}^{60}$$

线形情况：

$$\mathbf{x}(\mathbf{t})_i=cos(i)*t_{[i/20]},\ i=0,1,\cdots,59$$

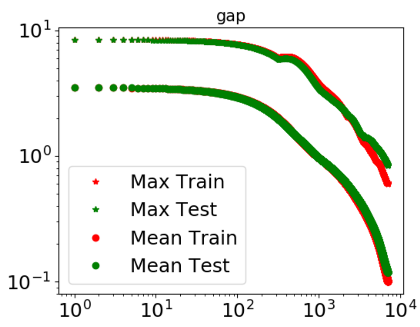

图 23: gap

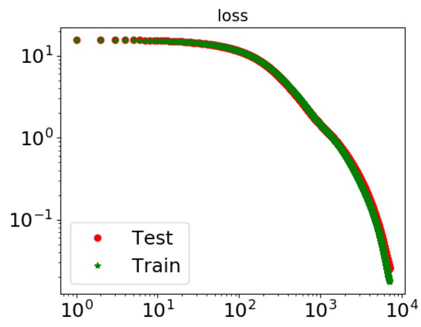

图 24: loss

非线性情况：

$$\mathbf{x}(\mathbf{t})_i=cos(cos(i)*t_{[i/20]}),\ i=0,1,\cdots,59$$

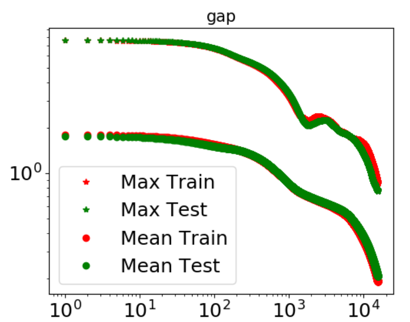

图 25: gap

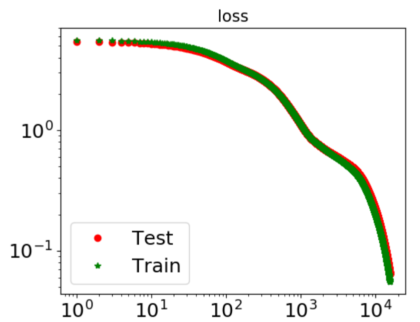

图 26: loss

在网络 60-200-200-200-1，激活函数为 $tanh$，$learning\_rate=1e-6$，$learning\_rateDecay=2e-7$ 条件下，用 Adam 算法优化. 可以看到非线性情况收敛要慢很多. 在这里的条件下，若激活函数改为 $sReLU$，速度并无明显提升.

2.5 高频函数

$$f(x) = sin(23x) + sin(137x),\ x \in [-\pi/2, \pi/2]$$

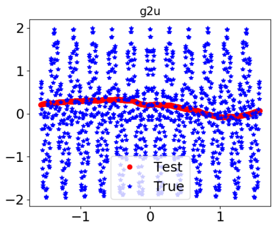

图 27: 0 epoch

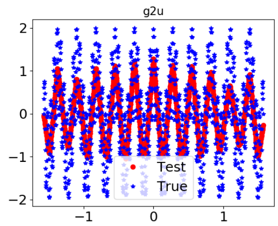

图 28: 500 epoch

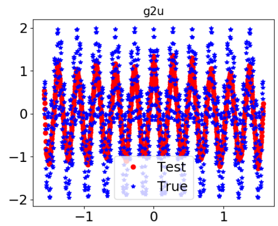

图 29: 2500 epoch

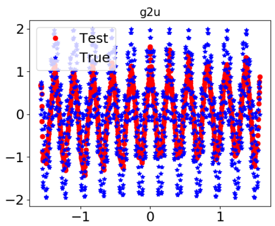

图 30: 5000 epoch

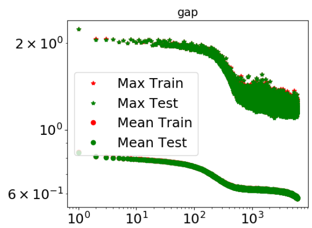

图 31: gap

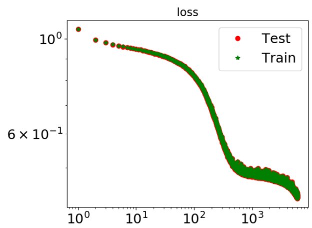

图 32: loss

在网络 1-600-300-300-300-1，激活函数为 $sReLU(= ReLU(x) \cdot ReLU(1-x))$，$learning\_rate = 5e-5$，$learning\_rateDecay = 2e-6$ 条件下，用 Adam 算法优化. 可以看到效果不是很好.

若采用 $tanh$ 或者 $ReLU$ 作为激活函数，经试验，效果更差.

下列图修改了网络的第一层，对输入的训练集初始化时，将输入的量放大 100 倍得到新的网络，其他不变，效果如下：

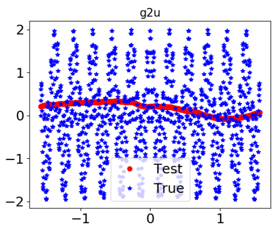

图 33: 0 epoch

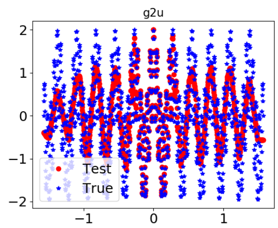

图 34: 500 epoch

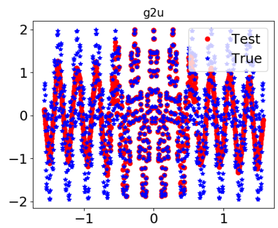

图 35: 2500 epoch

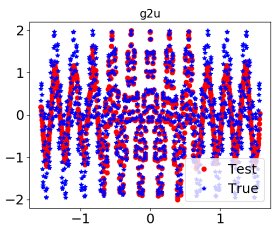

图 36: 5000 epoch

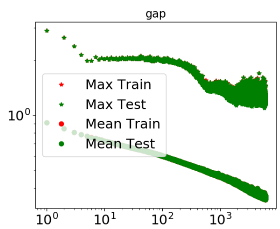

图 37: gap

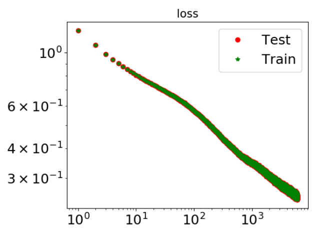

图 38: loss

相比较而言下面一种效果比较好. 但由于个人电脑性能问题，没有增加 epoch 到 10e4 以上.

2.6 解微分方程

求解 Laplace 方程

$$
\begin{aligned}
-\Delta u(\mathbf{x}) &= g(\mathbf{x}),\ \mathbf{x} \in \Omega \\
u(\mathbf{x}) &= f(\mathbf{x}),\ \mathbf{x} \in \partial\Omega
\end{aligned}
$$

有两种定义 Loss function 的方式，记我们得到的网络为 $h(x,\theta)$，有

$$
L_{Ritz} = \frac{1}{n}\sum_{x\in\Omega}(|\nabla h(x)|/2 - g(x)h(x)) + \beta * \frac{1}{\tilde{n}}\sum_{x\in\partial\Omega}(h(x)-f(x))^2
$$

或者

$$
L_{LSE} = \frac{1}{n}\sum_{x\in\Omega}(\Delta h(x) + g(x))^2 + \beta * \frac{1}{\tilde{n}}\sum_{x\in\partial\Omega}(h(x)-f(x))^2
$$

在下列图中，采用第二种 Loss function

当 $f(x) = cos(8x), g(x) = 64\ cos(8x), x \in [-\pi/2, \pi/2], n = 1000, \tilde{n} = 100, \beta = 1000$，结果如下：

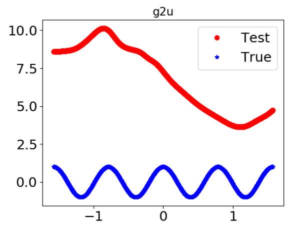

图 39: 0 epoch

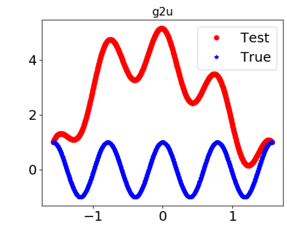

图 40: 1000 epoch

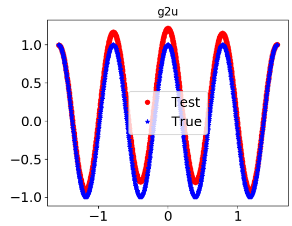

图 41: 3000 epoch

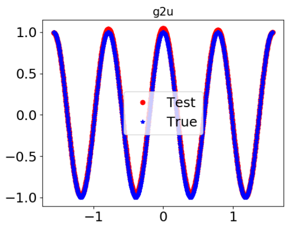

图 42: 5000 epoch

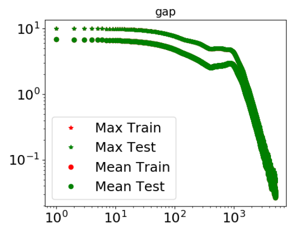

图 43: gap

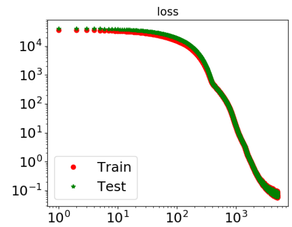

图 44: loss

在网络 1-100-100-100-1，激活函数为 $tanh$，$learning\_rate = 1e-5$，$learning\_rateDecay = 2e-6$ 条件下，用 Adam 算法优化.

当 $f(x,y) = cos(4x) + cos(4y), g(x,y) = 16\ cos(4x) + 16\ cos(4y), (x,y) \in [-\pi/2, \pi/2]^2, n = 1000, \tilde{n} = 100, \beta = 1000$，结果如下：

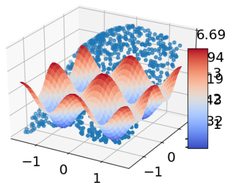

图 45: 0 epoch

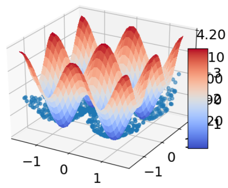

图 46: 500 epoch

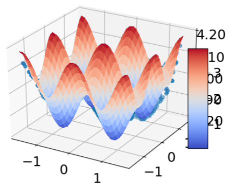

图 47: 1000 epoch

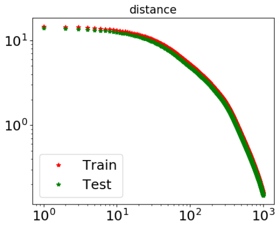

图 48: distance

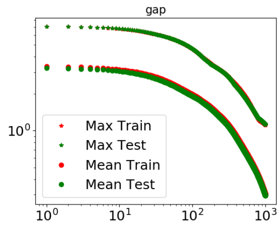

图 49: gap

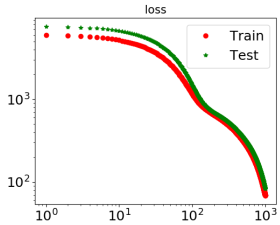

图 50: loss

在网络 2-200-200-200-1，激活函数为 $tanh$，$learning\_rate = 5e-6$，$learning\_rateDecay = 2e-6$ 条件下，用 Adam 算法优化. distance 表示和真值的 MSE Loss.

经试验，在一维情况下，第二种 Loss 平均一次 epoch 时间大概是第一种的 2-3 倍，但收敛比第一种快很多.

3 其他

总结下在调试过程中的经验：除了解调和方程的情况，其余用 DNN 的过程中，都是在中间处（0 处）最先拟合；高频或者定义域范围大振幅小的函数拟合不是很好；loss 不是平稳下降可能是 $learning\_rate$ 太大等.