2020 年春季学期
计算学部《机器学习》课程

Lab 1 实验报告

姓名	周牧云
学号	1180300315
班号	1803501
电子邮件	zhou_mu_yun@163.com
手机号码	13912263240

目录

建议写出：问题的描述，解决问题的思路，实验的做法，实验结果的分析，结论，自拟标题

一、实验目的

掌握最小二乘法求解（无惩罚项的损失函数）、掌握加惩罚项（2 范数）的损失函数优化、梯度下降法、共轭梯度法、理解过拟合、克服过拟合的方法(如加惩罚项、增加样本)

二、实验要求

1. 生成数据，加入噪声；
2. 用高阶多项式函数拟合曲线；
3. 用解析解求解两种 loss 的最优解（无正则项和有正则项）
4. 优化方法求解最优解（梯度下降，共轭梯度）；
5. 用你得到的实验数据，解释过拟合。
6. 用不同数据量，不同超参数，不同的多项式阶数，比较实验效果。
7. 语言不限，可以用 matlab，python。求解解析解时可以利用现成的矩阵求逆。梯度下降，共轭梯度要求自己求梯度，迭代优化自己写。不许用现成的平台，例如 pytorch，tensorflow 的自动微分工具。

三、实验内容

1、算法原理

本实验需要用多项式来拟合正弦函数。在 m 阶多项式中，有 m+1 个待定系数，m+1 个系数（由低到高）组成的（列）向量记作 w。要确定 w，用最小二乘法。

设 $E(w) = 1/2 * (Xw - Y)^T(Xw - Y)$，其中，X 为多项式中各个未知项代入观测数据求得的矩阵，若记 X_i 为 X 的第 i 行的向量，则 $X_i[j]$为第 i 个观测数据 x_i 的 j 次方，记有 n 组观测数据，多项式最高次为 m，易知 X 的维度为 $n * (m+1)$。Y 为观测标签向量。即 Y[j]为第 j 组观测数据的标签值（即 y 值）。从而问题转化为：求向量 w，使得 E(w)最小。
若不加入正则项，令损失函数导数为零，求 w
若加入正则项，令损失函数导数为零，求 w
加入正则项，对损失函数用梯度下降，当损失函数收敛时，求 w
加入正则项，对损失函数用共轭梯度法，循环迭代 m+1 次，求 w

2、算法实现

1）生成数据，加入噪声

```
def genData(mu, sigma):
    train_x = np.arange(0, 1, 1 / sample_n)
    gauss_noise = np.random.normal(mu, sigma, sample_n)
    train_y = np.sin(train_x * 2 * np.pi) + gauss_noise
    return train_x, train_y
```

2）用高阶多项式函数拟合曲线；

```
def np_polyfit(train_x, train_y):
    w = np.polyfit(train_x, train_y, poly_degree)
    poly = np.poly1d(w)
    return poly
```

3）用解析解求解两种 loss 的最优解（无正则项和有正则项）
无正则项：

```
def lsm_loss(train_x, train_y):
    X = genMatrixX(train_x)
    Y = train_y.reshape((sample_n, 1))
    w = np.linalg.inv(np.dot(X.T, X)).dot(X.T).dot(Y)
    poly = np.poly1d(w[::-1].reshape(poly_degree + 1))
    return poly
```

有正则项：

```
def lsm_punished_loss(train_x, train_y, lam):
    X = genMatrixX(train_x)
    Y = train_y.reshape((sample_n, 1))
    w = np.linalg.inv(np.dot(X.T, X) + lam * np.eye(X.shape[1])).dot(X.T).dot(Y)
    poly = np.poly1d(w[::-1].reshape(poly_degree + 1))
    return poly
```

4）优化方法求解最优解（梯度下降，共轭梯度）

梯度下降法：

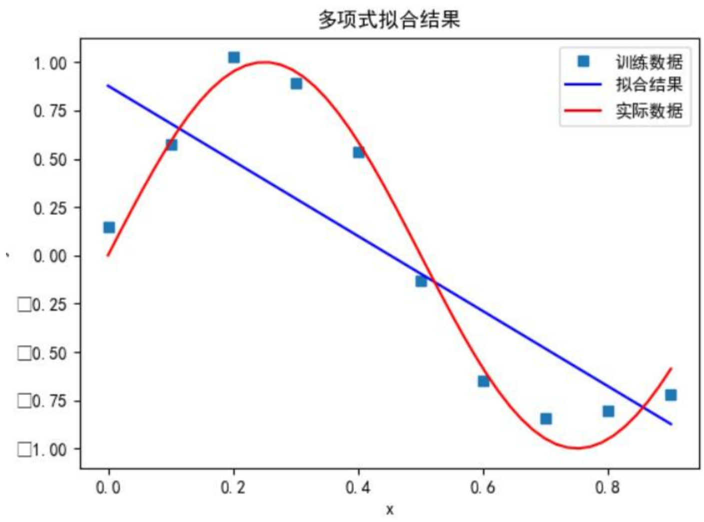
```python
def descent_gradient(train_x, train_y, lam, epoch, eta, eps):
    w = 0.1 * np.ones((poly_degree + 1, 1))
    X = genMatrixX(train_x)
    Y = train_y.reshape((sample_n, 1))
    epoch_list = np.zeros(epoch)
    loss_list = np.zeros(epoch)

    for i in range(epoch):
        old_loss = abs(loss(train_x, train_y, w))
        partial_deriv = X.T.dot(X).dot(w) - X.T.dot(Y) + lam * w
        w = w - eta * partial_deriv
        new_loss = abs(loss(train_x, train_y, w))
        epoch_list[i] = i
        loss_list[i] = new_loss
        if(abs(new_loss - old_loss) < eps):
            epoch_list = epoch_list[:i+1]
            loss_list = loss_list[:i+1]
            break
    poly = np.poly1d(w[::-1].reshape(poly_degree + 1))
    return poly, epoch_list, loss_list
```

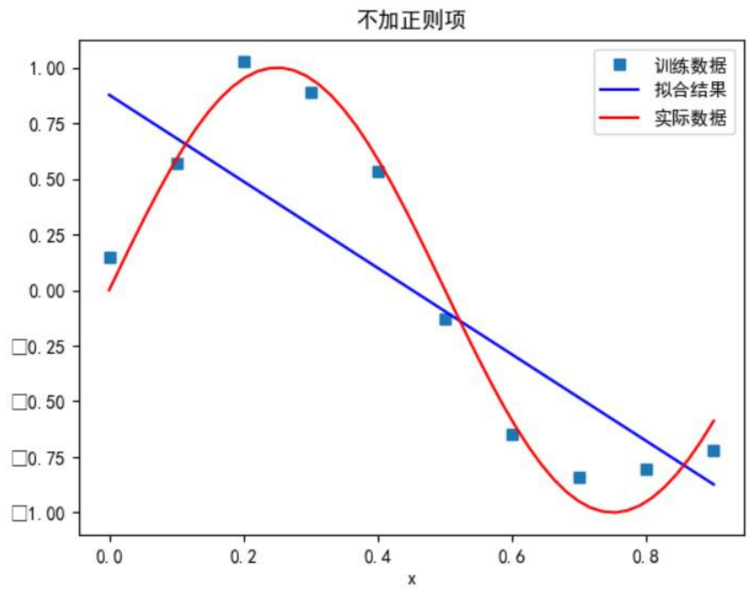
共轭梯度法：

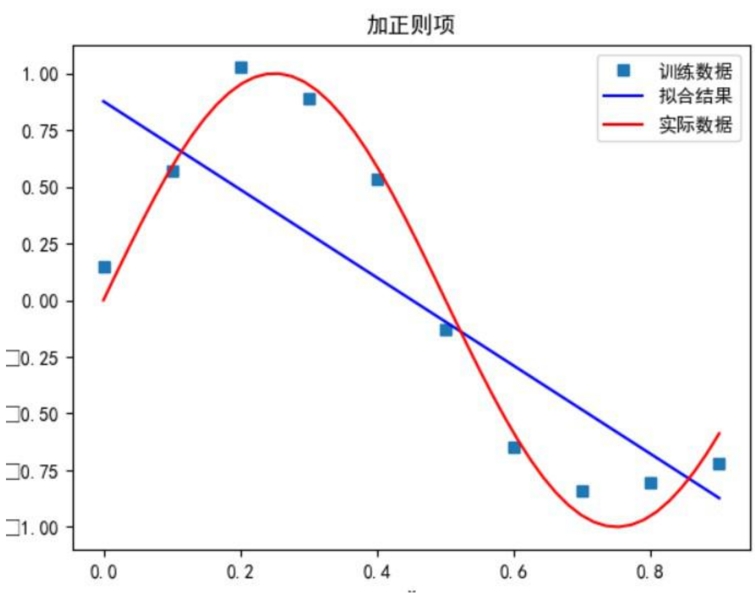
```python
def conjugate_gradient(train_x, train_y, lam, eps):
    X = genMatrixX(train_x)
    Y = train_y.reshape((sample_n, 1))
    Q = np.dot(X.T, X) + lam * np.eye(X.shape[1])
    w = np.zeros((poly_degree + 1, 1))
    gradient = np.dot(X.T, X).dot(w) - np.dot(X.T, Y) + lam * w
    r = -gradient
    p = r
    for i in range(poly_degree + 1):
        a = (r.T.dot(r)) / (p.T.dot(Q).dot(p))
        r_prev = r
        w = w + a * p
        r = r - (a * Q).dot(p)
        beta= (r.T.dot(r)) / (r_prev.T.dot(r_prev))
        p = r + beta * p
    poly = np.poly1d(w[::-1].reshape(poly_degree + 1))
    return poly
```

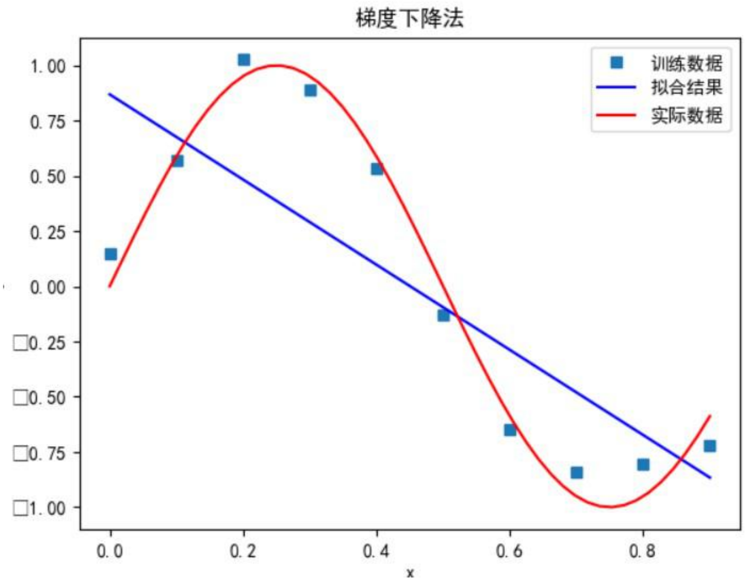
5）用你得到的实验数据，解释过拟合。

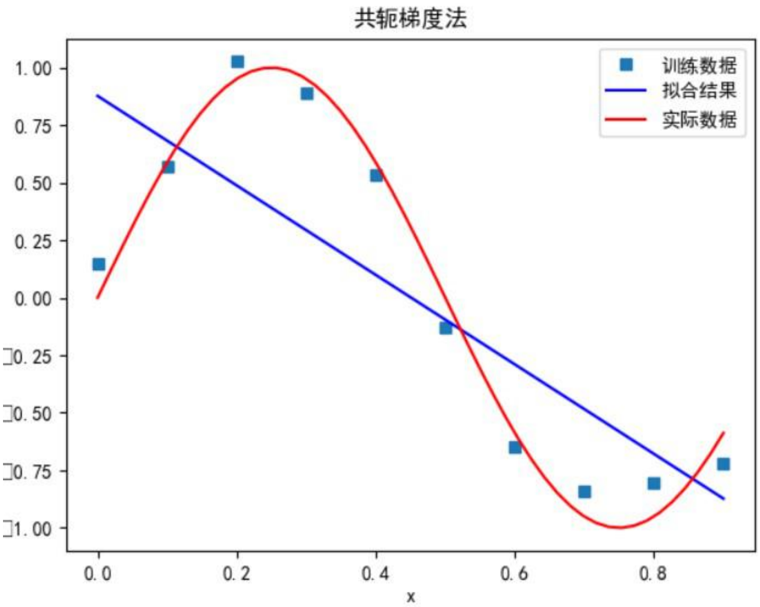
多项式次数为 1 时：


多项式拟合结果
训练数据
拟合结果
实际数据
x

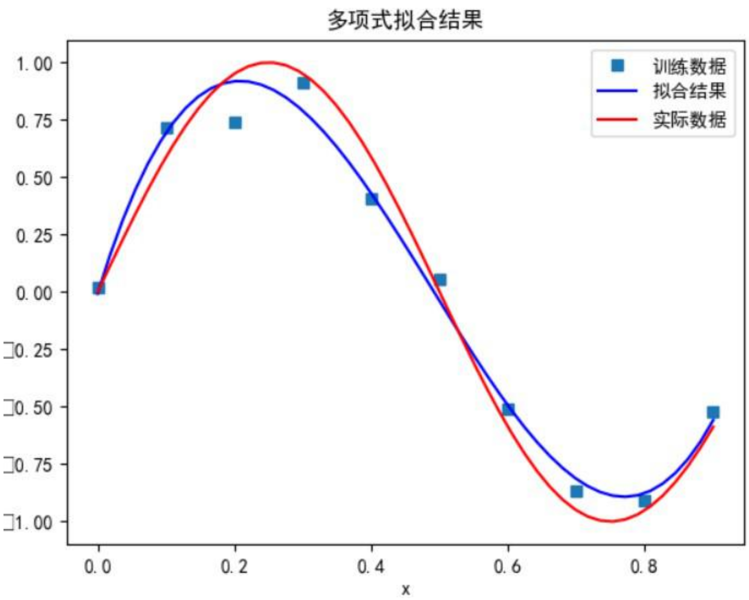

不加正则项
训练数据
拟合结果
实际数据
x

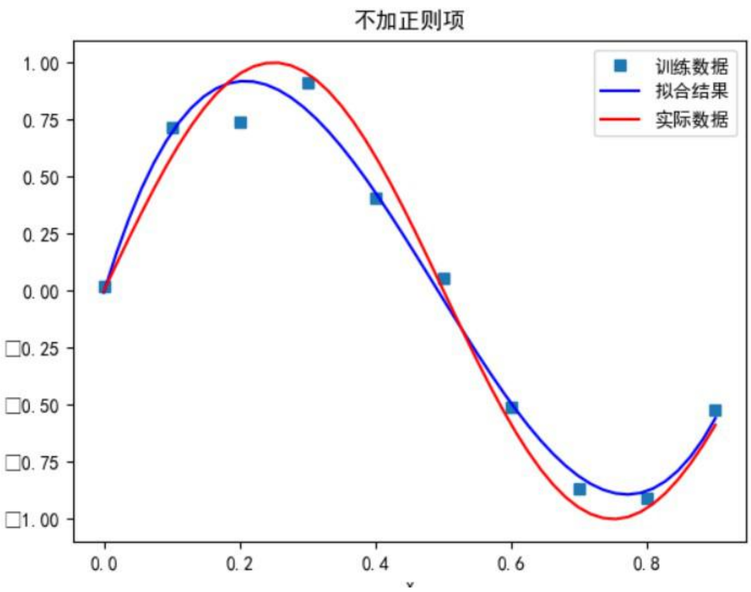

加正则项
训练数据
拟合结果
实际数据

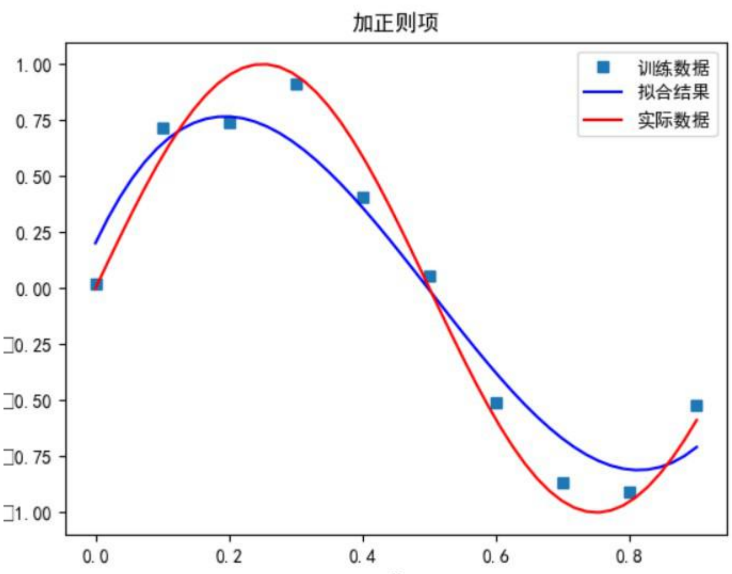

梯度下降法
训练数据
拟合结果
实际数据
1.00
0.75
0.50
0.25
0.00
0.25
0.50
0.75
1.00
0.0
0.2
0.4
0.6
0.8
x

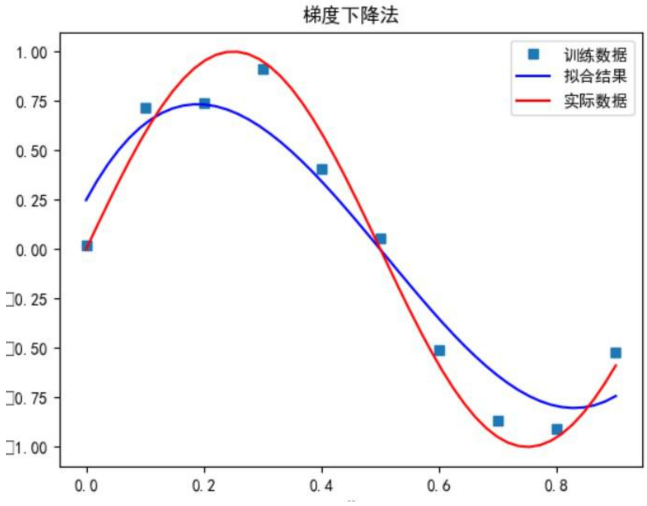

共轭梯度法
训练数据
拟合结果
实际数据
1.00
0.75
0.50
0.25
0.00
0.25
0.50
0.75
1.00
0.0
0.2
0.4
0.6
0.8
x

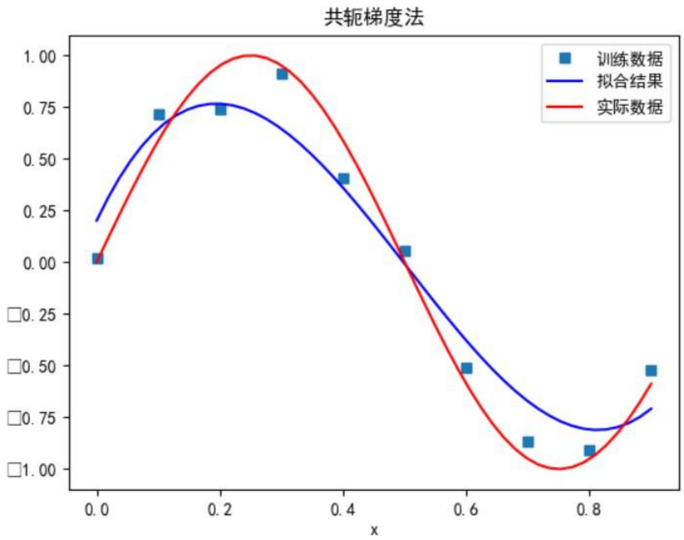
多项式次数为 3 时:


多项式拟合结果
训练数据
拟合结果
实际数据
x

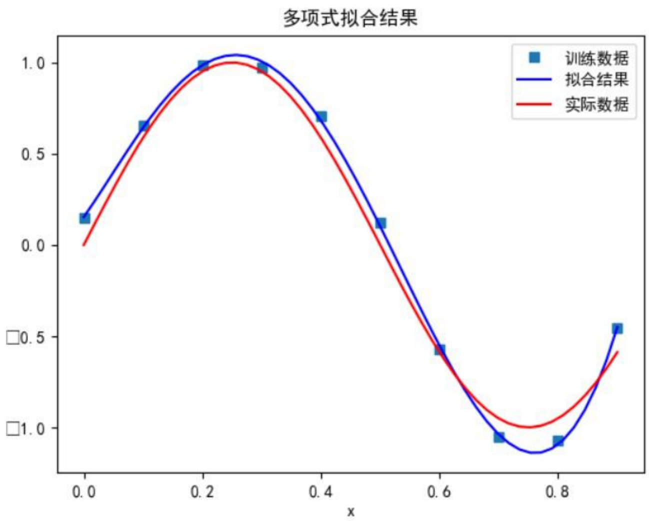

不加正则项
训练数据
拟合结果
实际数据

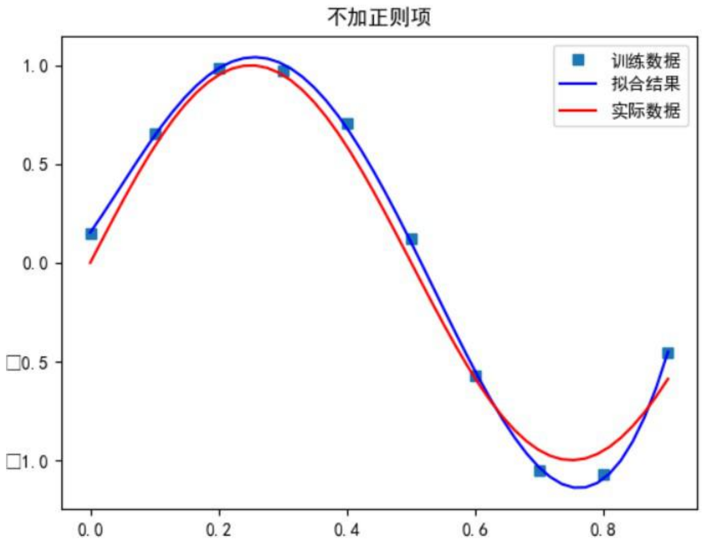

加正则项
训练数据
拟合结果
实际数据

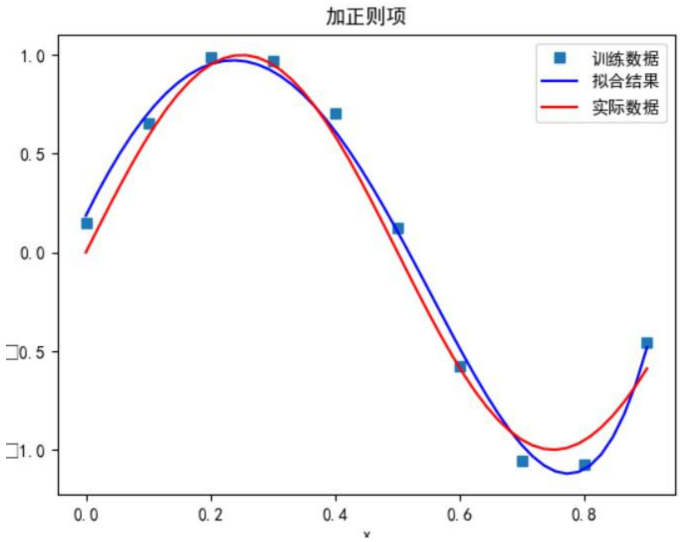

梯度下降法
训练数据
拟合结果
实际数据
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8

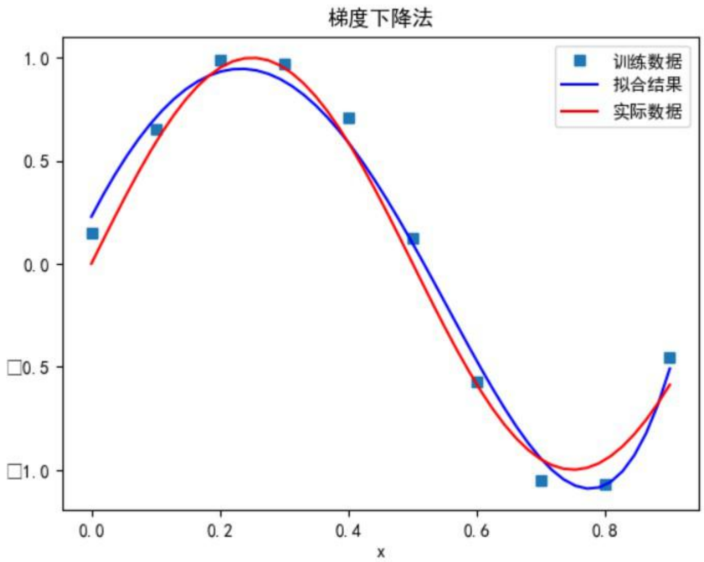

共轭梯度法
训练数据
拟合结果
实际数据
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
x

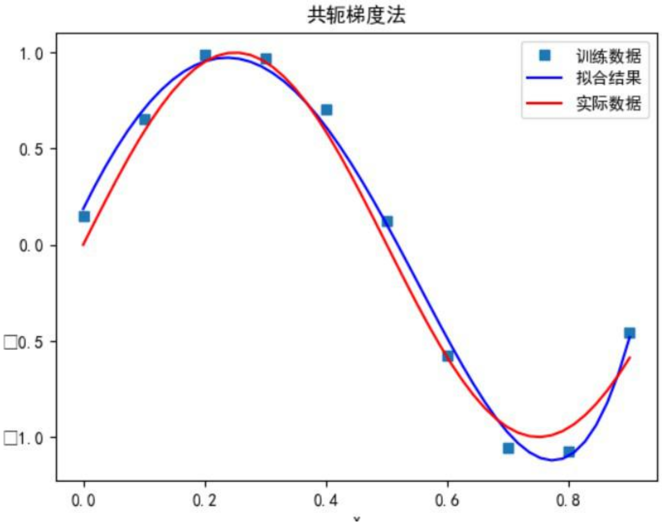
当多项式次数为 5 时：


多项式拟合结果
训练数据
拟合结果
实际数据
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
x

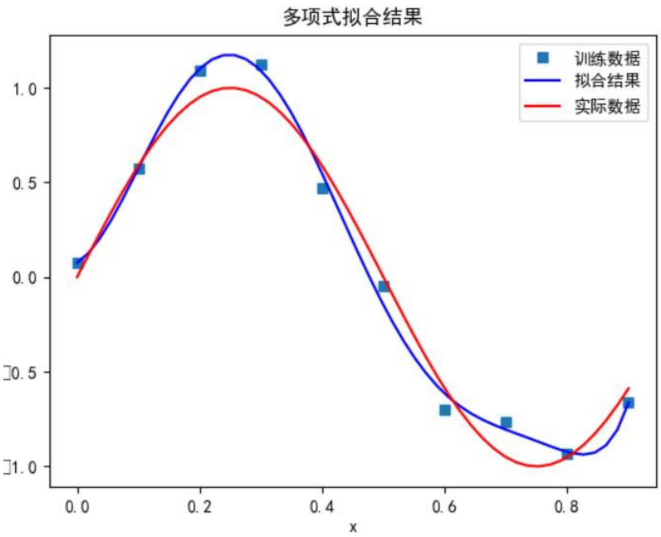

不加正则项
训练数据
拟合结果
实际数据

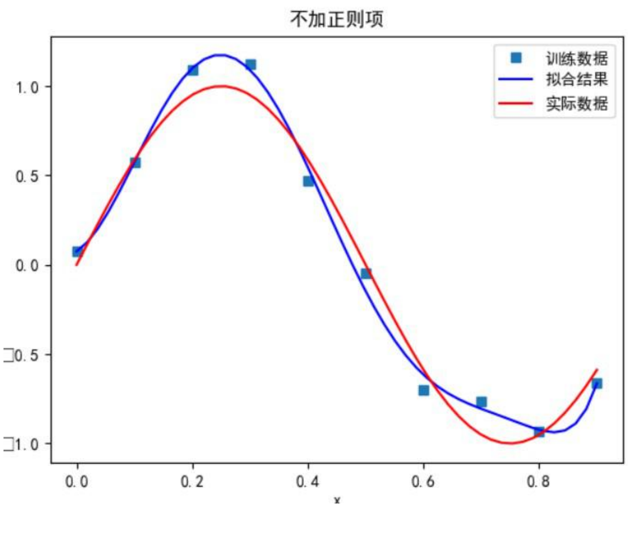

加正则项
训练数据
拟合结果
实际数据
x

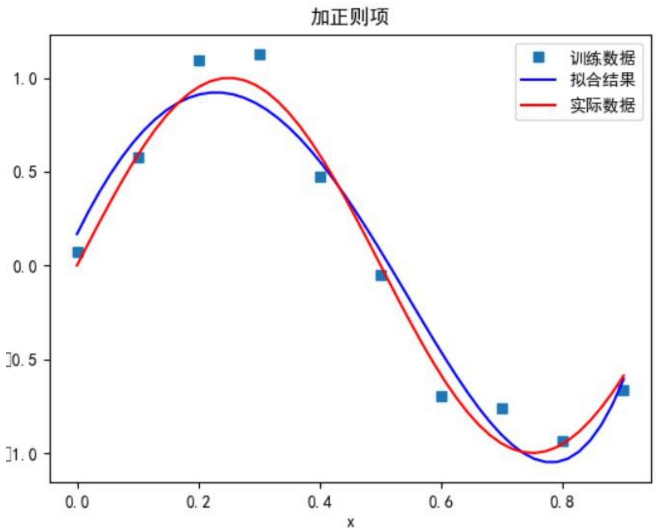

梯度下降法
训练数据
拟合结果
实际数据
x

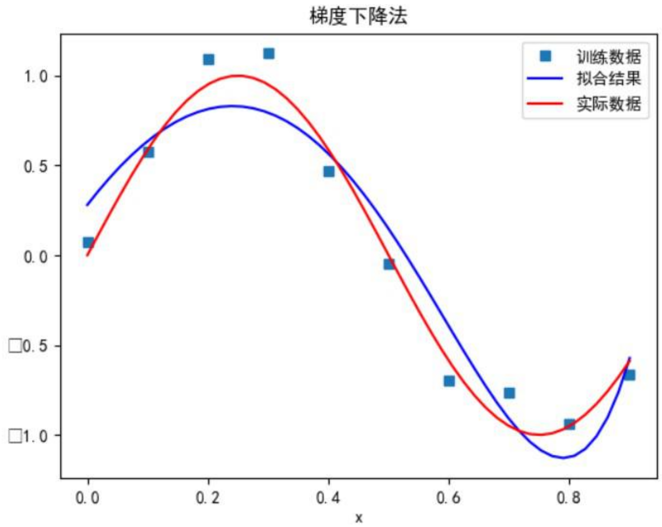

共轭梯度法
训练数据
拟合结果
实际数据
x

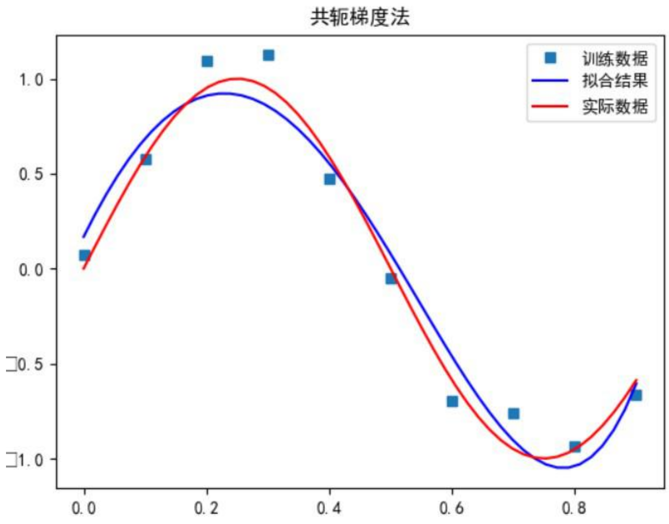
当多项式次数为 7 时：


多项式拟合结果
训练数据
拟合结果
实际数据
x

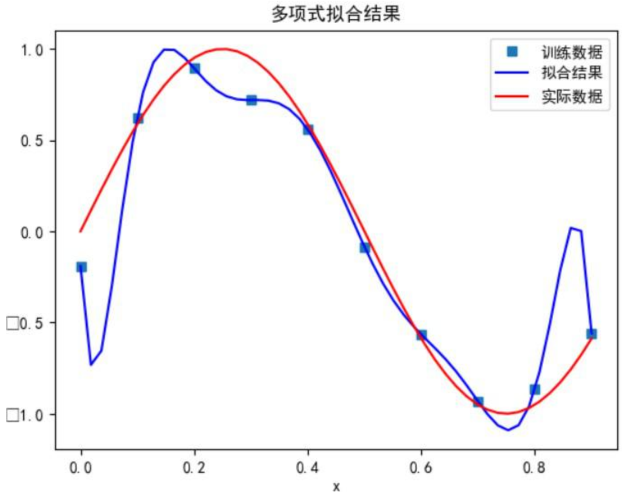

不加正则项
训练数据
拟合结果
实际数据
x

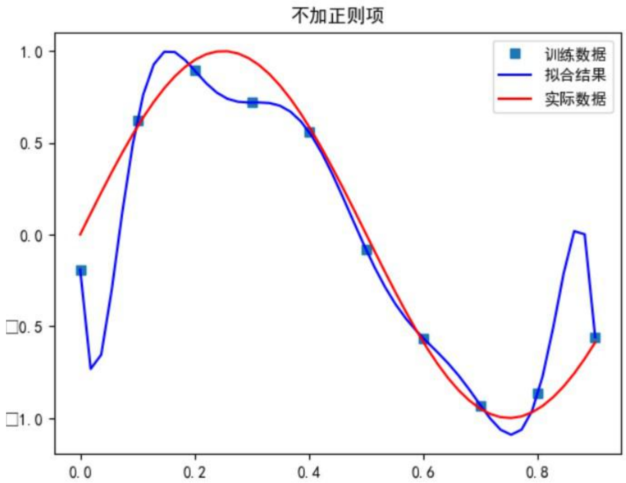

加正则项
训练数据
拟合结果
实际数据
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
x

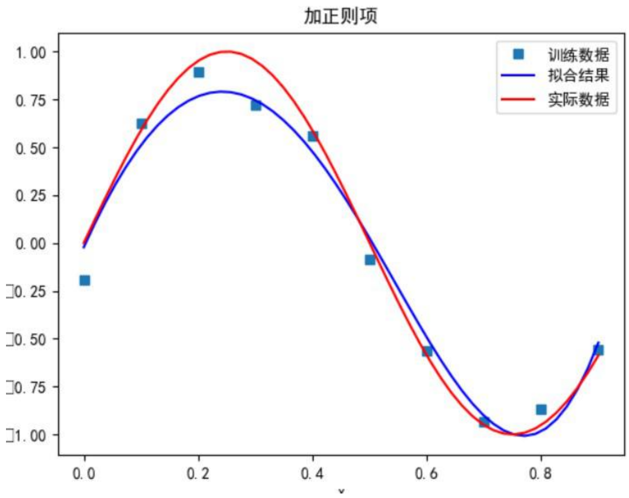

梯度下降法
训练数据
拟合结果
实际数据
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
x

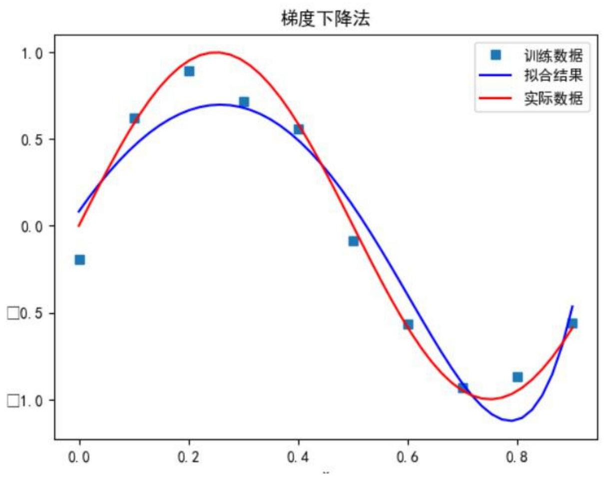

共轭梯度法
训练数据
拟合结果
实际数据
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8

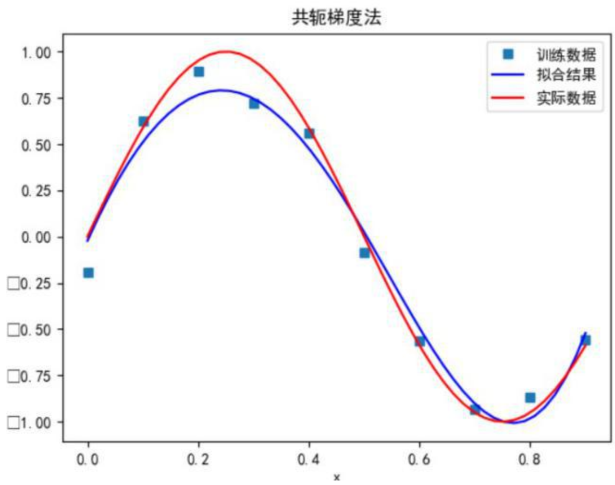
当多项式次数为 9 时：


多项式拟合结果
训练数据
拟合结果
实际数据
x


不加正则项
训练数据
拟合结果
实际数据

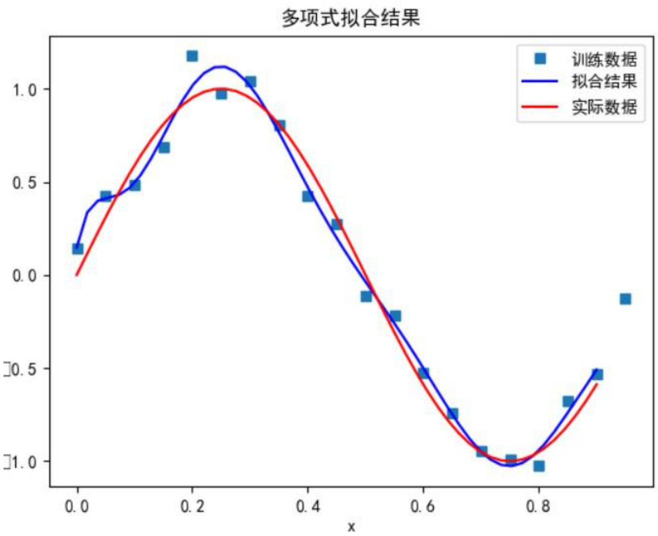

加正则项
训练数据
拟合结果
实际数据

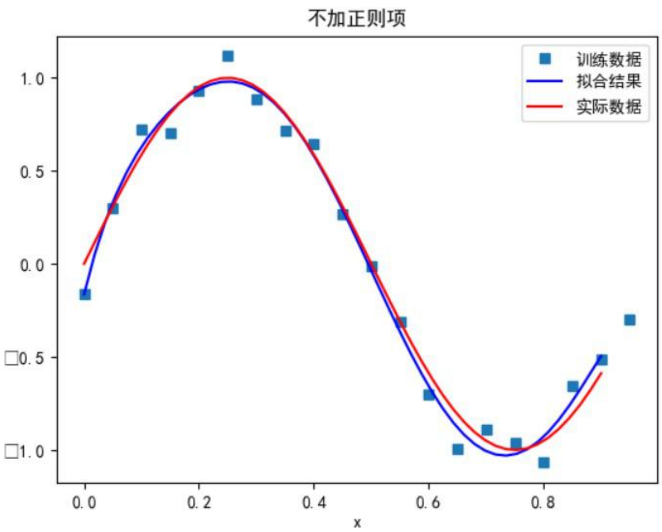


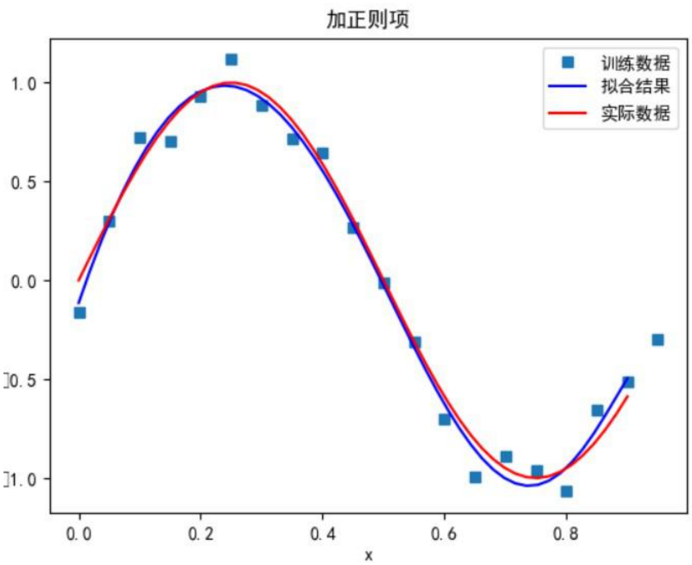


可以看到，多项式次数并不是越高拟合的效果就越好。
次数太小时无法正确拟合，次数在 5 时拟合的效果比较好，但再之后多项式次数过大时拟合的效果反而会变差，也就是出现了过拟合的情形。过拟合是由于样本数量过少，模型能力过强导致的。此时如果没有采用加入正则项等优化处理时拟合效果会非常差。

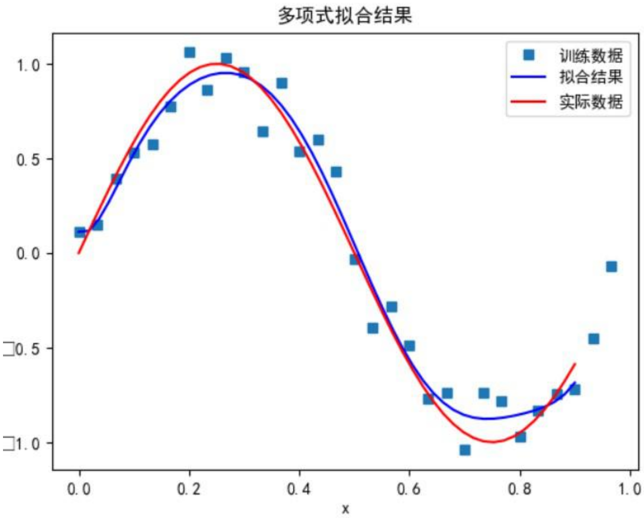
6）用不同数据量，不同超参数，不同的多项式阶数，比较实验效果。
数据量为 20，多项式次数为 9，超参数为 0.0001 时：

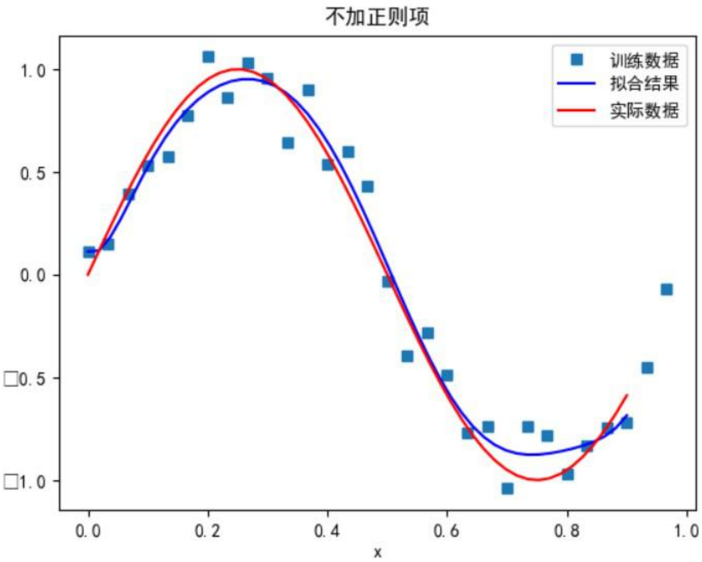

多项式拟合结果
训练数据
拟合结果
实际数据
1.0
0.5
0.0
0.5
1.0
0.0
0.2
0.4
0.6
0.8
x

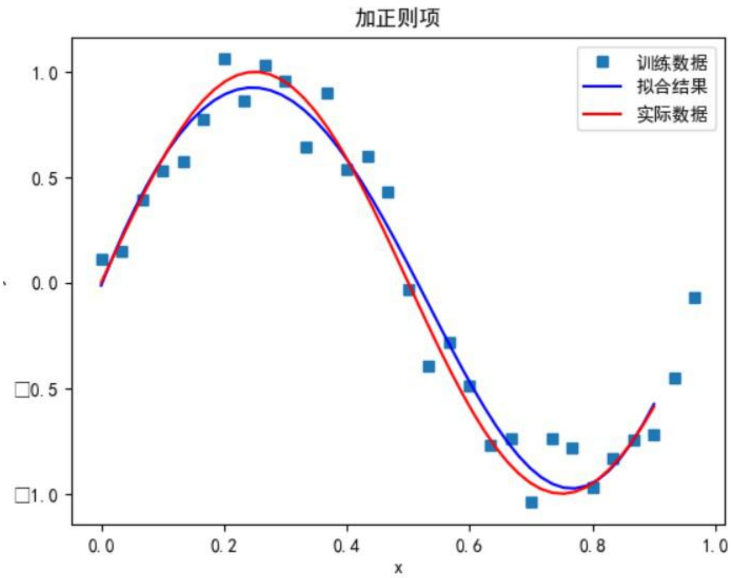

不加正则项
训练数据
拟合结果
实际数据
1.0
0.5
0.0
0.5
1.0
0.0
0.2
0.4
0.6
0.8
x


加正则项
训练数据
拟合结果
实际数据
1.0
0.5
0.0
0.5
1.0
0.0
0.2
0.4
0.6
0.8
x

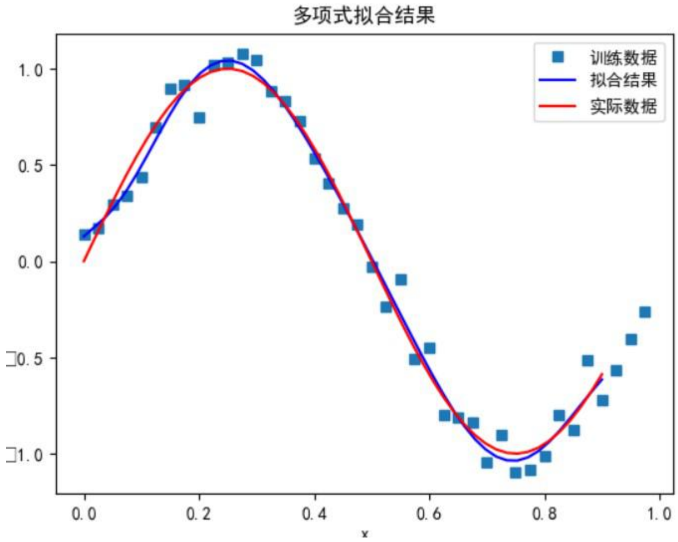
数据量为 30，多项式次数为 9，超参数为 0.0001 时：

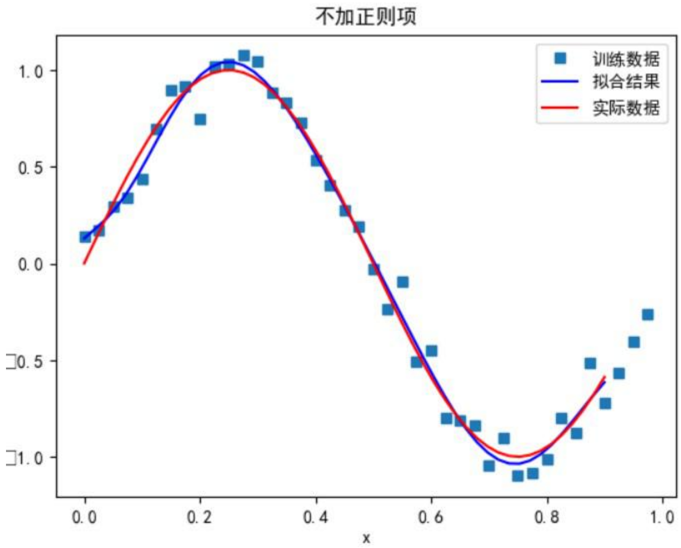


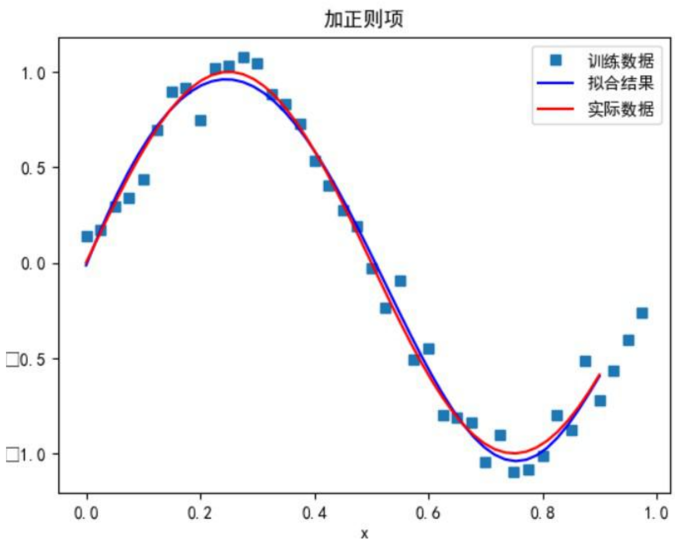




数据量为 40，多项式次数为 9，超参数为 0.0001 时：

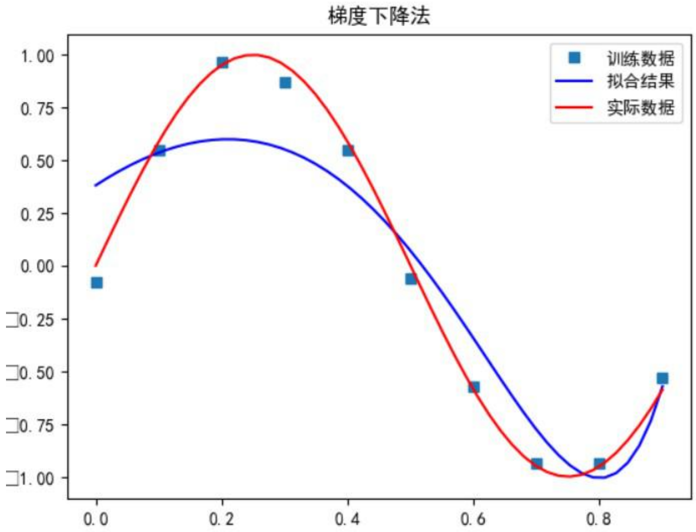




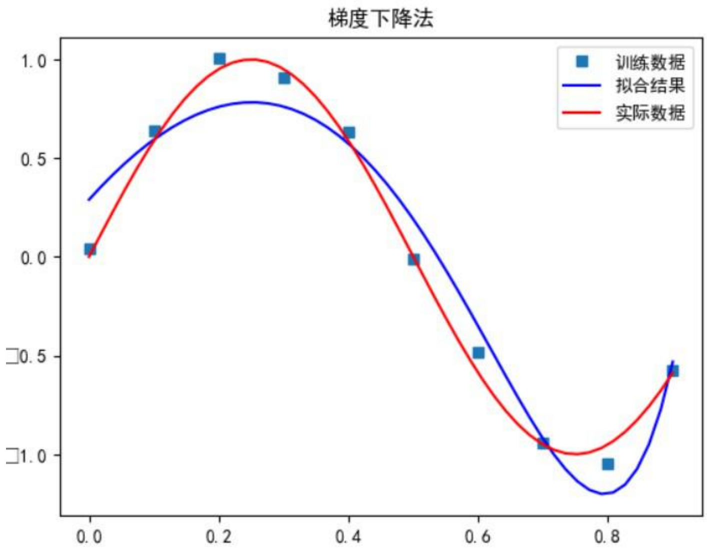


可以看到，相同条件下样本数据越多，拟合效果越好

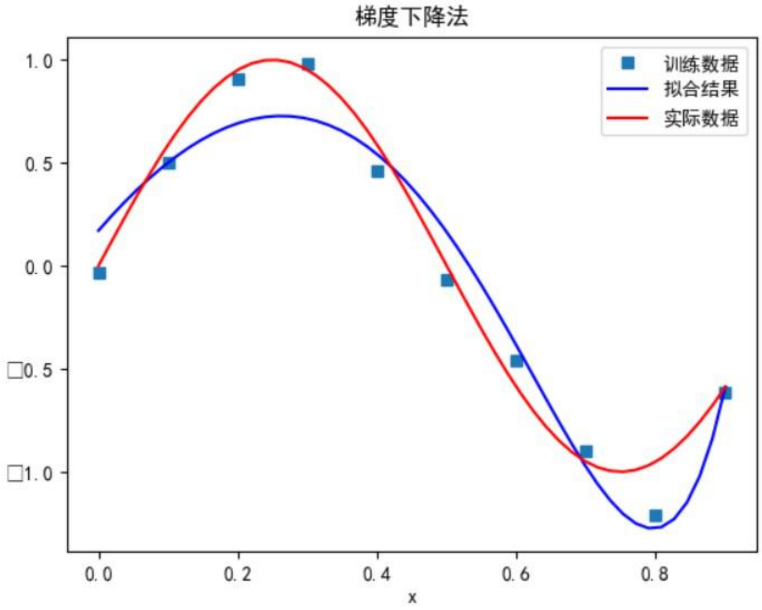
数据量为 40，多项式次数为 9，超参数为 0.01 时：



数据量为 40，多项式次数为 9，超参数为 0.001 时：



数据量为 40，多项式次数为 9，超参数为 0.0001 时：



可以看到超参数对拟合效果影响很有限