智能之门

神经网络和深度学习入门

（基于Python的实现）

STEP 4 非线性回归

第 9 章

单入单出的双层神经网络

非线性回归

9.1 非线性回归问题

9.2 多项式回归法拟合曲线

9.3 验证与测试

9.4 神经网络实现非线性回归

9.5 神经网络的曲线拟合

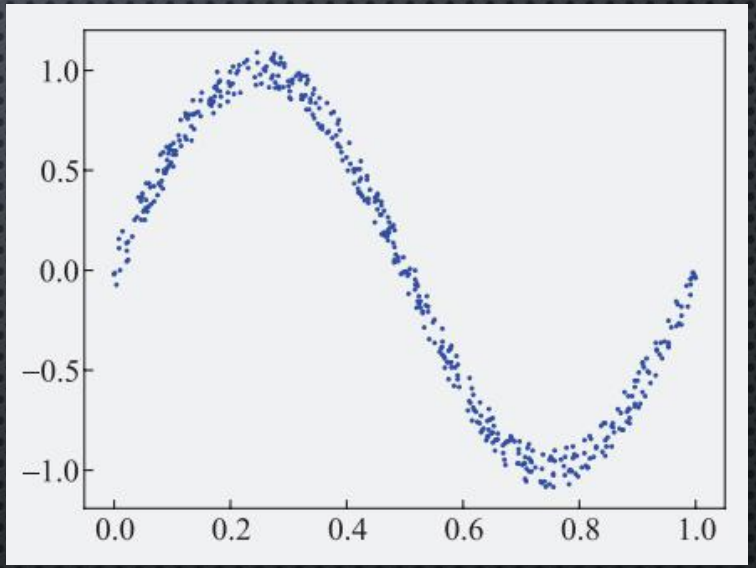
9.6 参数调优初步

本章我们将验证著名的万能近似定理，建立一个双层的神经网络，来拟合一个比较复杂的函数，实现非线性回归问题。在上面的双层神经网络中，已经出现了很多的超参，都会影响到神经网络的训练结果。所以在完成了基本的拟合任务之后，我们将会尝试着调试这些参数，得到更好的训练效果，从而得到超参调试的第一手经验。

9.1 非线性回归问题

前面学习了线性回归的解决方案，但是在工程实践中，最常遇到不是线性问题，而是非线性问题，如下面正弦函数的例子。

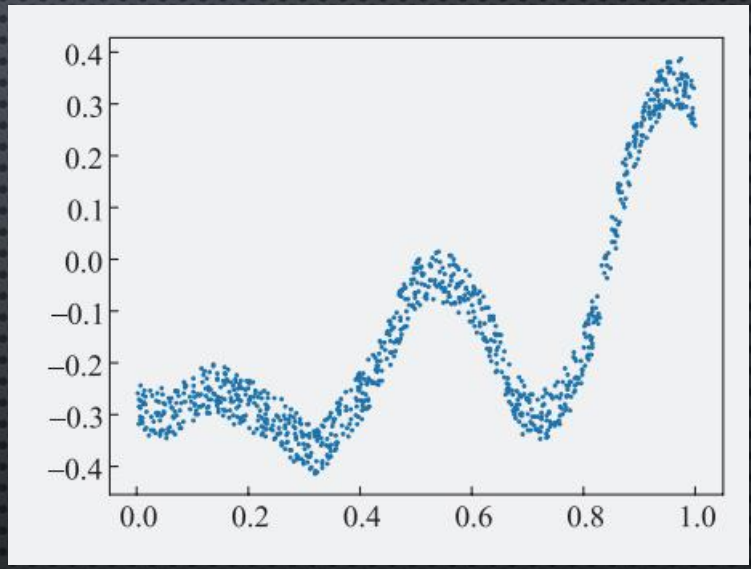
样本	*x*	*y*
1	0.1199	0.6108
2	0.0535	0.3832
3	0.6978	0.9496
...	...	...



9.1 非线性回归问题

正弦函数非常有规律，也许单层神经网络很容易就能完成拟合了。如果是更复杂的曲线，单层神经网络还能轻易做到吗？看下面的例子。

样本	*x*	*y*
1	0.606	−0.113
2	0.129	−0.269
3	0.582	0.027
…	…	…
1000	0.199	−0.281



上面这条“蛇形”曲线，实际上是由下面这个公式添加噪音后生成的。

$$y = 0.4x^2 + 0.3x \sin 15x + 0.01 \cos 50x - 0.3$$

9.1 非线性回归问题

➢ 回归模型评估标准

- 平均绝对误差（MAE）

$$MAE = \frac{1}{m}\sum_{i=1}^{m}|a_i - y_i|$$

- 绝对平均值率误差（MAPE）

$$MAPE = \frac{100}{m}\sum_{i=1}^{m}\left|\frac{a_i - y_i}{y_i}\right|$$

- 和方差（SSE）

$$SSE = \sum_{i=1}^{m}(a_i - y_i)^2$$

- 均方差（MSE）

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(a_i - y_i)^2$$

- 均方根误差（RMSE）

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(a_i - y_i)^2}$$

- R平方（R-Squared）

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(a_i - y_i)^2}{\sum_{i=1}^{m}(\bar{y} - y_i)^2} = 1 - \frac{MSE(a,y)}{Var(y)}$$

9.2 多项式回归法拟合曲线

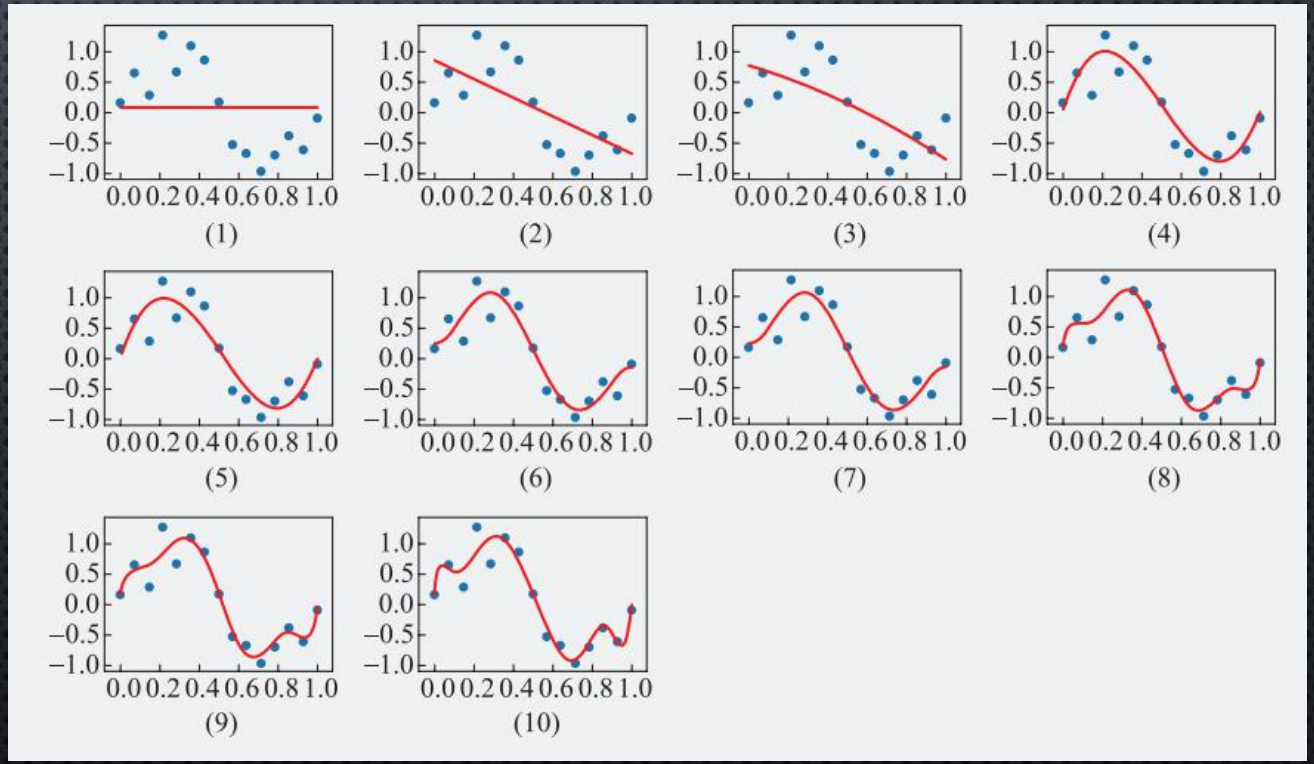
➢ 多项式回归的几种形式

- 一元一次线性模型，可解决单变量线性回归问题：$z = xw + b$。
- 多元一次线性模型，可解决多变量线性回归问题：$z = x_1 w_1 + x_2 w_2 + \cdots + x_m w_m + b$。
- 一元多次多项式：$z = xw_1 + x^2 w_2 + \cdots + x^m w_m + b$。
  - ✓ 以特征值的高次方作为新的特征值，即令 $x_k = x^k, k = 1,2,\cdots,m$，则化为多变量线性回归问题。
  - ✓ Weierstrass 定理：闭区间上的任意连续函数一定可以被某个多项式一致逼近。
- 多元多次多项式：多变量的非线性回归，参数与特征组合繁复，但是最终可以化为多变量线性回归问题。

9.2 多项式回归法拟合曲线

➢ 过拟合与欠拟合

- 右图展示了不同次数多项式对含有噪音的正弦曲线的拟合结果，依次表示1到10次多项式的拟合结果。其中图1、2、3欠拟合，多项式次数较低；图4、5、6、7拟合效果较为理想；图8、9、10过拟合，多项式次数较高。



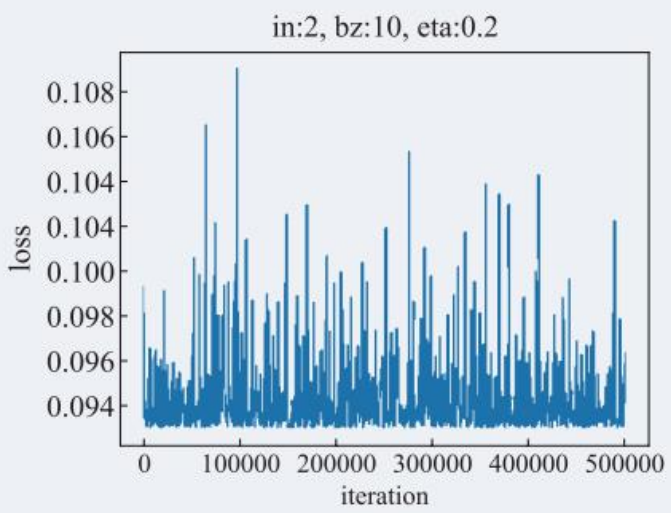
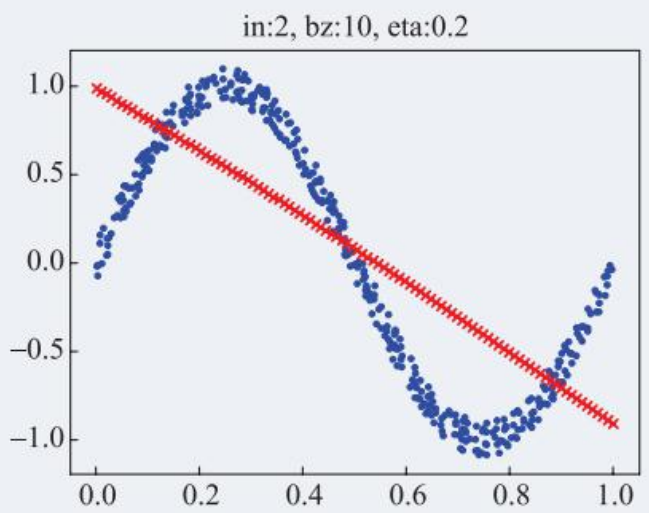
9.2 多项式回归法拟合曲线

- **多项式拟合参数**
  - 由于数据已经作了归一化处理，所以项次数越高，权重值越大，特征值与权重值的乘积才能成为一个不太小的数，以此弥补特征值过小的问题。

多项式次数 / 权重	1	2	3	4	5	6	7	8	9	10
x_1	−0.096	0.826	0.823	0.033	0.193	0.413	0.388	0.363	0.376	0.363
x_2		−1.84	−1.82	9.68	5.03	−7.21	−4.50	1.61	−6.46	18.39
x_3			−0.017	−29.80	−7.17	90.05	57.84	−43.49	131.77	−532.78
x_4				19.85	−16.09	−286.93	−149.63	458.26	−930.65	5 669.0
x_5					17.98	327.00	62.56	−1 669.06	3 731.38	−29 316.1
x_6						−123.61	111.33	2 646.22	−8 795.97	84 982.2
x_7							−78.31	−1 920.56	11 551.86	−145 853
x_8								526.35	−7 752.23	147 000
x_9									2 069.6	−80 265.3
x_{10}										18 296.6

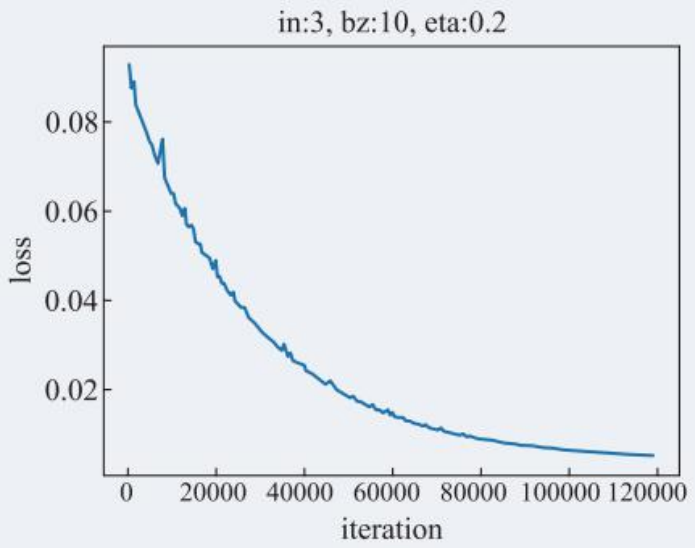
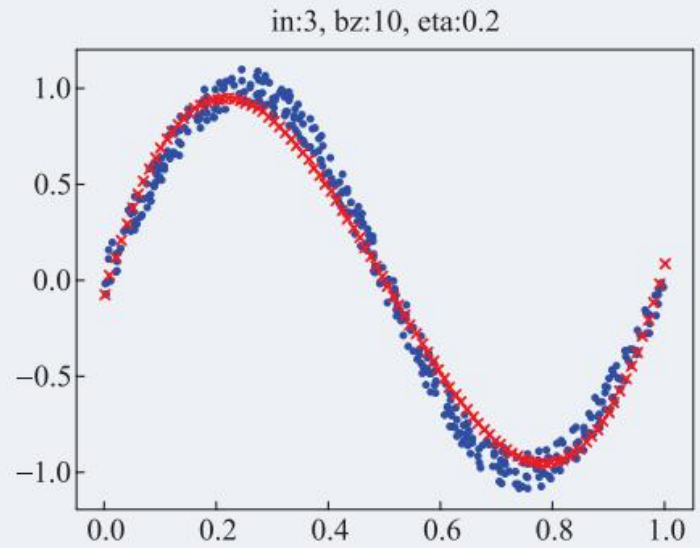
9.2 多项式回归法拟合曲线

- **正弦曲线-训练结果**
  - 二次多项式

损失函数值	拟合结果
	

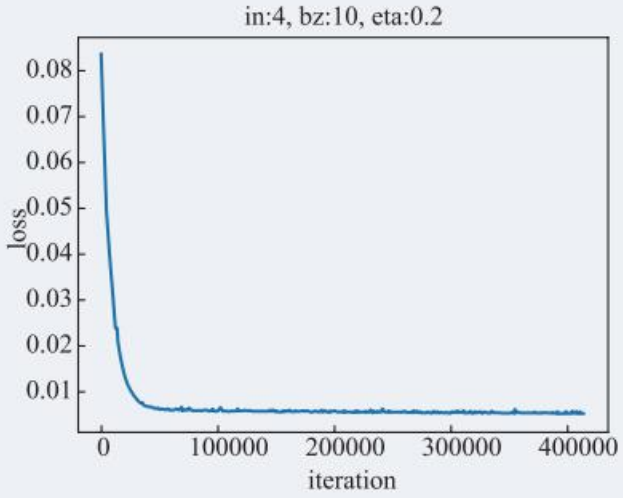
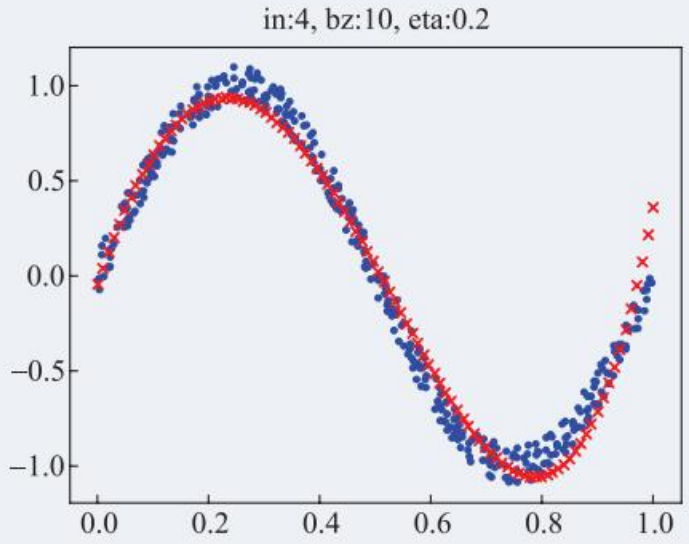
9.2 多项式回归法拟合曲线

- 三次多项式

损失函数值	拟合结果
	

9.2 多项式回归法拟合曲线

- 四次多项式

损失函数值	拟合结果
	

9.2 多项式回归法拟合曲线

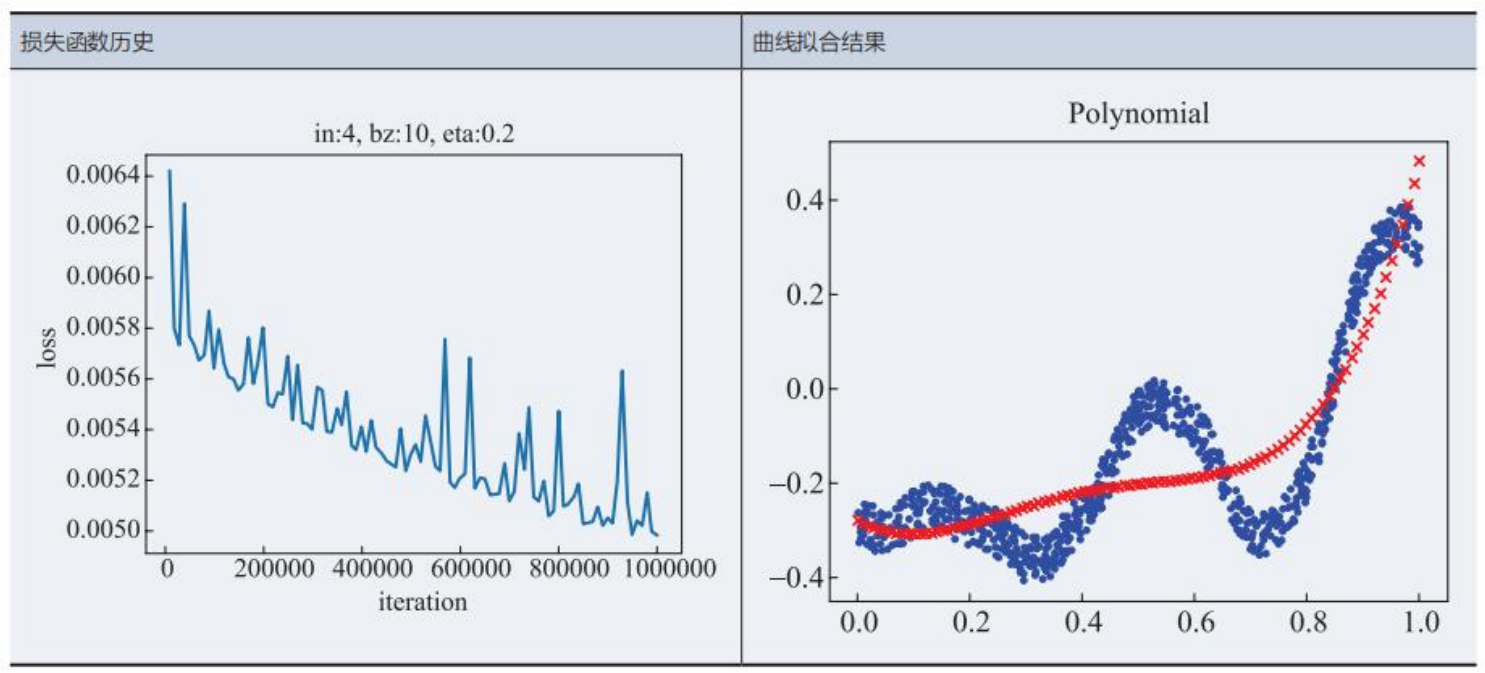
➢ 结果比较

- 二次多项式的损失值在下降了一定程度后，一直处于平缓期，不再下降，说明网络能力到了一定的限制，直到 10000 次迭代也没有达到目的。
- 损失值达到 0.005 时，四项式迭代了 8290 次，比三次多项式要多很多，说明四次多项式多出的一个特征值，不仅没有带来好处，反而是增加了网络训练的复杂度。

多项式次数	迭代数	损失函数值
2	10 000	0.095
3	2 380	0.005
4	8 290	0.005

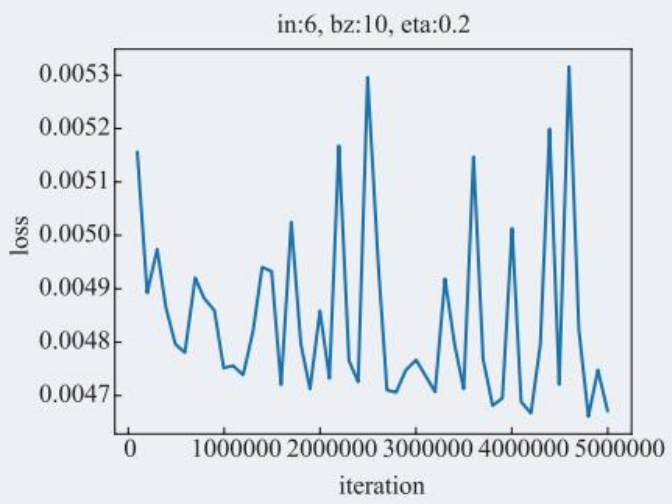
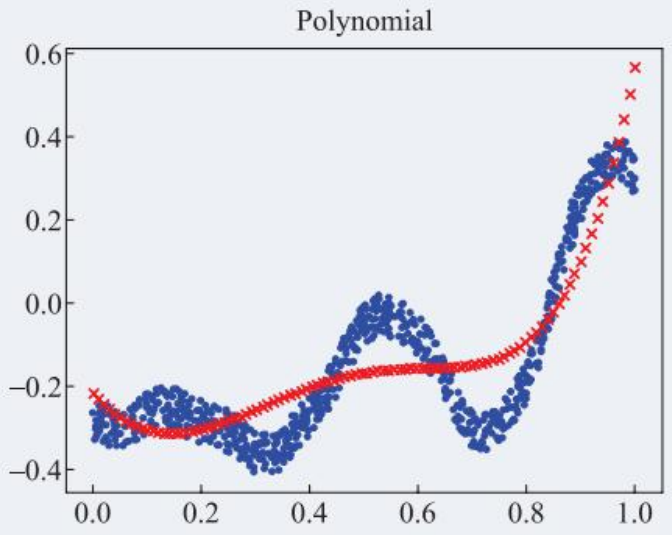
9.2 多项式回归法拟合曲线

- **蛇形曲线-训练结果**
  - 四次多项式



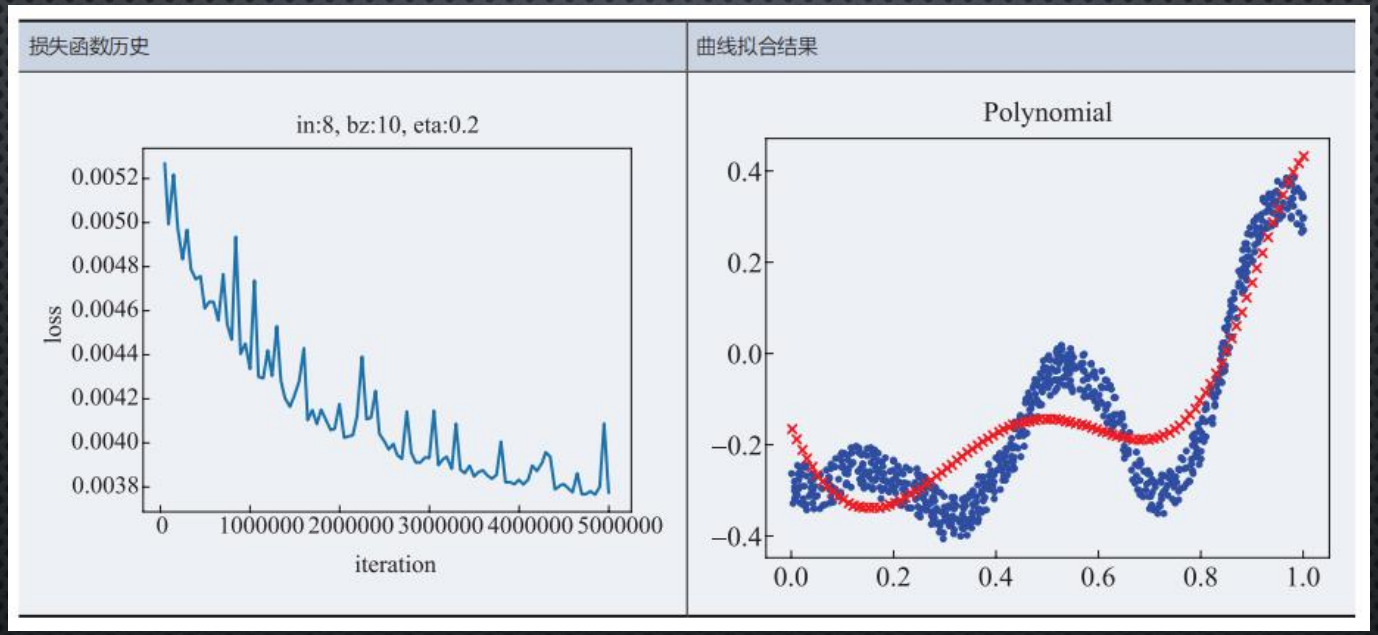
9.2 多项式回归法拟合曲线

- 六次多项式

损失函数历史	曲线拟合结果
	

9.2 多项式回归法拟合曲线

- 八次多项式

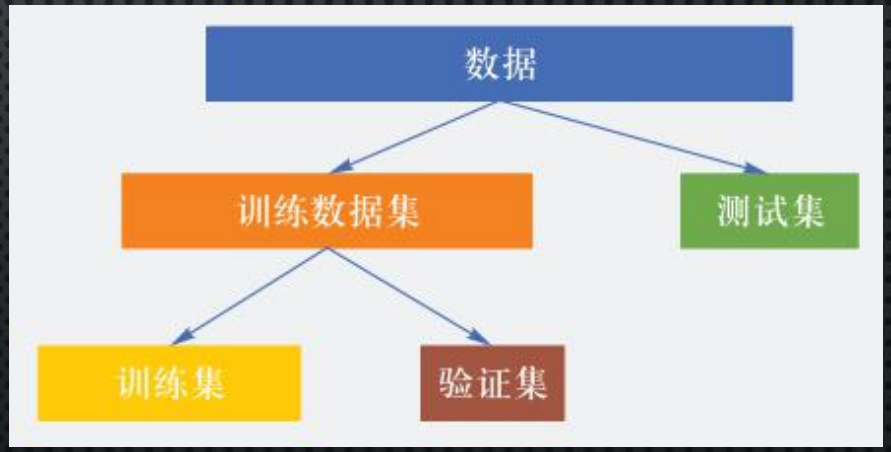


多项式回归确实可以解决复杂曲线拟合问题，但是代价有些高，迭代次数过大。

9.3 验证与测试

➢ 基本概念

- 训练集：用于模型训练的数据样本。
- 验证集：模型训练过程中单独留出的样本集，它可以用于调整模型的超参数和用于对模型的能力进行初步评估。在神经网络中，验证数据集用于：
  - ✓ 寻找最优的网络深度。
  - ✓ 决定反向传播算法的停止点。
  - ✓ 在神经网络中选择隐藏层神经元的数量。
  - ✓ 交叉验证，就是把训练数据集本身再细分成不同的验证数据集去训练模型。
- 测试集：用来评估最终模型的泛化能力；但不能作为调参、选择特征等算法相关的选择依据。



9.3 验证与测试

➢ 交叉验证

- 随机将训练数据分成 K 等份（通常建议$K = 10$），得到$D_0, D_1, \cdots, D_9$；
- 对于一个模型，选择D_9为验证集，其它为训练集，训练若干轮，用D_9验证，得到误差E。再训练，再用D_9测试，如此N次。对N次的误差做平均，得到平均误差；
- 换一个不同参数的模型的组合，比如神经元数量，或者网络层数，激活函数，重复步骤2，但是这次用D_8去得到平均误差；
- 重复步骤2，一共验证10组组合；
- 最后选择具有最小平均误差的模型结构，用所有的$D_0, D_1, \cdots, D_9$再次训练，不再验证；
- 用测试集测试。

9.3 验证与测试

➢ 留出法

- 使用交叉验证的方法虽然比较保险，但是非常耗时，尤其是在大数据量时，训练出一个模型都要很长时间，没有可能去训练出10个模型再去比较。
- 在深度学习中，有另外一种方法使用验证集，称为留出法。亦即从训练数据中保留出验证样本集，主要用于解决过拟合情况，这部分数据不用于训练。如果训练数据的准确度持续增长，但是验证数据的准确度保持不变或者反而下降，说明神经网络亦即过拟合了，此时需要停止训练，用测试集做最终测试。

```
for each epoch
    shuffle
    for each iteraion
        获得当前小批量数据
        前向计算
        反向传播
        更新梯度
        if is checkpoint
            用当前小批量数据计算训练集的 loss 值和 accuracy 值并记录
            计算验证集的 loss 值和 accuracy 值并记录
            如果 loss 值不再下降，停止训练
            如果 accuracy 值满足要求，停止训练
        end if
    end for
end for
```

9.3 验证与测试

- **比例**
  - 关于三者的比例关系，在传统的机器学习中，三者可以是6: 2: 2。
  - 在深度学习中，一般要求样本数据量很大，所以可以给训练集更多的数据，比如8: 1: 1。
  - 如果有些数据集已经给了你训练集和测试集，那就不关心其比例问题了，只需要从训练集中留出 10% 左右的验证集就可以了。

9.4 神经网络实现非线性回归

➢ 万能近似定理

- 万能近似定理是深度学习最根本的理论依据。它证明了在给定网络具有足够多的隐藏单元的条件下，配备一个线性输出层和一个带有任何“挤压”性质的激活函数（如Sigmoid激活函数）的隐藏层的前馈神经网络，能够以任何想要的误差量近似任何从一个有限维度的空间映射到另一个有限维度空间的Borel可测的函数。
- 万能近似定理其实说明了理论上神经网络可以近似任何函数。但实践上我们不能保证学习算法一定能学习到目标函数，即使可以，学习也可能因为两个不同的原因而失败：
  - ✓ 用于训练的优化算法可能找不到用于期望函数的参数值；
  - ✓ 训练算法可能由于过拟合而选择了错误的函数。
- 根据“没有免费的午餐”定理，普遍优越的机器学习算法不存在。前馈网络提供了表示函数的万能系统，在这种意义上，给定一个函数，存在一个前馈网络能够近似该函数。但不存在万能的过程既能够验证训练集上的特殊样本，又能够选择一个函数来扩展到训练集上没有的点。

9.4 神经网络实现非线性回归

➢ 神经网络结构

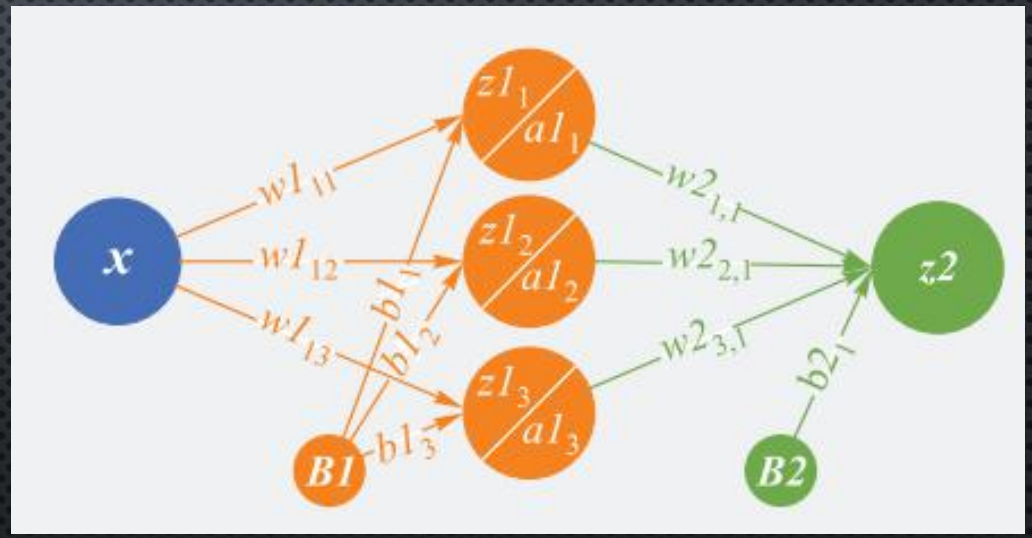
- 采用双层神经网络。
- 输入：X，一个神经元。
- 隐藏层：$Z1, A1$，各含三个神经元。
- 输出：$Z2$，一个神经元。
- 权重矩阵

$$W1 = (W1_{11} \quad W1_{12} \quad W1_{13})$$

$$B1 = (B1_1 \quad B1_2 \quad B1_3)$$

$$W2 = (W2_{11} \quad W2_{21} \quad W2_{31})^T$$

$$B2 = (B2_1)$$



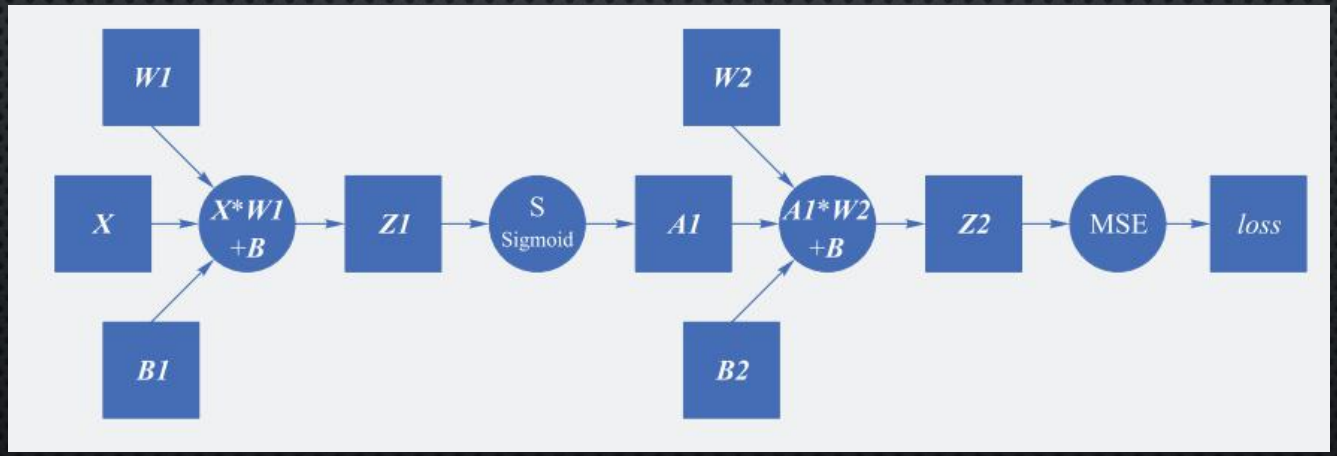
9.4 神经网络实现非线性回归

➢ 前向计算

- 层间计算

$$Z1 = X \cdot W1 + B1, \qquad A1 = Sigmoid(Z1), \qquad Z2 = A1 \cdot W2 + B2$$

- 损失函数：MSE。



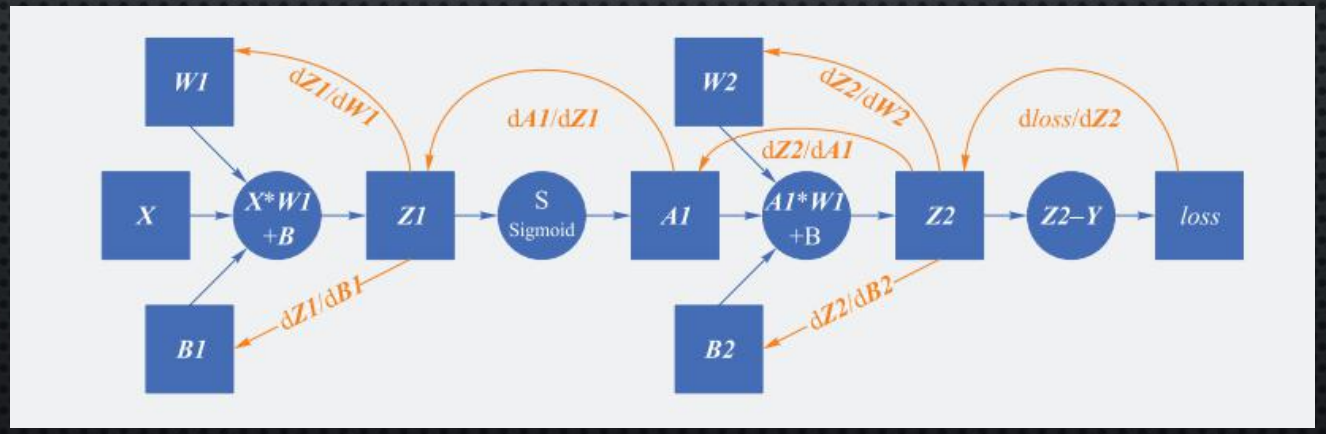
9.4 神经网络实现非线性回归

➢ 反向传播

$$dZ2 = \frac{\partial loss}{\partial Z2} = Z2 - Y, \qquad dW2 = \frac{\partial loss}{\partial W2} = A1^T dZ2, \qquad dB2 = \frac{\partial loss}{\partial B2} = Z2 - Y$$

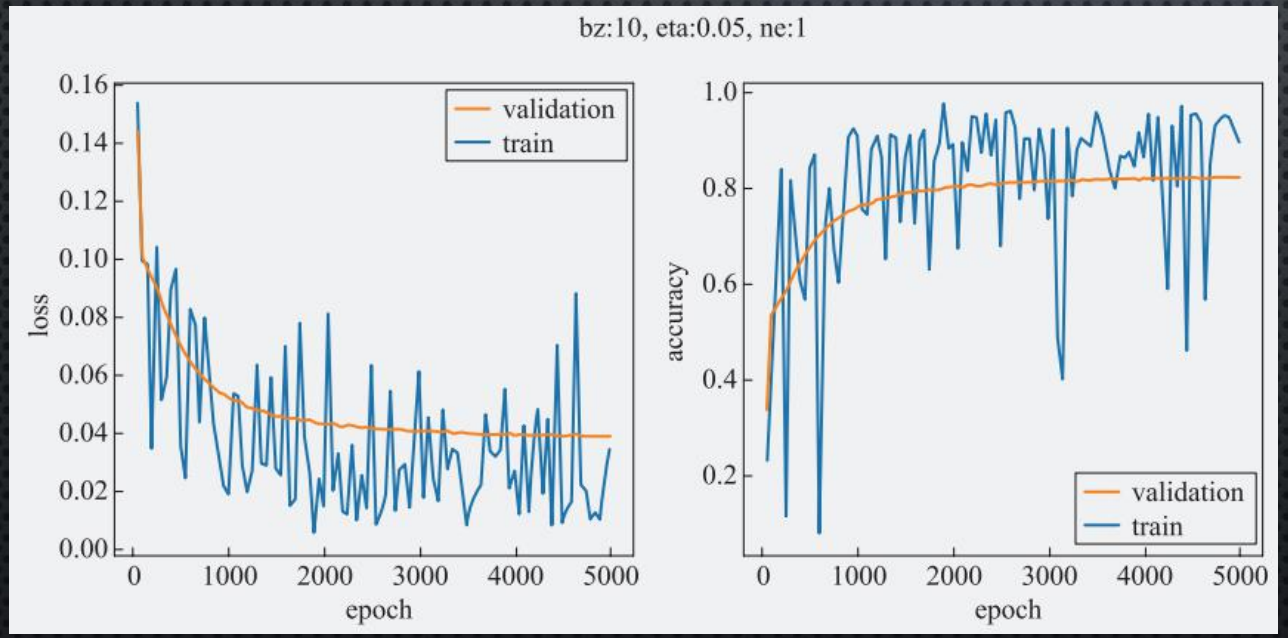
$$dA1 = \frac{\partial loss}{\partial A1} = dZ2 \cdot W2^T, \qquad dZ1 = \frac{\partial loss}{\partial Z1} = dZ2 \cdot W2^T \odot dA1,$$

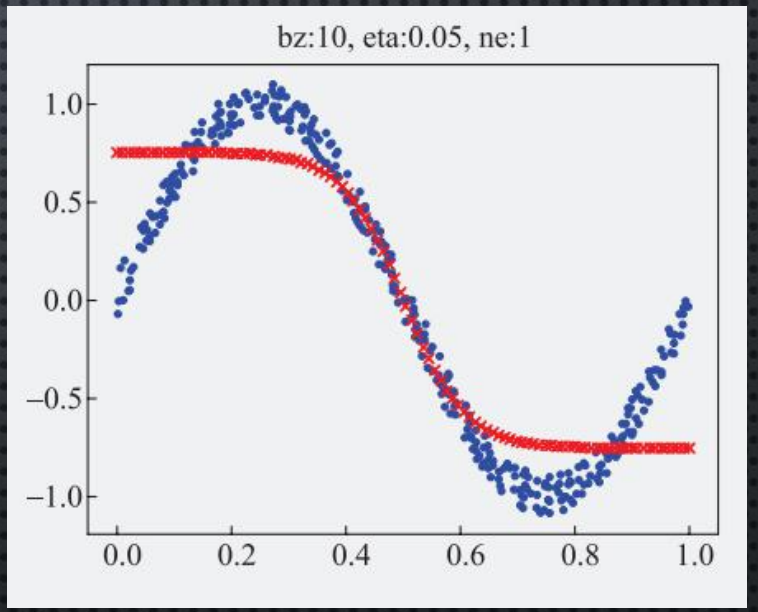
$$dW1 = X^T \cdot dZ1, \qquad dB1 = dZ1$$



9.5 神经网络的曲线拟合

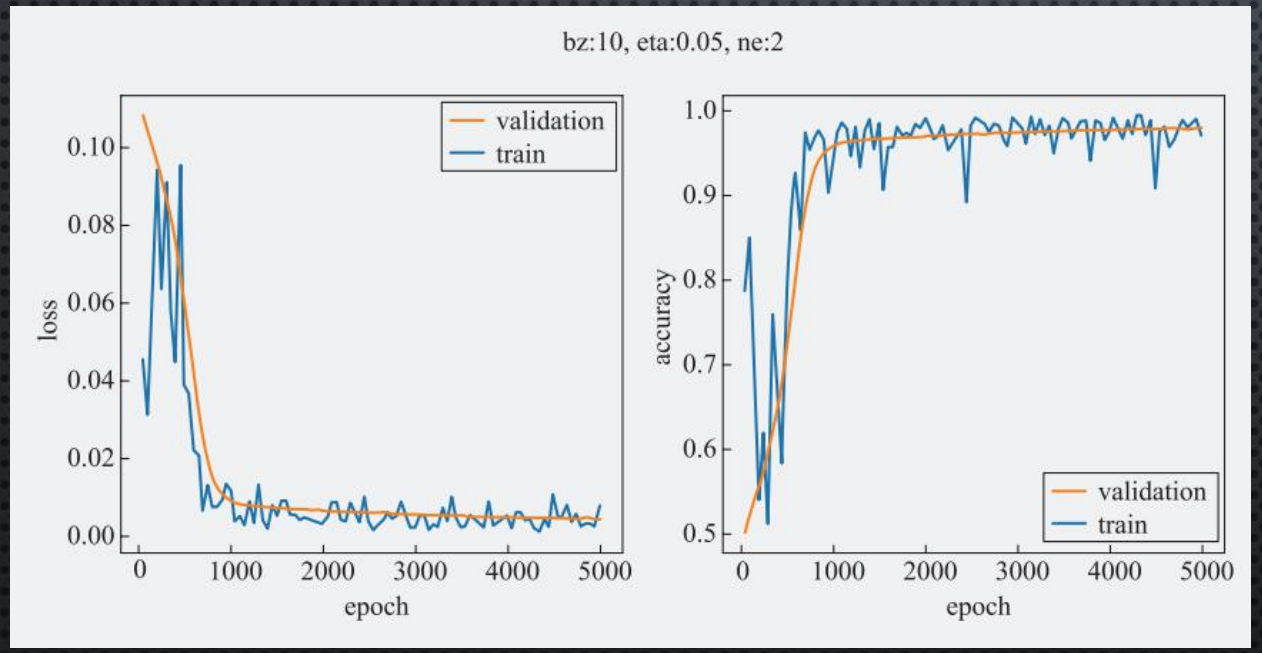
- 正弦曲线的拟合
  - 隐藏层只有一个神经元

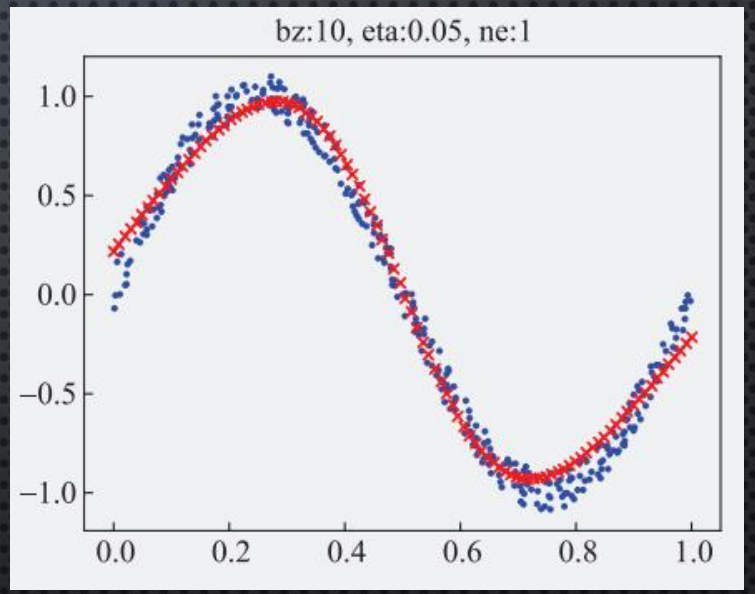




9.5 神经网络的曲线拟合

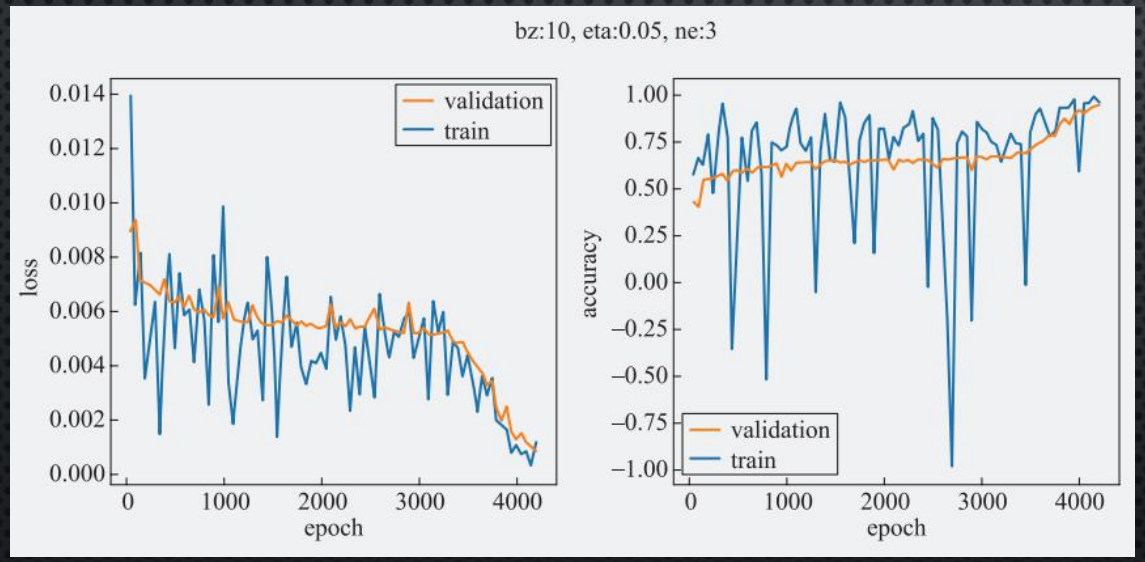
- 隐藏层有2个神经元

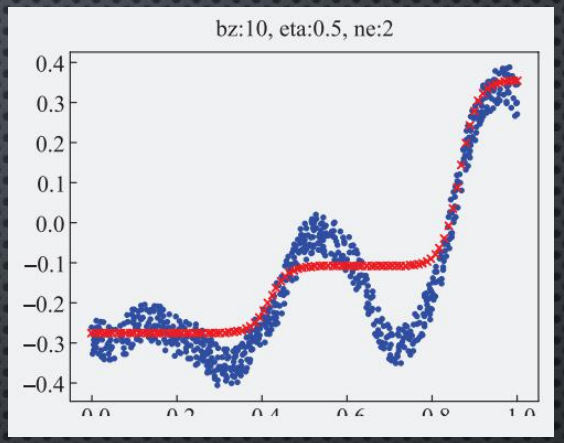


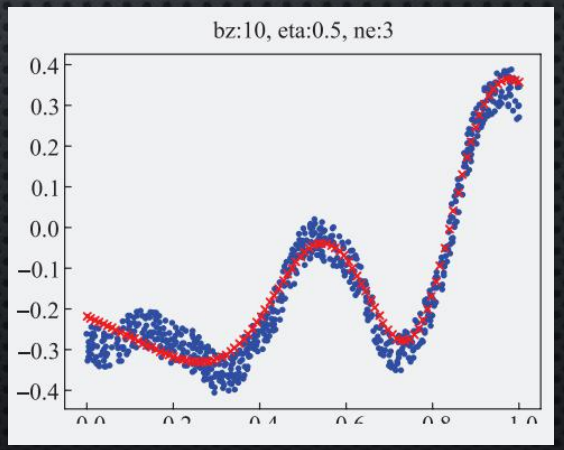


9.5 神经网络的曲线拟合

- “蛇形”曲线的拟合
  - 隐藏层有2个神经元，拟合效果不理想，如右图。
  - 隐藏层有3个神经元的情况，如下图。







9.5 神经网络的曲线拟合

- 广义的回归、拟合
  - “曲线”在这里是一个广义的概念，它不仅可以代表二维平面上的数学曲线，也可以代表工程实践中的任何拟合问题，比如房价预测问题，影响房价的自变量可以达到20个左右，显然已经超出了线性回归的范畴，此时我们可以用多层神经网络来做预测。在后面我们会讲解这样的例子。简言之，只要是数值拟合问题，确定不能用线性回归的话，都可以用非线性回归来尝试解决。
- 多项式回归和双层神经网络解法的比较

	多项式回归	双层神经网络
特征提取方式	特征值的高次方	线性变换拆分
特征值数量级	高几倍的数量级	数量级与原特征值相同
训练效率	低，需要迭代次数多	高，比前者少好几个数量级

9.6 参数调优初步

- **超参数优化主要存在两方面的困难：**
  - 超参数优化是一个组合优化问题，无法像一般参数那样通过梯度下降方法来优化，也没有一种通用有效的优化方法。
  - 评估一组超参数配置的时间代价非常高，从而导致一些优化方法（比如演化算法）在超参数优化中难以应用。
- **对于超参数的设置，比较简单的方法有人工搜索、网格搜索和随机搜索。**
- **权重矩阵中的参数，是神经网络要学习的参数，所以不能称作超参数。**

参数	缺省值	是否可调	注释
输入层神经元数	1	No	
隐层神经元数	4	Yes	影响迭代次数
输出层神经元数	1	No	
学习率	0.1	Yes	影响迭代次数
批样本量	10	Yes	影响迭代次数
最大epoch	10000	Yes	影响终止条件,建议不改动
损失门限值	0.001	Yes	影响终止条件,建议不改动
损失函数	MSE	No	
权重矩阵初始化方法	Xavier	Yes	参看15.1

9.6 参数调优初步

➢ 手动调整参数

- 手动调整超参数的主要目标是调整模型的有效容量以匹配任务的复杂性。有效容量受限于3个因素：
  - ✓ 模型的表示容量；
  - ✓ 学习算法与代价函数的匹配程度；
  - ✓ 代价函数和训练过程正则化模型的程度。

超参数	目标	作用	副作用
学习率	调至最优	低的学习率会导致收敛慢，高的学习率会导致错失最佳解	容易忽略其它参数的调整
隐层神经元数量	增加	增加数量会增加模型的表示能力	参数增多、训练时间增长
批大小	有限范围内尽量大	大批量的数据可以保持训练平稳，缩短训练时间	可能会收敛速度慢

9.6 参数调优初步

➢ 网格搜索

- 当有3个或更少的超参数时，常见的超参数搜索方法是网格搜索。对于每个超参数，选择一个较小的有限值集去试验。然后，这些超参数的笛卡儿乘积（所有的排列组合）得到若干组超参数，网格搜索使用每组超参数训练模型。挑选验证集误差最小的超参数作为最好的超参数组合。
- 用学习率和隐层神经元数量来举例，横向为学习率，取值[0.1,0.3,0.5,0.7]；纵向为隐层神经元数量，取值[2,4,8,12]，在每个组合上测试验证集的精度，结果如右表。其中最佳的组合精度达到0.97，学习率0.5，神经元数8，那么这个组合就是我们需要的模型超参。
- **网格搜索带来的一个明显问题是，计算代价会随着超参数数量呈指数级增长。**

	eta=0.1	eta=0.3	eta=0.5	eta=0.7
ne=2	0.63	0.68	0.71	0.73
ne=4	0.86	0.89	0.91	0.3
ne=8	0.92	0.94	0.97	0.95
ne=12	0.69	0.84	0.88	0.87

9.6 参数调优初步

- **随机搜索：是一个替代网格搜索的方法，并且编程简单，使用更方便，能更快地收敛到超参数的良好取值。**
  - 首先，我们为每个超参 数定义一个边缘分布，例如，Bernoulli分布或范畴分布（分别对应着二元超参数或离散超参数），或者对数尺度上的均匀分布（对应着正实值超参数）。
  - 与网格搜索不同，我们不需要离散化超参数的值。这允许我们在一个更大的集合上进行搜索，而不产生额外的计算代价。实际上，当有几个超参数对性能度量没有显著影响时，随机搜索相比于网格搜索指数级地高效。
  - 与网格搜索一样，我们通常会重复运行不同版本的随机搜索，以基于前一次运行的结果改进下一次搜索。
  - 在网格搜索中，其他超参数将在这两次实验中拥有相同的值，而在随机搜索中，它们通常会具有不同的值。因此，如果这两个值的变化所对应的验证集误差没有明显区别，网格搜索没有必要重复两个等价的实验，而随机搜索仍然会对其他超参数进行两次独立的探索。

THE END

谢谢！