# PINNs模型求解PDE的正向问题

# ——以一维Burgers方程为例

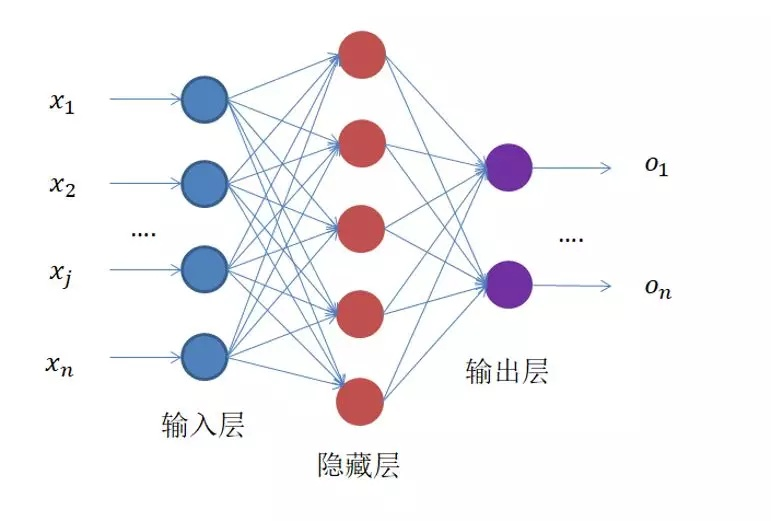
**一、全连接神经网络简介**

图1.1 全连接神经网络示意图

顾名思义，全连接神经网络即n-1层的任意一个节点，都和第n层所有节点有连接。即第n层的每个节点在进行计算的时候，激活函数的输入是n-1层所有节点的加权。

基本的结构：输入层(样本数据)，隐藏层(隐藏层的层数和每层的神经元数目需要自己给定)，输出层(预测目标)

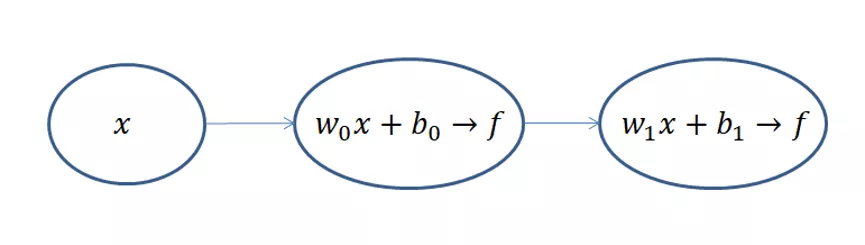


图1.2 全连接神经网络运算示意图

左边输入，中间计算，右边输出。输入为上一层各节点的线性组合，然后经过激活函数，使网络获得非线性性，最后输出至下一层。

不算输入层，上面的网络结构总共有两层，隐藏层和输出层，它们“圆圈”里的计算都是公式(1.1)和(1.2)的计算组合：

其中f为激活函数，使网络获得非线性性。每一级都是利用前一级的输出做输入，再经过圆圈内的组合计算，输出到下一级。

深度学习的基石：反向传播算法！我们需要更新参数，那就需要求得损失函数的梯度，从而pytorch所带有的自动求导功能就给我们提供了很大帮助。

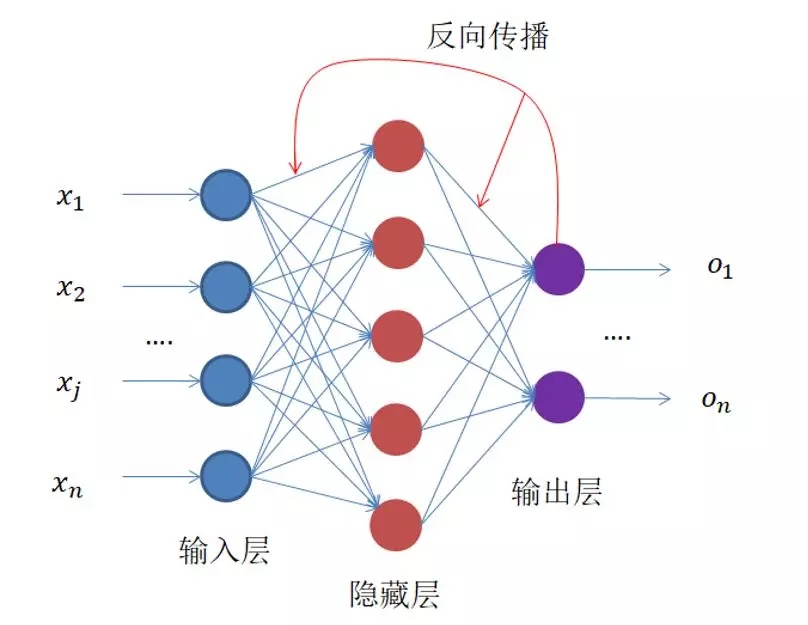


图1.3反向传播示意图

流程为：

开始

输入u，t

隐藏层

输出net\_u

得到损失函数Loss

否

Loss<eps

是

结束

**二、PINNs模型简介**

PINN求解PDE的**原理图：**

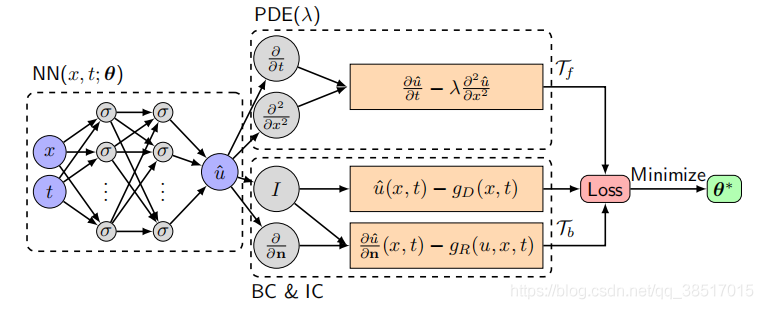


图2.1 PINNs模型原理图

PINN求解PDE的**算法步骤：**

step1：构造一个参数为θ的神经网络函数u\*(x,θ)

step2：为方程和边界及初始条件指定两个数据集T*f*和T*b*

step3：通过求偏微分方程和边界条件残差的L2范数来指定损失函数

step4：训练神经网络，通过最小化损失函数L(θ，T)来寻找最佳参数θ\*。

流程图为：

开始

输入u，t

隐藏层

输出net\_u

通过net\_u得到f

由net\_u和f得到损失函数

Loss<eps

否

是

是

结束

**三、PINNs模型求解一维Burgers方程正向问题**

Burgers方程：

(3.1)

定义：,利用Neural-Networks来逼近*u(x,t)*。定义损失函数为，

其中

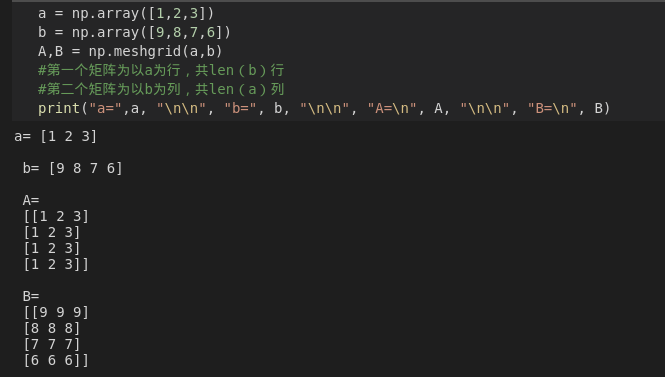
*=* (3.2)

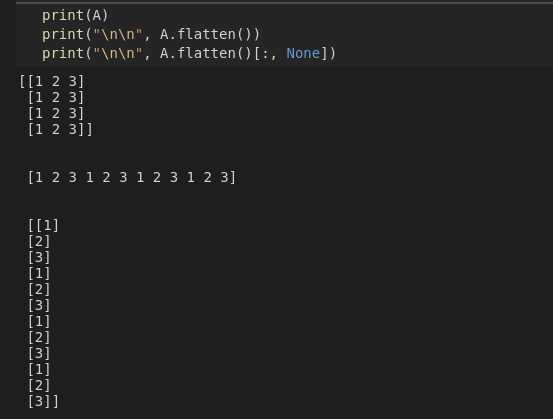
(3.3)

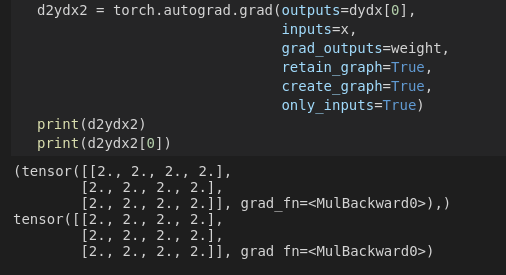
这里定义了u(x,t)的初边值训练数据，定义了f(x,t)的配置点的训练数据（内部）。

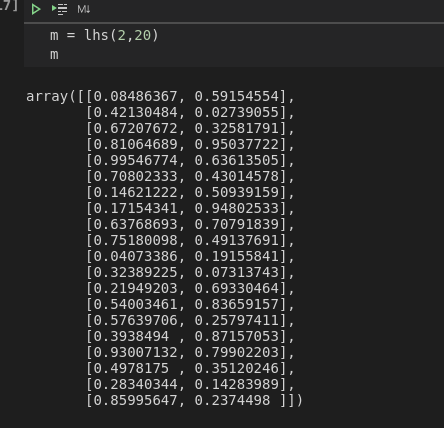
如果神经网络能很好地求解出PDE的解，那么对于来自初边值的任一个点，其值对于内部的配置点，因为很好地拟合了微分方程，。也就是说，损失函数的值为 0 时，我们便可以说在训练集上每个点都有。这样便问题便转化为如何优化损失函数。

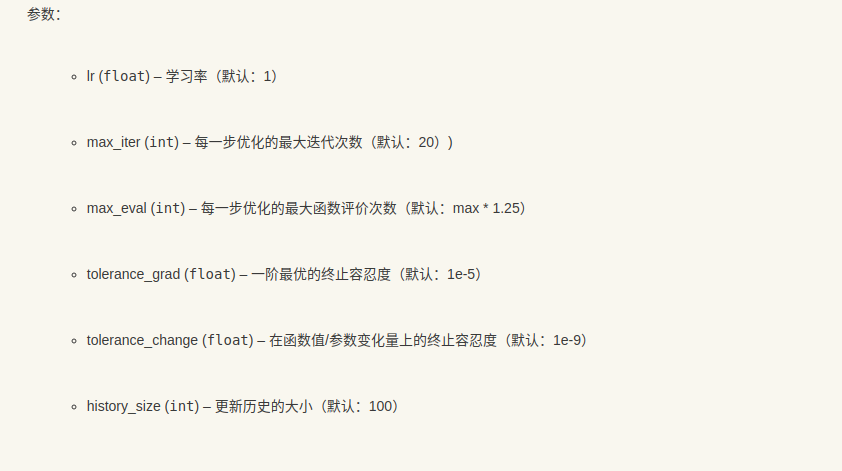
1. **代码解读**

****

****

****

****

****