2020 年春季学期
计算学部《机器学习》课程

Lab 1 实验报告

姓名	包宇昊
学号	1180300605
班号	1803105
电子邮件	2568235796@qq.com
手机号码	18267116288

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

1 .实验目标

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)

2 .实验要求及环境

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。

实验环境：

pycharm 2020.2.2

python 3.7

win10

X86-64

3 .数学思想

定义 Loss：$\mathrm{L}(\omega) = \frac{1}{2N}\sum_{\omega}(y - X\omega)^2$

其中：

$$X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1d} & 1 \\ x_{21} & x_{22} & \cdots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{md} & 1 \end{pmatrix} = \begin{pmatrix} x_1^T & 1 \\ x_2^T & 1 \\ \vdots & \vdots \\ x_m^T & 1 \end{pmatrix}$$

求解ω得：

$$\hat{\omega}^* = \underset{\hat{\omega}}{argmin}(y - X\hat{\omega})^T(y - X\hat{\omega})$$

令$E(\omega) = (y - X\omega)^T(y - X\omega)$,对ω求导得

$$\frac{\partial E_{\hat{\omega}}}{\partial \hat{\omega}} = \frac{1}{N}X^T(X\hat{\omega} - y)$$

为了让 Loss 最小，即让$E(\omega)$最小，求导数为 0 时，得到下式：

$$(X^TX)\omega = X^Ty$$

对该方程解析解的不同解法产生了几种不同的算法。

X^TX可逆的情况下：

$$\omega = (X^TX)^{-1}X^Ty$$

而为了避免过拟合，可以在损失函数中加入惩罚项，比例因子为 λ。

加入惩罚后的 Loss 为

$$L(\omega) = \frac{1}{2N}\sum_{\omega}(y - X\omega)^2 + \frac{\lambda}{2}||\omega||_2$$

$X^TX + \lambda NI$一定可逆，解得：

$$\omega = (X^TX + \lambda NI)^{-1}X^Ty$$

4 .算法实现

生成数据：

```
def data(size):
    x = np.linspace(0, 1, num=size)
    return x.reshape(size, 1)
```

加入噪声：

```
def create_data(size):
    x = data(size)
    np.random.seed(50)
    y = sin_func(x) + np.random.normal(scale=0.1, size=x.shape)  # 加入噪声
    return x, y
```

填充矩阵：

```
def create_matrix10(d):  # 构造x的10个样本的d阶多项式的矩阵
    X = np.empty([10, d + 1])
    for row in range(0, 10):
        X[row, d] = 1
        for col in range(d):
            # a = np.logspace(1, 10, base=x_train[row])
            X[row, col] = np.power(x_train[row],col+1)
    return X
```

画图：

```
plt.scatter(x_train,y_train,facecolor="none",edgecolor="green",s=30,label="training data")
plt.plot(x_test,y_test,c="b",label="sin(2πx)")
plt.xlabel("x",size=20)
plt.ylabel("y",size=20)
plt.legend()
plt.show()
```

4.1 无惩罚项的最小二乘法

直接通过$\omega = (X^T X)^{-1} X^T y$进行求解。

```
X = create_matrix10(d)
X1 = create_matrix100(d)
y = y_train.reshape((10,1))
S1 = np.dot(X.T,X)
S2 = np.dot(X.T,y)
w = np.linalg.solve(S1,S2)
Y = np.dot(X1,w)
```

4.2 加入惩罚项的最小二乘法

先加入噪声，再通过$\omega = (X^T X + \lambda N I)^{-1} X^T y$ 进行求解

```
X = create_matrix10(d)
X1 = create_matrix100(d)
y = y_train.reshape((10,1))
S1 = np.dot(X.T,X)+10*lemda*np.eye(d+1)
S2 = np.dot(X.T,y)
w = np.linalg.solve(S1,S2)
Y = np.dot(X1,w)
```

4.3 梯度下降法

梯度的反方向就是函数减少最快的方向。

以 $\frac{\partial E_{\hat{\omega}}}{\partial \hat{\omega}} = \frac{1}{N} X^T (X\hat{\omega} - y)$ 为梯度，设置学习率λ不断进行逼近，

数学公式为 $$x_1^{(i+1)} = x_1^{(i)} - \eta \cdot \frac{\partial f}{\partial x_1}(\mathbf{x}^{(i)})$$ ，x 在此为ω。

我设置 20000000 次循环，阈值为10^{-7}，达到其中之一条件即跳出。

```
for num in range(1,20000000):
    E1=E
    S1 = np.dot(X,w)-y
    S2 = 1/10*np.dot(X.T,S1)
    S3 = -S1
    E = np.dot(S3.T,S3)
    w = w - a*S2
    if (abs(E1-E) < 1e-7):
        break
```

4.4 共轭梯度法

共轭梯度法主要关注优化方向和优化步长，是每个方向都实现最大程度的逼近，从而减少迭代次数。

计算方法 PPT 中给出：

- 共轭梯度法推算步骤公式(算法)

(1)任取初值$\boldsymbol{x}^{(0)} \in \mathbf{R}^n$;

(2) $\boldsymbol{p}_0=\boldsymbol{r}^{(0)}=\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{(0)}$

(3)对于 k=0,1,2,…,N

$$\alpha_k=\frac{(\boldsymbol{r}^{(k)},\boldsymbol{p}_k)}{(\boldsymbol{A}\boldsymbol{p}_k,\boldsymbol{p}_k)}$$

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\alpha_k\boldsymbol{p}_k$$

$$\boldsymbol{r}^{(k+1)}=\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^{(k+1)}=\boldsymbol{r}^{(k)}-\alpha_k\boldsymbol{A}\boldsymbol{p}_k$$

$$\beta_k=-\frac{(\boldsymbol{r}^{(k+1)},\boldsymbol{A}\boldsymbol{p}_k)}{(\boldsymbol{p}_k,\boldsymbol{A}\boldsymbol{p}_k)}$$

$$\boldsymbol{p}_{k+1}=\boldsymbol{r}^{(k+1)}+\beta_k\boldsymbol{p}_k$$

实现如下：

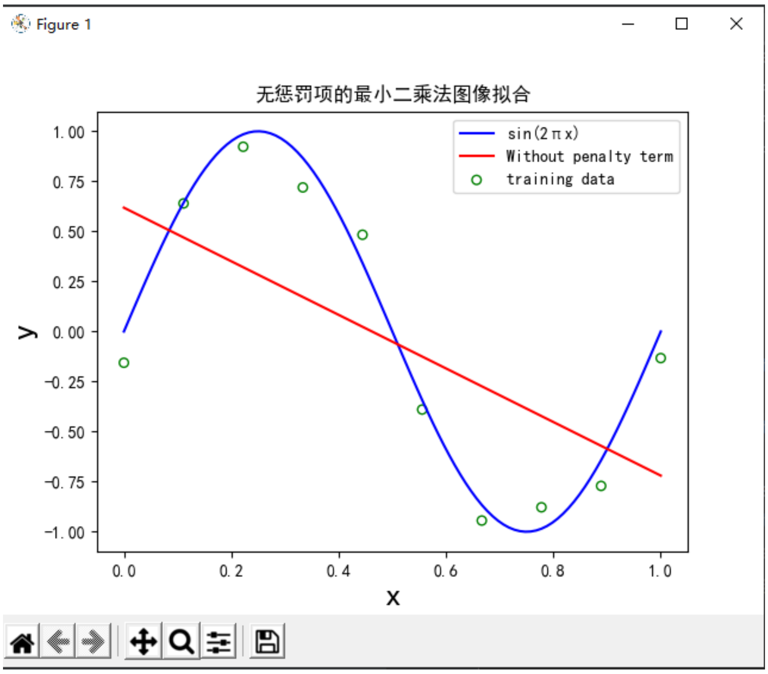
```
X = create_matrix10(d)
X1 = create_matrix100(d)
y = y_train.reshape((10, 1))
A = np.dot(X.T,X)
b = np.dot(X.T,y)
w = np.zeros([d+1,1])
r = b - np.dot(A,w)
p = r
for k in range(0,d):
    S1 = np.dot(r.T,p)
    S2 = np.dot(A,p)
    S3 = np.dot(S2.T,p)
    alpha = S1/S3     #优化步长
    w = w + alpha*p
    r = b - np.dot(A,w)
    S4 = np.dot(A,p)
    S5 = np.dot(r.T,S4)
    S6 = np.dot(p.T,S4)
    beta = -S5/S6
    p = r + beta*p    #优化方向

Y = np.dot(X1,w)
```

5 .拟合效果

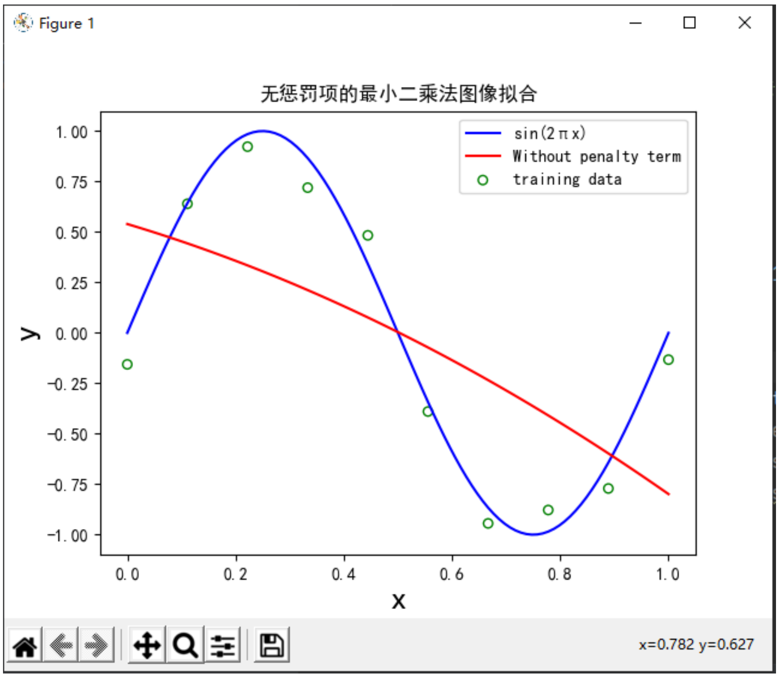
5.1 无惩罚项的最小二乘法

数据量为 10

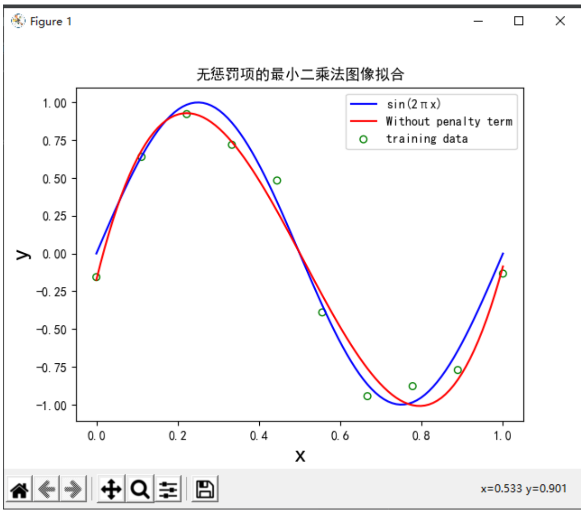
阶数为 1：



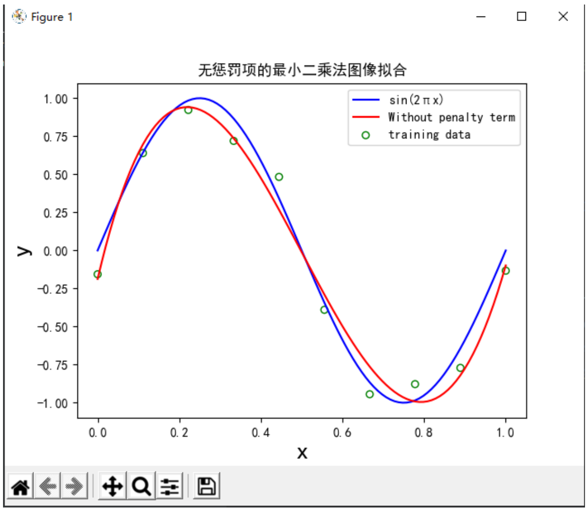
阶数为 2：



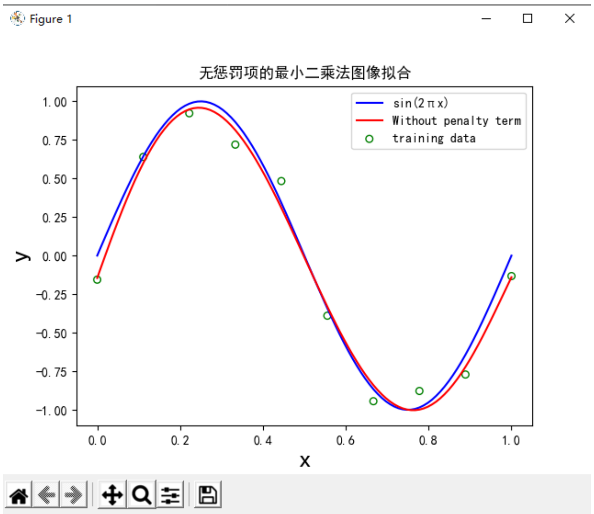
阶数为 3：



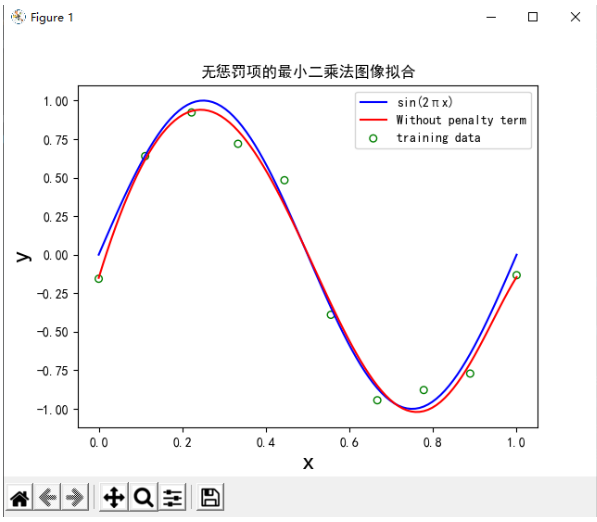
阶数为 4：



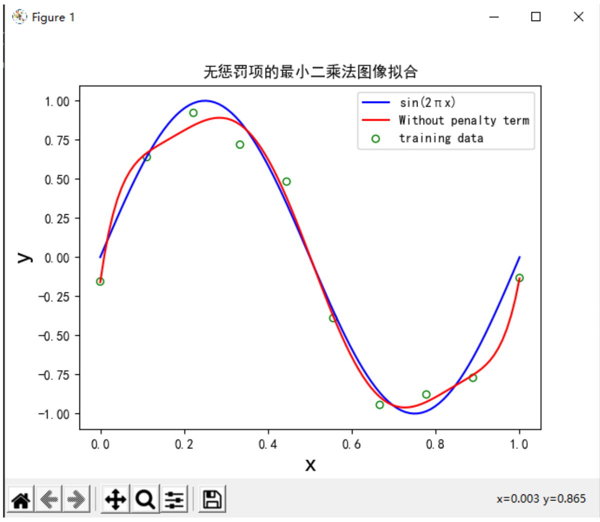
阶数为 5：



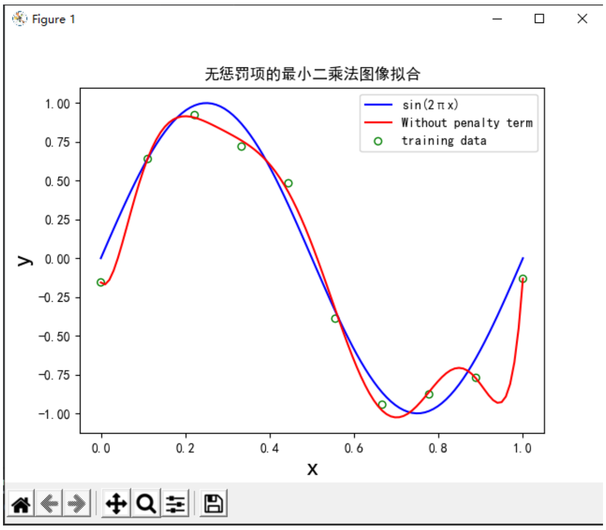
阶数为 6：



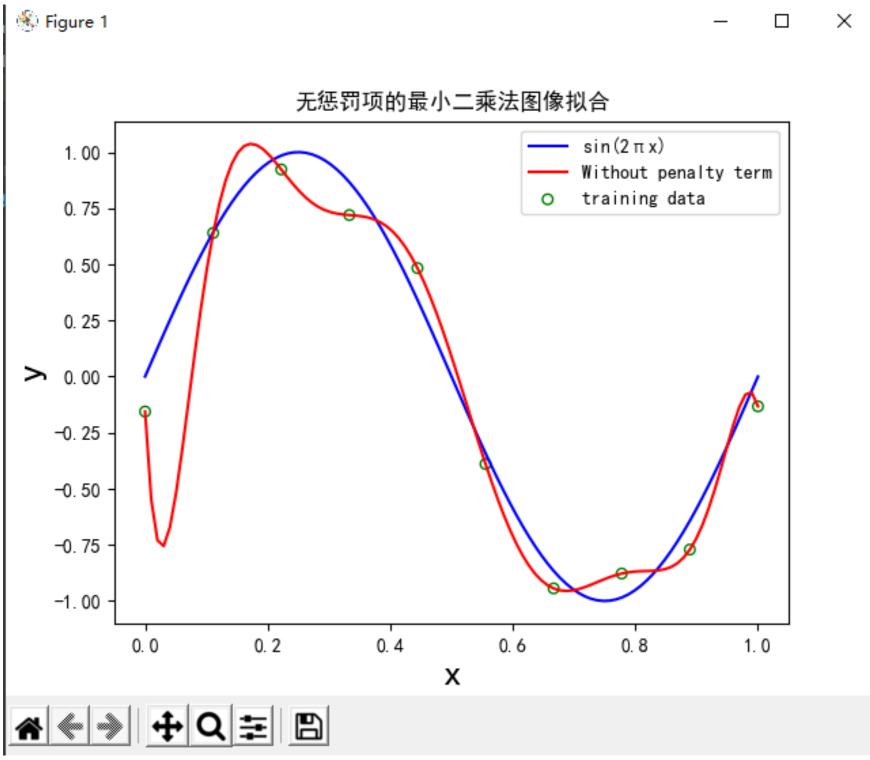
阶数为 7：



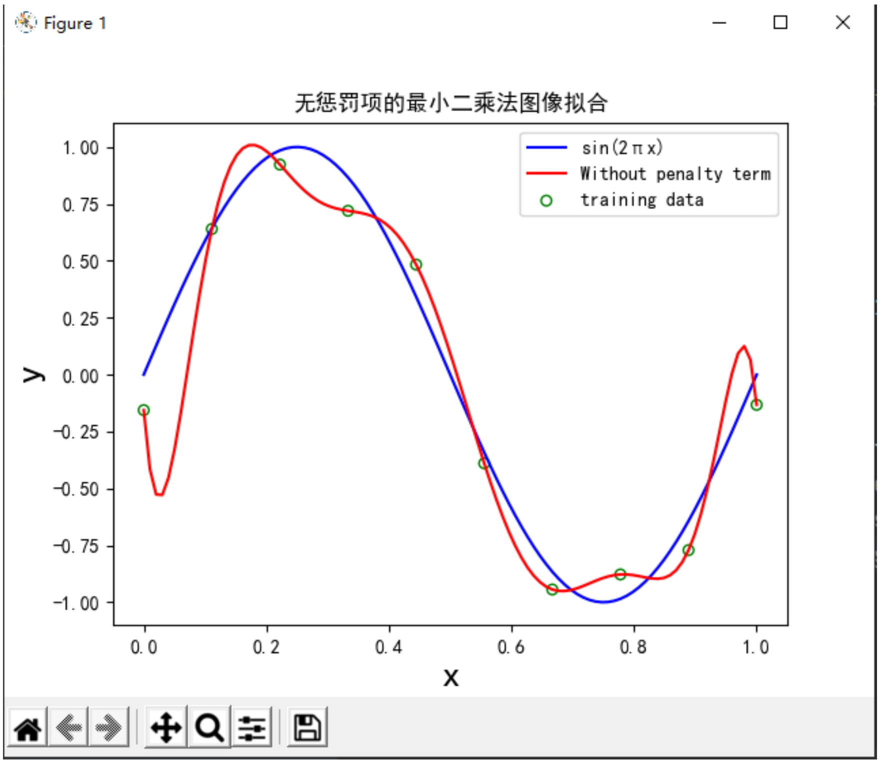
阶数为 8：



阶数为 9：



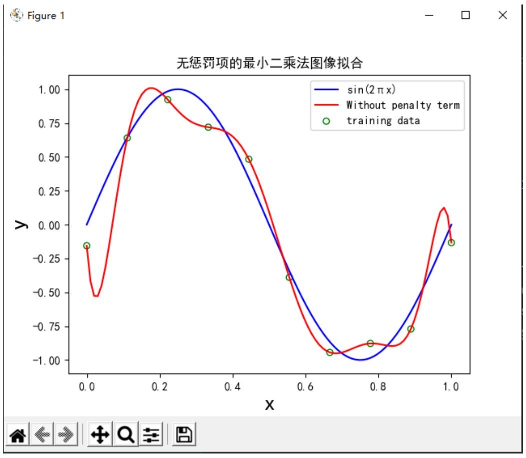
阶数为 10：



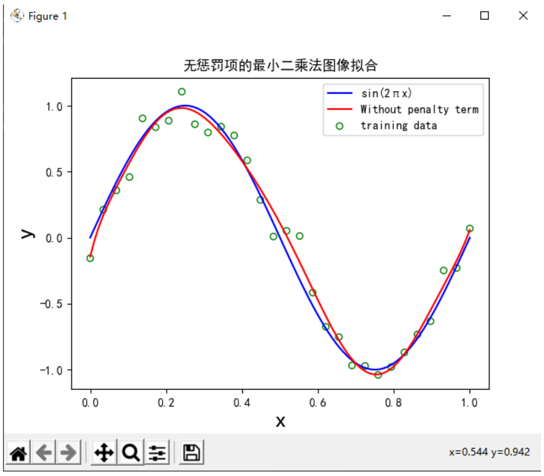
如图，显然：1、2 阶是欠拟合，7 阶以后出现了过拟合

选取阶数为 10 的情况：

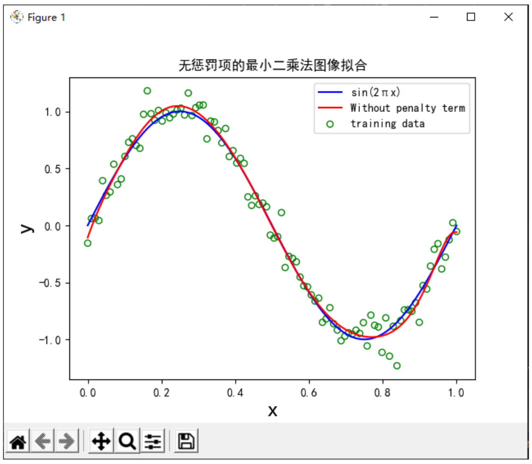
选取数据量 10：



选取数据量 30：



选取数据量 100：

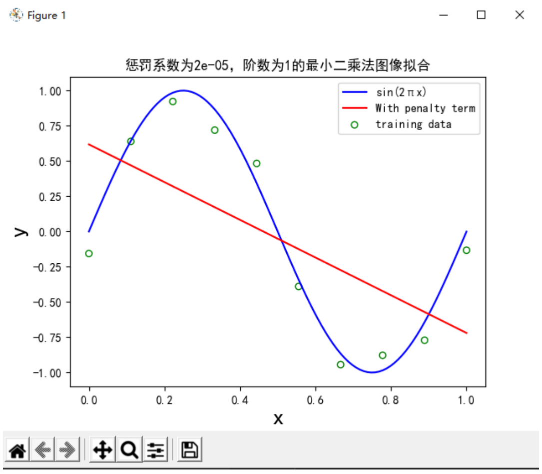


可以看出，数据量小时容易出现过拟合情况，但是误差较小；数据量大时较难出现过拟合情况，但误差有些大

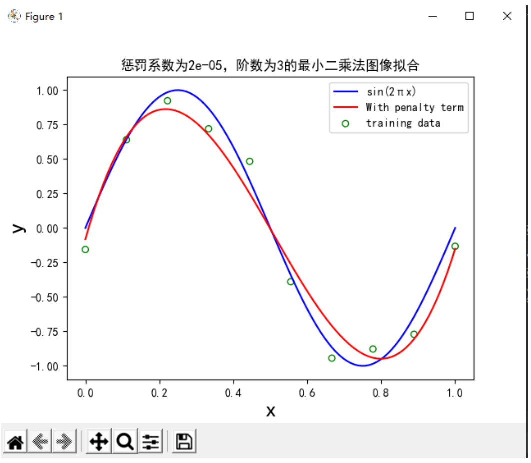
5.2 加入惩罚项的最小二乘法

数据量为 10，λ=0.00002 时：

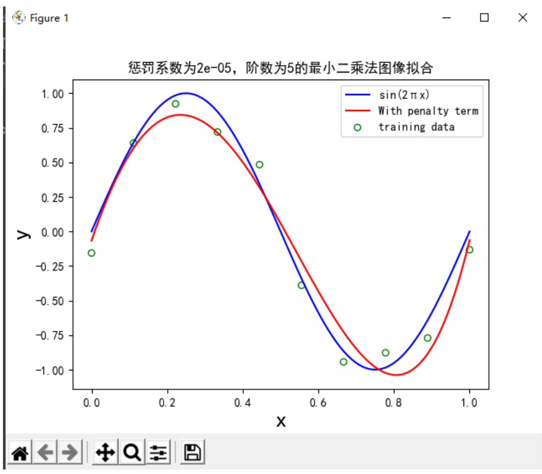
阶数为 1：



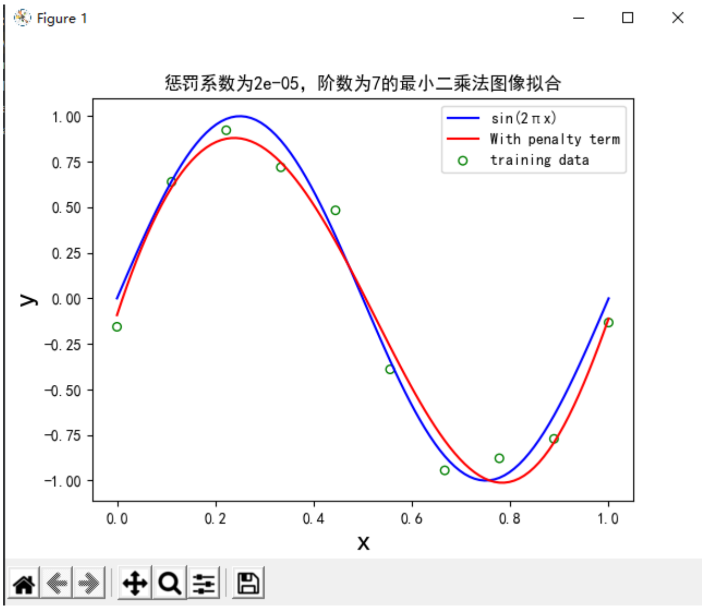
阶数为 3：



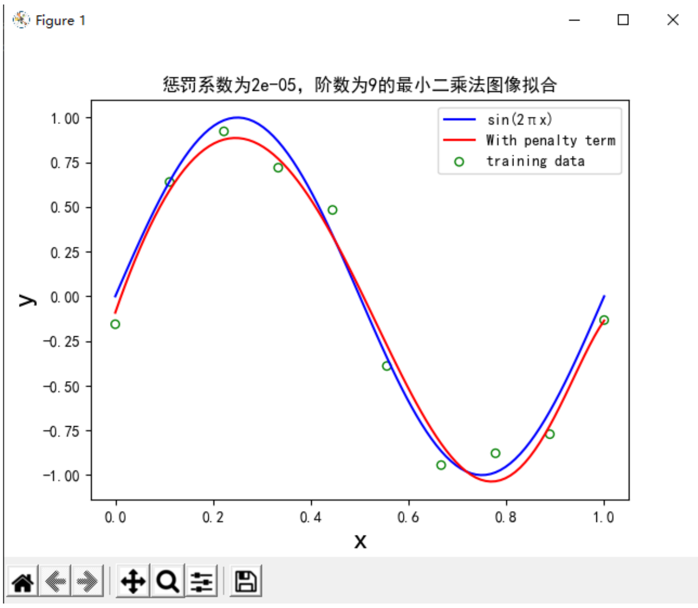
阶数为 5：



阶数为 7：



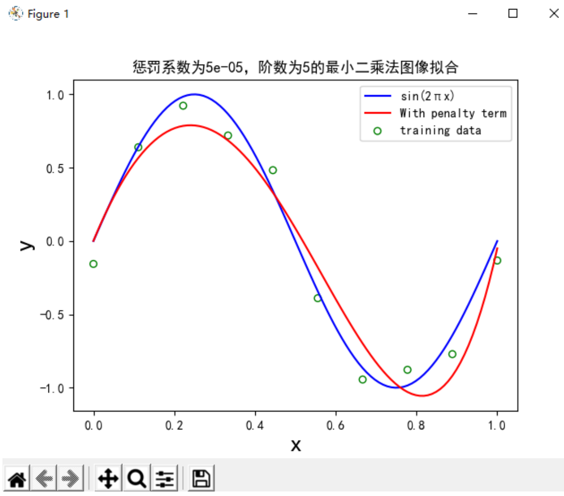
阶数为 9：



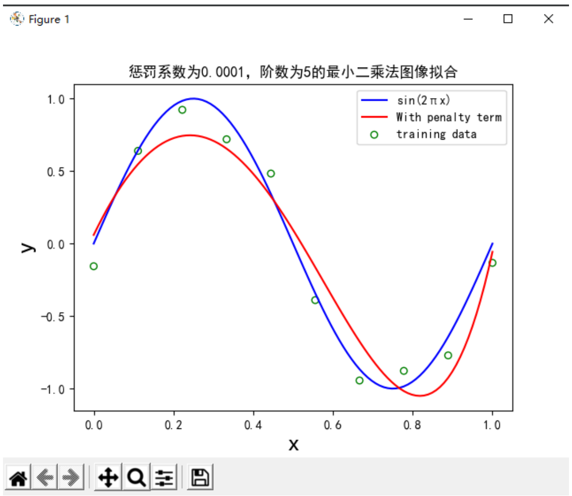
明显可以看出加入惩罚项后，过拟合情况得到改善。

数据量为 10，阶数为 5：

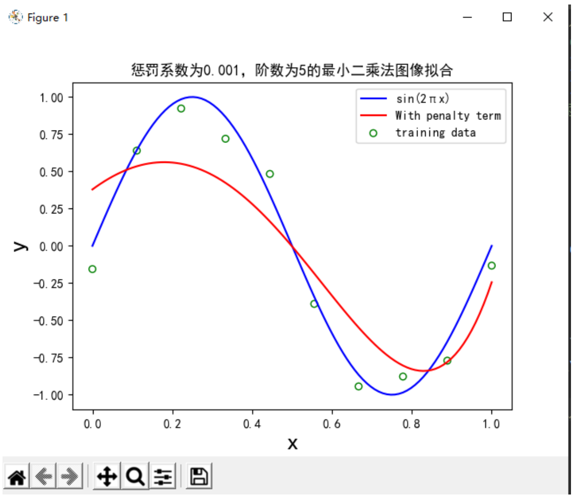
λ=0. 00005：



λ=0. 0001：

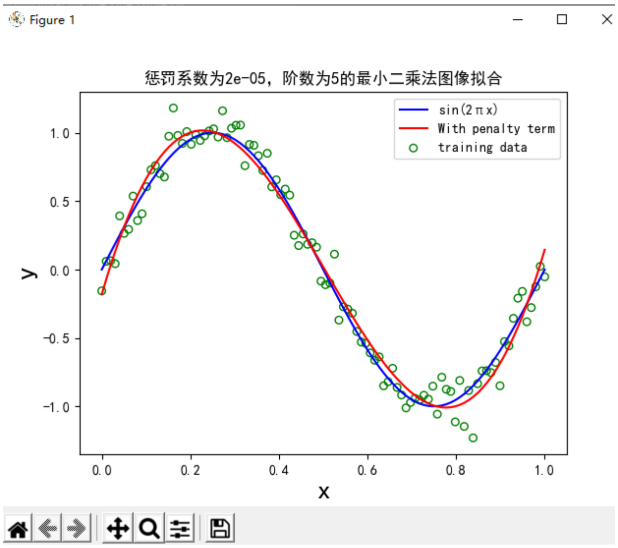


λ=0. 001：



观察得到 λ 变化对过拟合情况有影响，λ 在 0.00001——0.0001 间情况较好。

数据量为 100，阶数为 5，λ 为 0.00002 时：

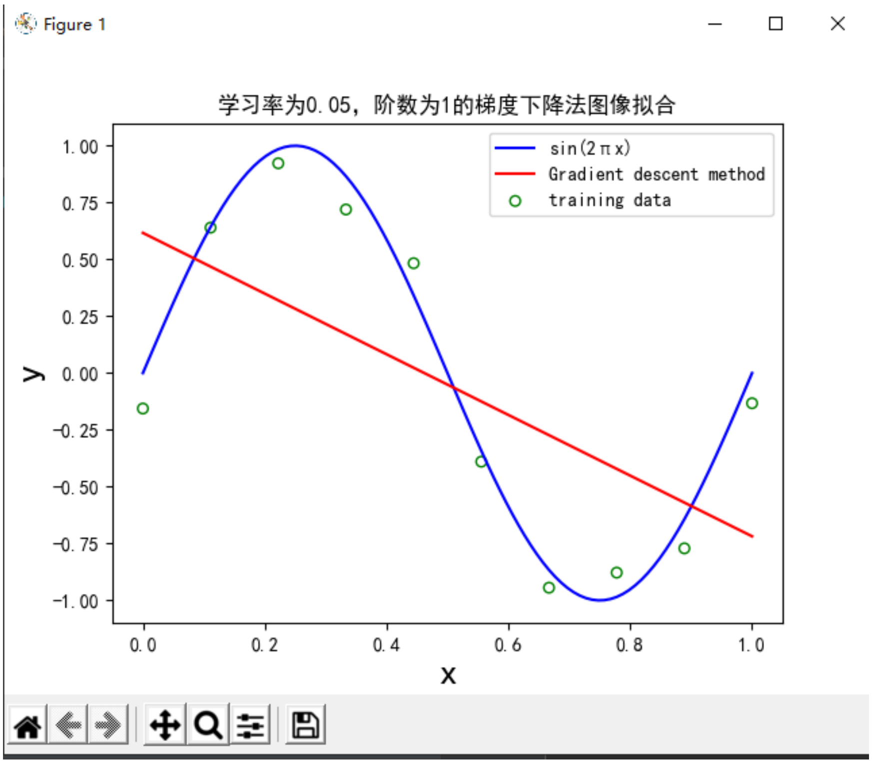


可以看出，随着数据量变大，拟合效果也变好，过拟合情况基本消失。

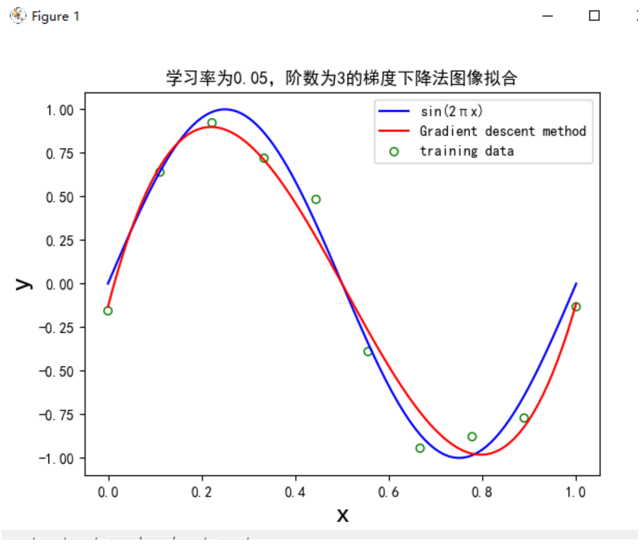
5.3 梯度下降法

数据量为 10，学习率 a=0.05 时：

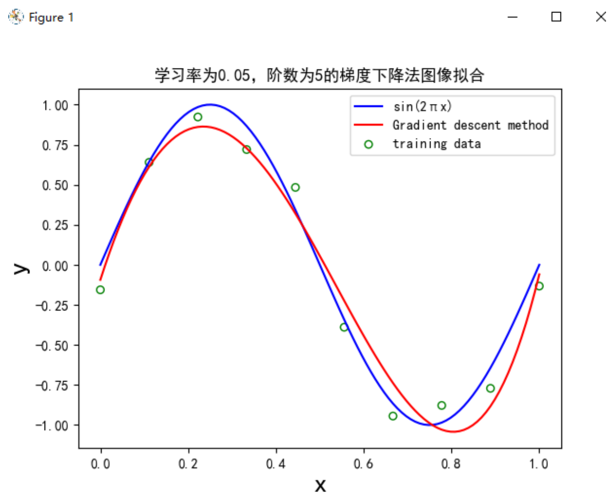
阶数为 1：



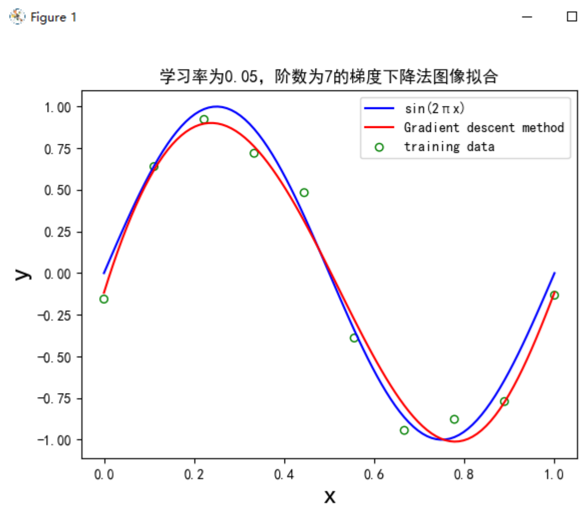
阶数为 3：



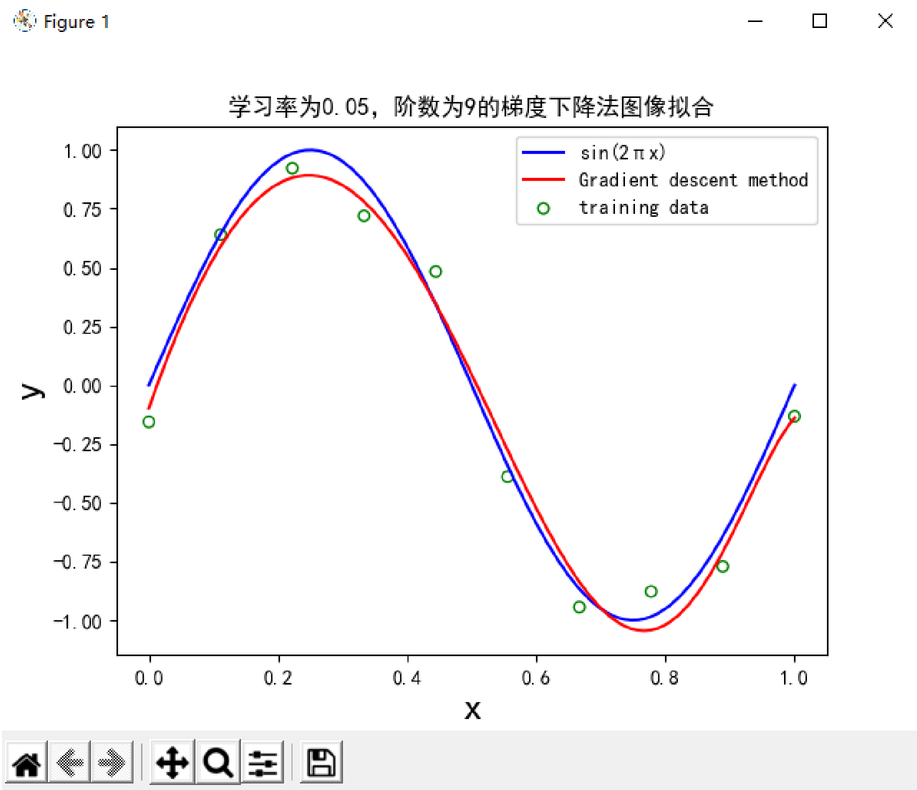
阶数为 5：



阶数为 7：

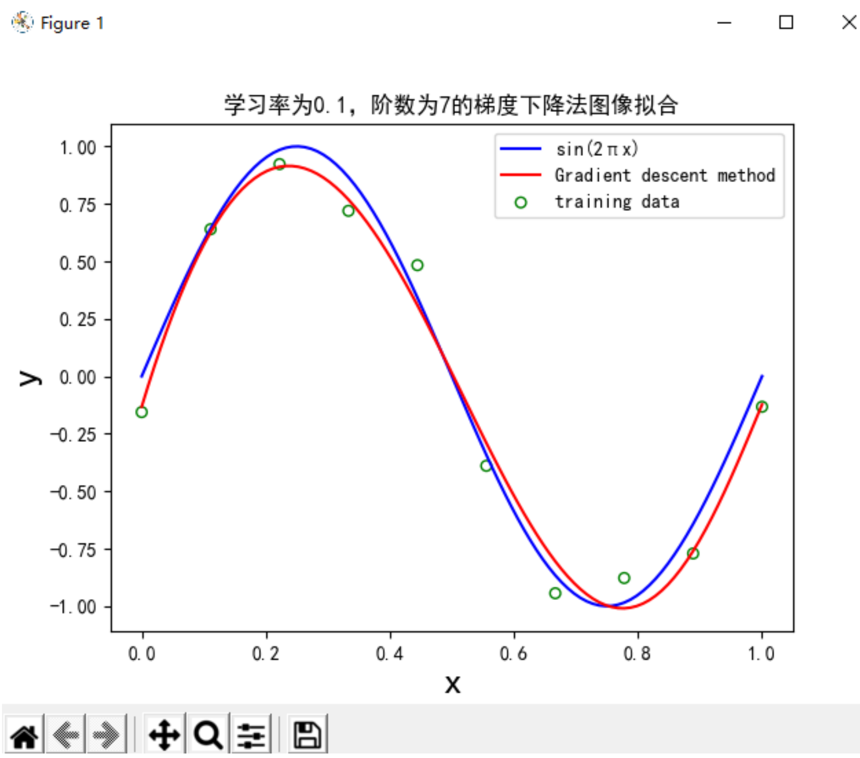


阶数为 9：

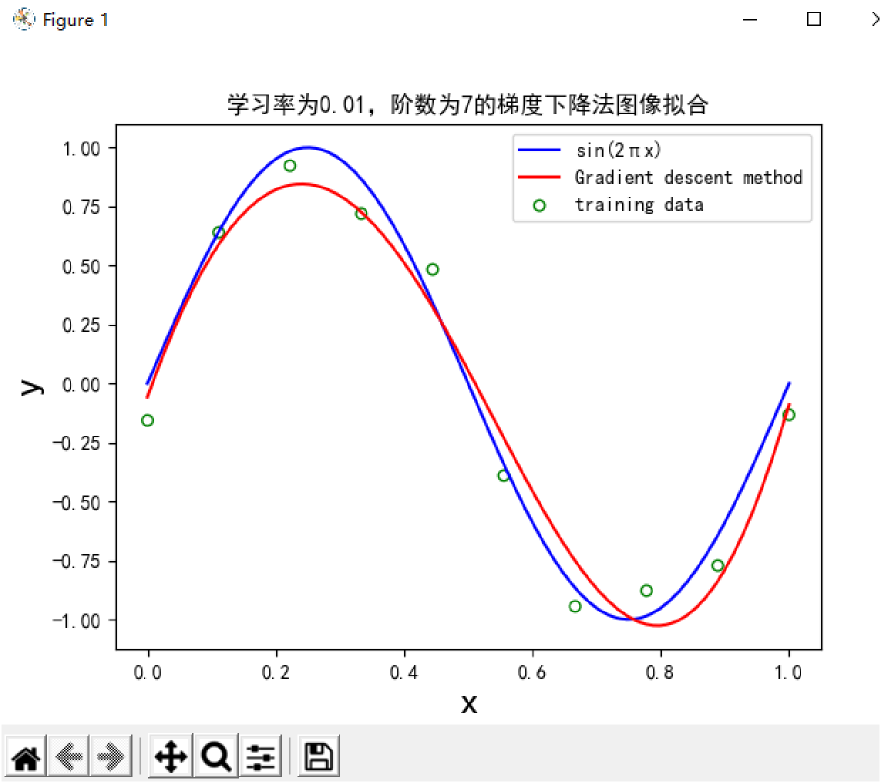


数据量为 10，阶数为 7 时：

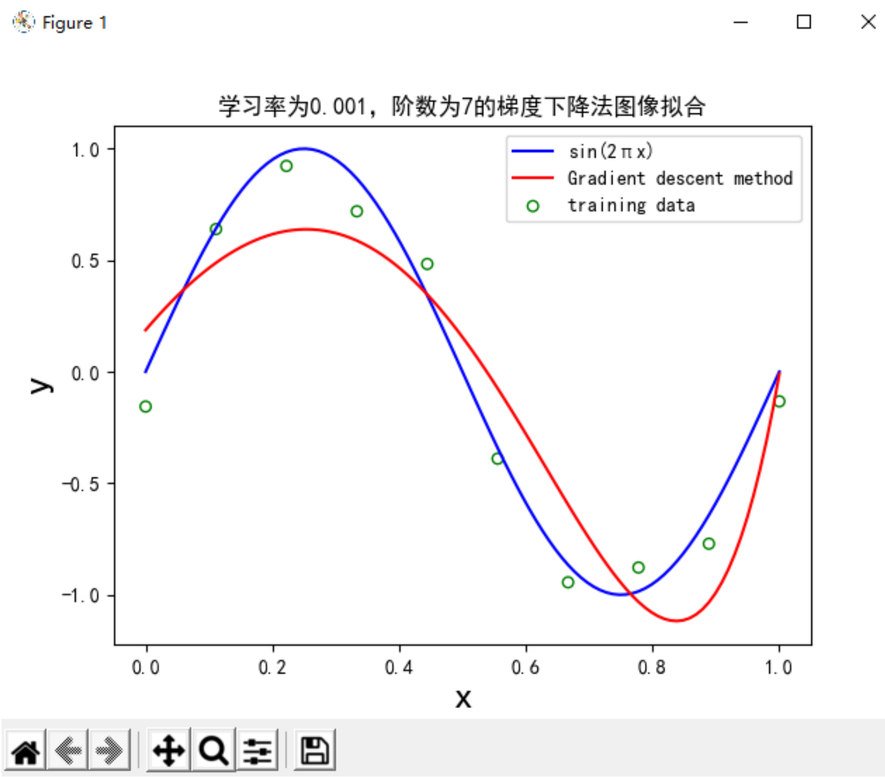
学习率 a=0.1 时：



学习率 a=0.01 时：



学习率 a=0.001 时：

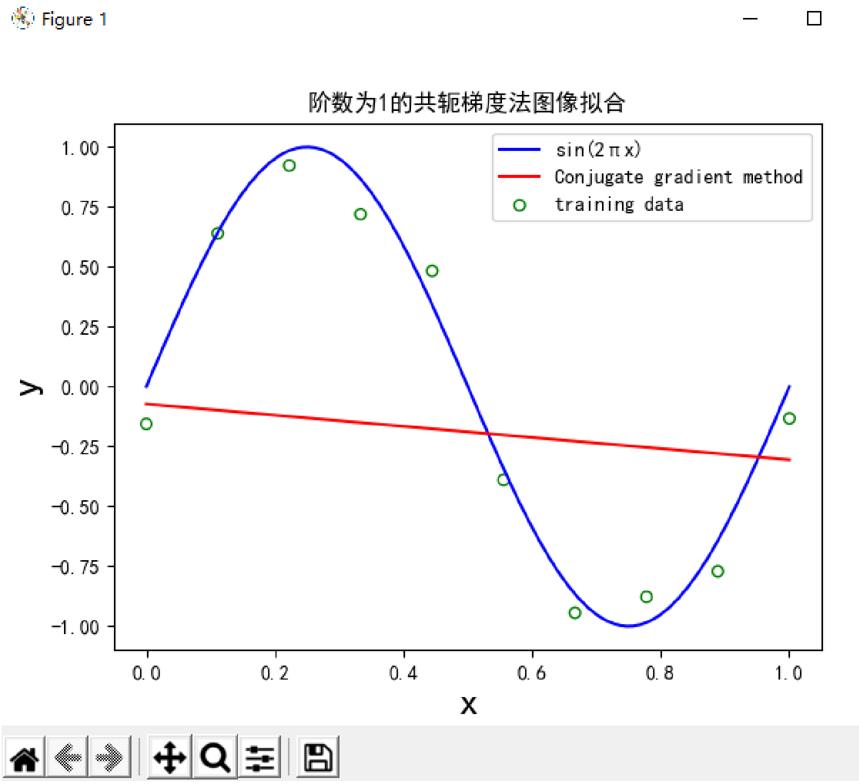


如图，a 太小出现了过拟合情况，适合的 a 区间在 0.01——0.1

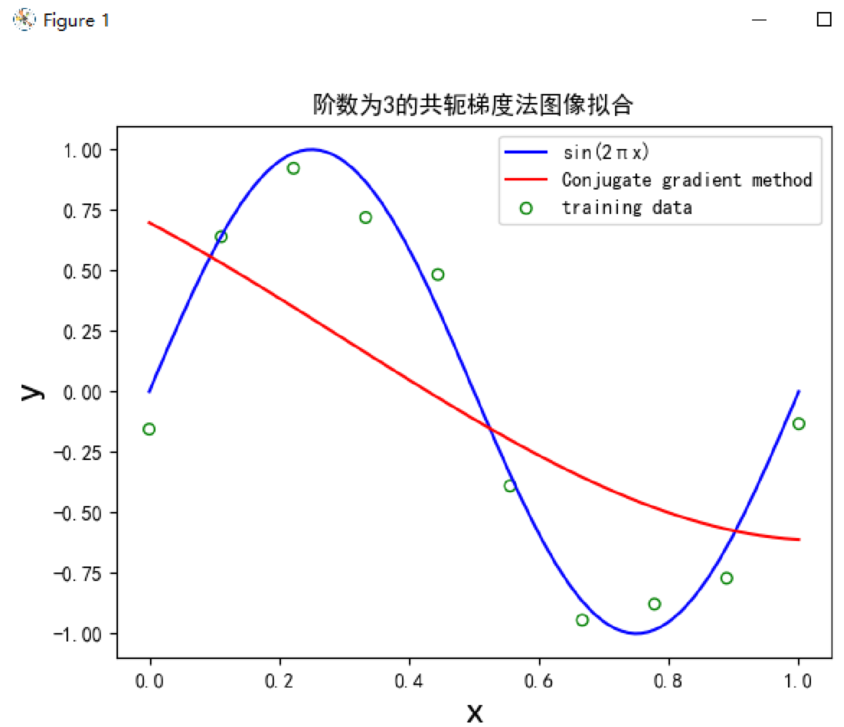
5.4 共轭梯度法

数据量为 10 时：

阶数为 1：



阶数为 3：



阶数为 5：

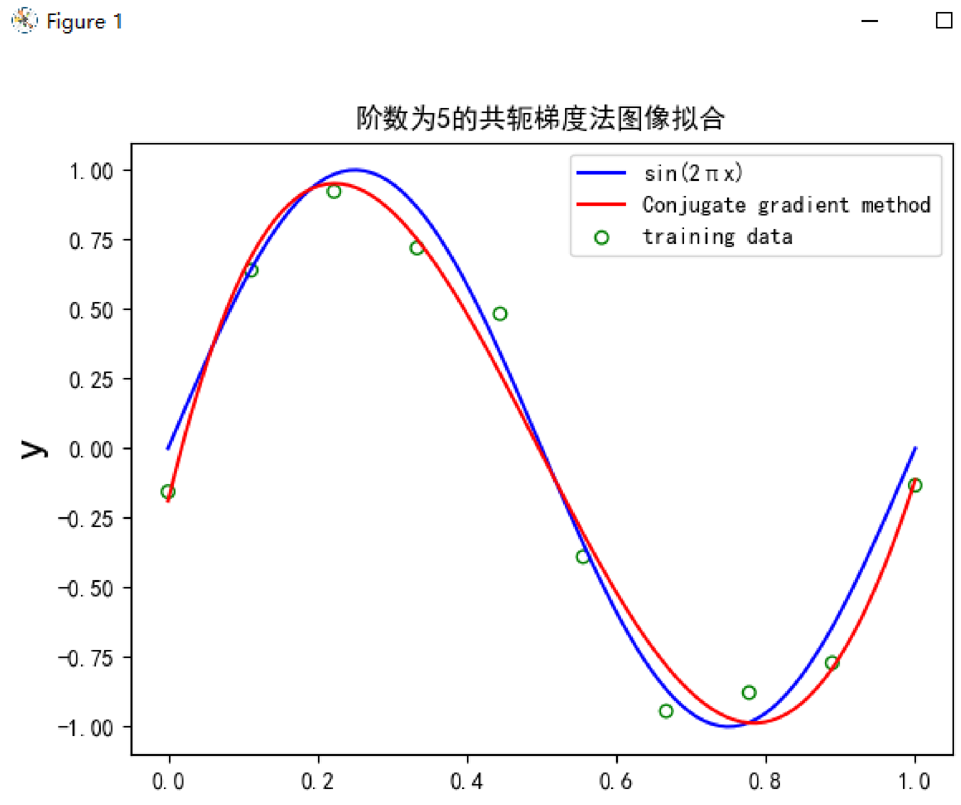


Figure 1
阶数为5的共轭梯度法图像拟合
sin(2πx)
Conjugate gradient method
training data
y
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0

阶数为 7：

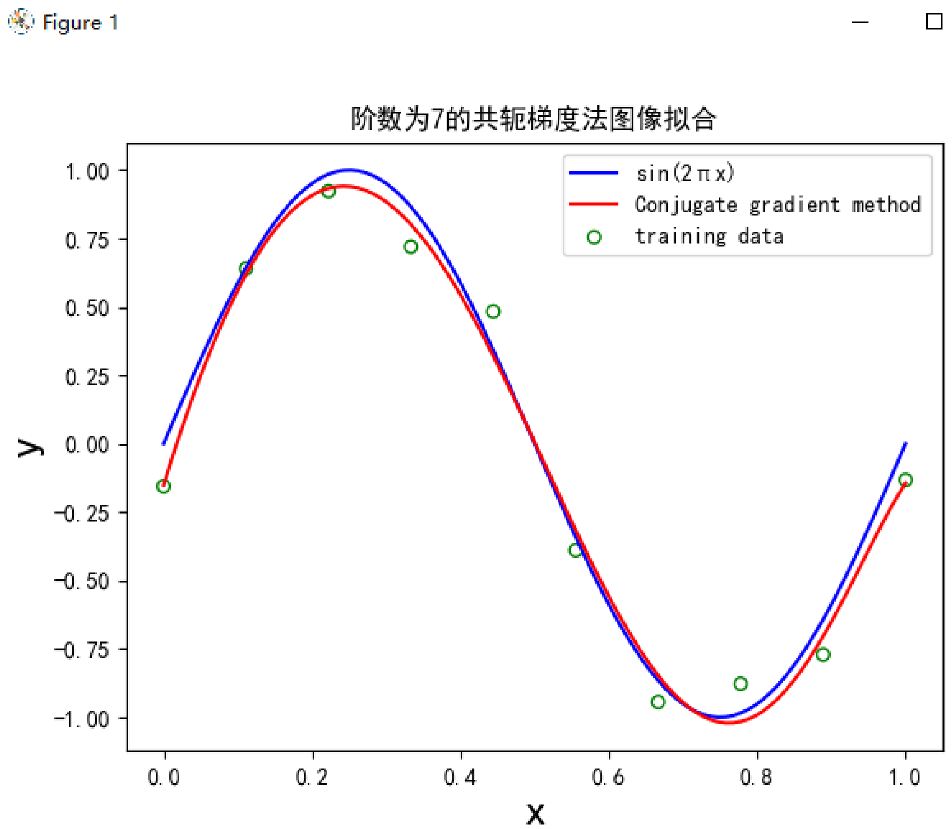
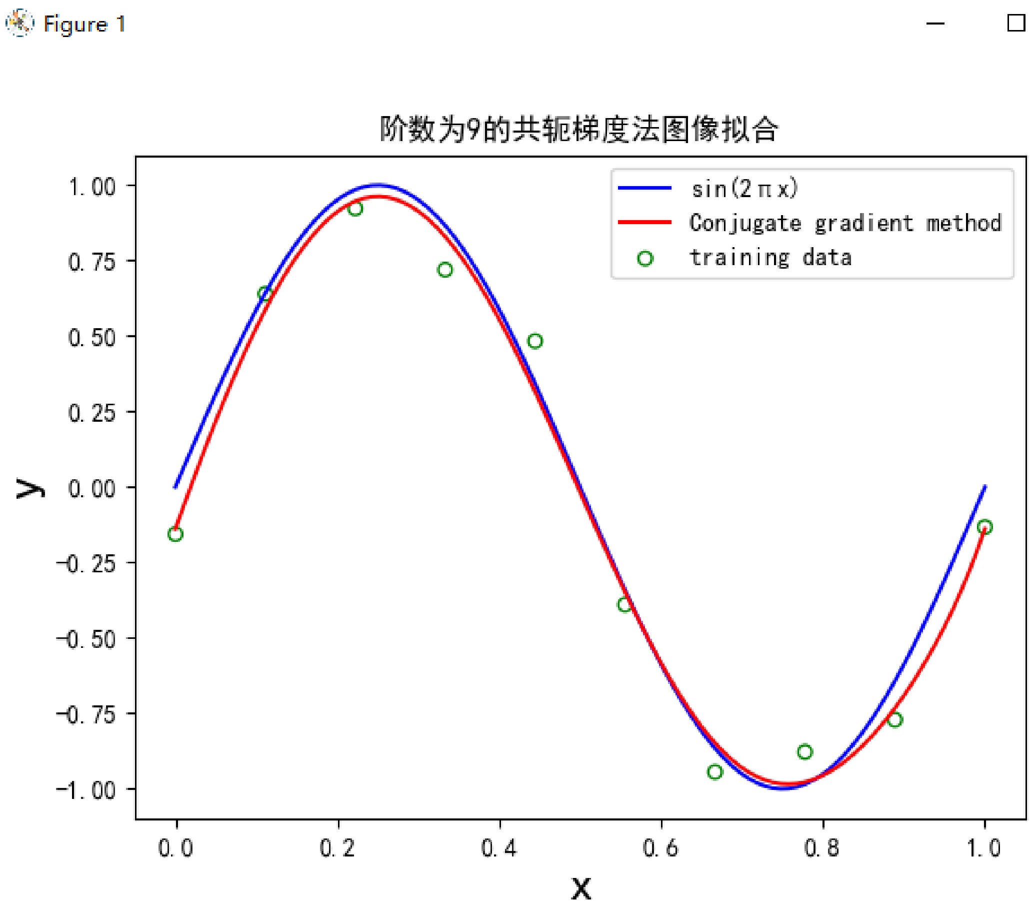


Figure 1
阶数为7的共轭梯度法图像拟合
sin(2πx)
Conjugate gradient method
training data
y
x
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.0
0.2
0.4
0.6
0.8
1.0

阶数为 9：



1、3 阶都欠拟合，5、7、9 阶拟合情况都比较好。

欠拟合分析：步数太少，难以选取合适方向进行极限逼近，导致误差较大。

6 .实验结论

1. 直接求解析解思路较为简单，缺点是X^TX可能不可逆，不可逆的情况下无法采用此方法求解。

2. 数据量较小时最小二乘法比较容易出现过拟合情况，可以通过加入惩罚项和增大数据量的办法进行改善。惩罚项系数的选取和数据量的大小有一定关系，需要选取合适的惩罚项系数进行拟合。我对选取系数还没有比较好的方法，暂时只能通过枚举进行粗略的选举。

3. 梯度下降法：运行速度很慢，并且学习率 a 的选取对拟合的影响比较大，我感觉我选取的参数拟合结果不是很理想。暂时没有特别好的解决办法。

4. 共轭梯度法：数学思想和推导比较难理解，但是代码实现很好。逼近速度很快，拟合效果也很不错。趋势上看阶数越高拟合效果越好，基本贴合。

5. 解释过拟合：过拟合即训练集上效果很好，测试集上效果不佳的情况。加入惩罚项就是减少连续样本的损失，使图像在整体上更为贴合，改善过拟合情况。为改善过拟合可以通过正则化、增加数据量等方法。

6. 梯度下降法可能存在的问题：下降陷入某个“谷底”，反复来回跳转，无法跳出，使局部最优解代替了想要得到的最优解。
   可能的解决办法：间断地加入一个较大的步长，使遇到上述情况时可以跳过谷底再次迭代跳转寻找最优解。