《模式识别与机器学习A》实验报告

实验题目： 多项式拟合正弦函数

班级： 2203601

学号： 2022113416

姓名： 刘子康

**实验报告内容**

**一、实验目的**

·基于梯度下降法，学会使用高阶多项式函数拟合正弦函数；

·掌握机器学习训练拟合原理（无惩罚项的损失函数）；

·掌握加惩罚项（L2范数）的损失函数优化、梯度下降法、理解过拟合、克服过拟合的方法（如加惩罚项、增加样本等）。

**二、实验内容**

生成一组某正弦函数上、添加了噪声的散点样本，尝试利用梯度下降法作为优化方法，使用高阶多项式函数拟合该正弦函数。比较不同数据量、不同超参数和不同多项式阶数下的拟合结果，解释现象原因，分析影响拟合效果的因素。

若模型出现过拟合现象，尝试通过添加和调整惩罚项、增加样本数量、调整超参数等方法克服过拟合问题，提高模型泛化能力。

**三、实验环境**

·操作系统：Windows 11

·编程语言：Python 3.10

·第三方库：Numpy 1.23.4，Matplotlib 3.8.2

·IDE：Pycharm 2022社区版

**四、实验过程、结果及分析**

**4.1 实验原理**

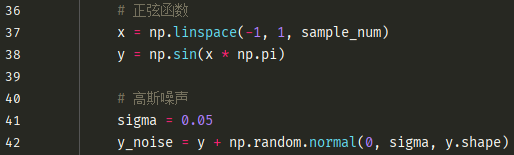
多项式函数拟合基于数学中的多项式逼近方法，它通过构造一个形如的多项式函数，来逼近已知数据点集所代表的真实函数或关系，即使得这些数据点到该多项式函数的距离（或误差）之和最小。

梯度下降法是一种一阶迭代优化算法，其核心原理是利用负梯度方向作为搜索方向，因为在多元函数的某一点处，函数值沿着负梯度方向下降最快。当使用均方误差作为损失函数时，梯度下降法从某个初始点开始，通过迭代地计算损失函数关于模型参数的梯度，并按照梯度的反方向（即负梯度方向）更新参数，从而使得损失函数值逐渐减小，直至达到一个局部最小值或全局最小值。

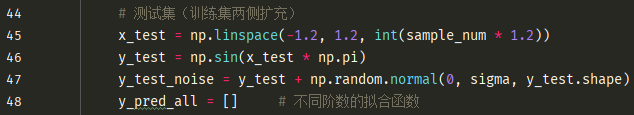
**4.2 实验过程**

**4.2.1 生成样本点**

使用Numpy库的linspace(-1, 1, num)函数生成(-1, 1)区间的num个x坐标点（num由自己指定），再使用np.sin(np.pi \* x)函数生成对应的函数值序列y，作为原始正弦函数。对生成的y添加均值为0、标准差为0.005的高斯噪声，得到y\_noise，作为多项式拟合的真实值。

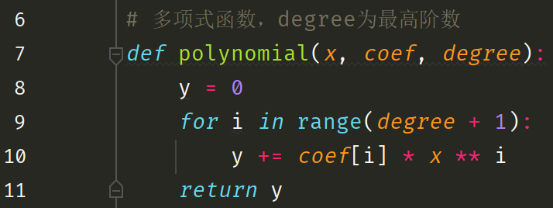


用上述方法，将训练集向两侧扩充1.2倍，作为测试集，用于测试拟合效果。

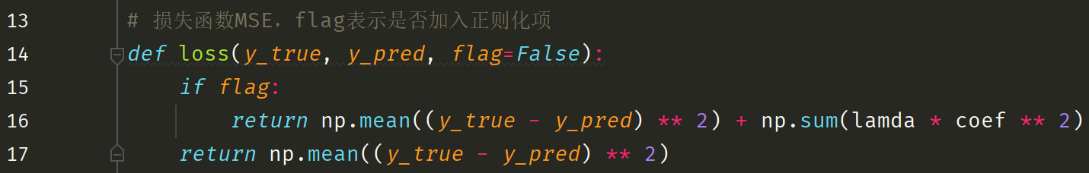


**4.2.2 定义多项式函数、损失函数和梯度计算函数**

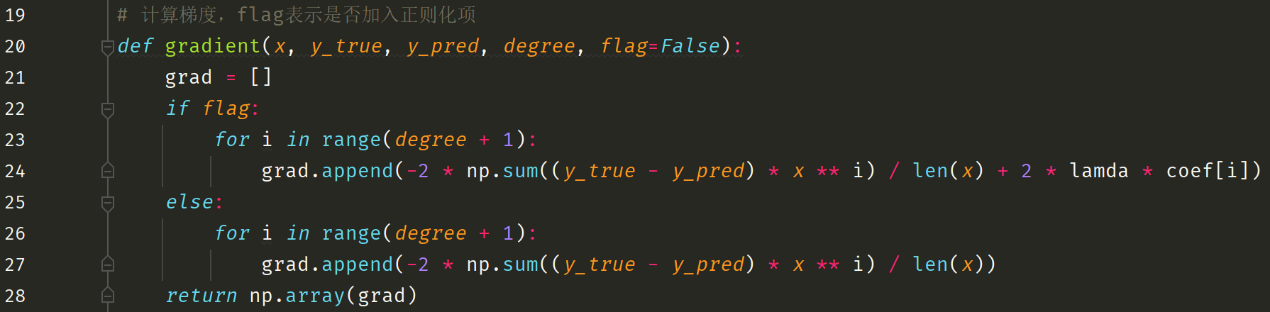
根据输入的最高阶数生成一个形如的多项式函数polynomial()，返回带入参数x向量计算后的函数值y。



损失函数loss()使用均方误差（，yi为第i个样本的真实值，y^i为第i个样本的预测值），通过计算预测值和真实值之间的误差平方和的均值作为差异，来衡量模型的预测性能。均方误差越小，表示模型的预测越准确；均方误差越大，表示模型的预测误差越大。



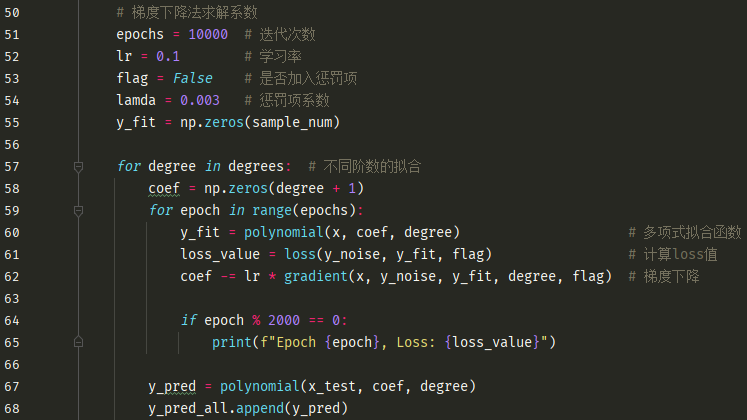
每次迭代过程的下降梯度通过损失函数对系数向量coef求偏导得到（），梯度计算函数为gradient()。



**4.2.3 训练过程**

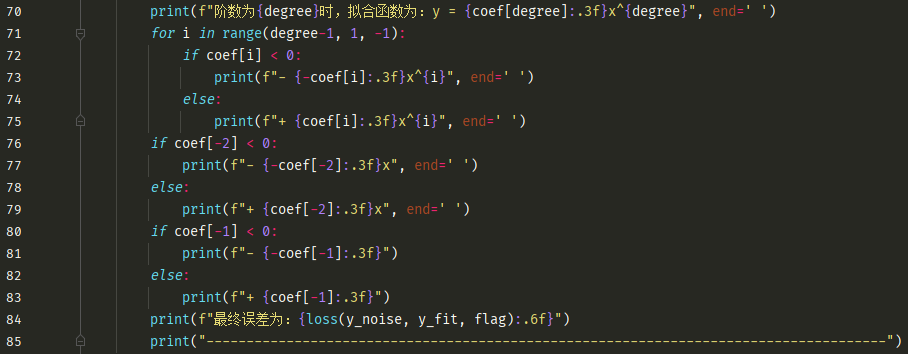
设置迭代次数epochs为10000，确定生成数据量，对于不同的多项式阶数，通过多项式函数polynomial()计算预测值y\_fit，之后通过loss()函数和gradient()函数计算当前loss值和下降梯度，并进入下一次迭代。

可以选择有惩罚项或无惩罚项，若选择有惩罚项，则在计算loss和梯度时加入正则化项，loss为均方误差加上，其中λ为惩罚项系数，为系数向量的L2范数的平方；梯度为对系数向量coef的偏导数加上。

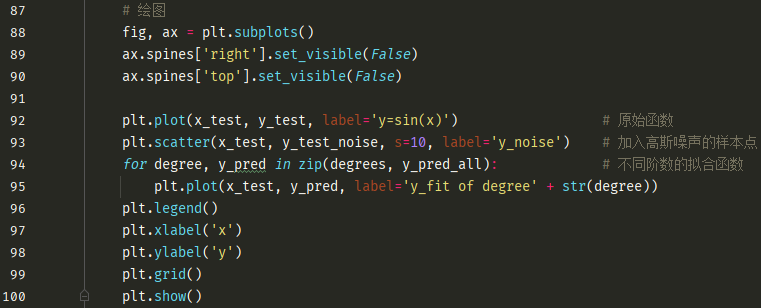


**4.2.4 输出结果并绘图**

当拟合完成后，对于不同的多项式阶数，分别输出拟合函数和最终误差。



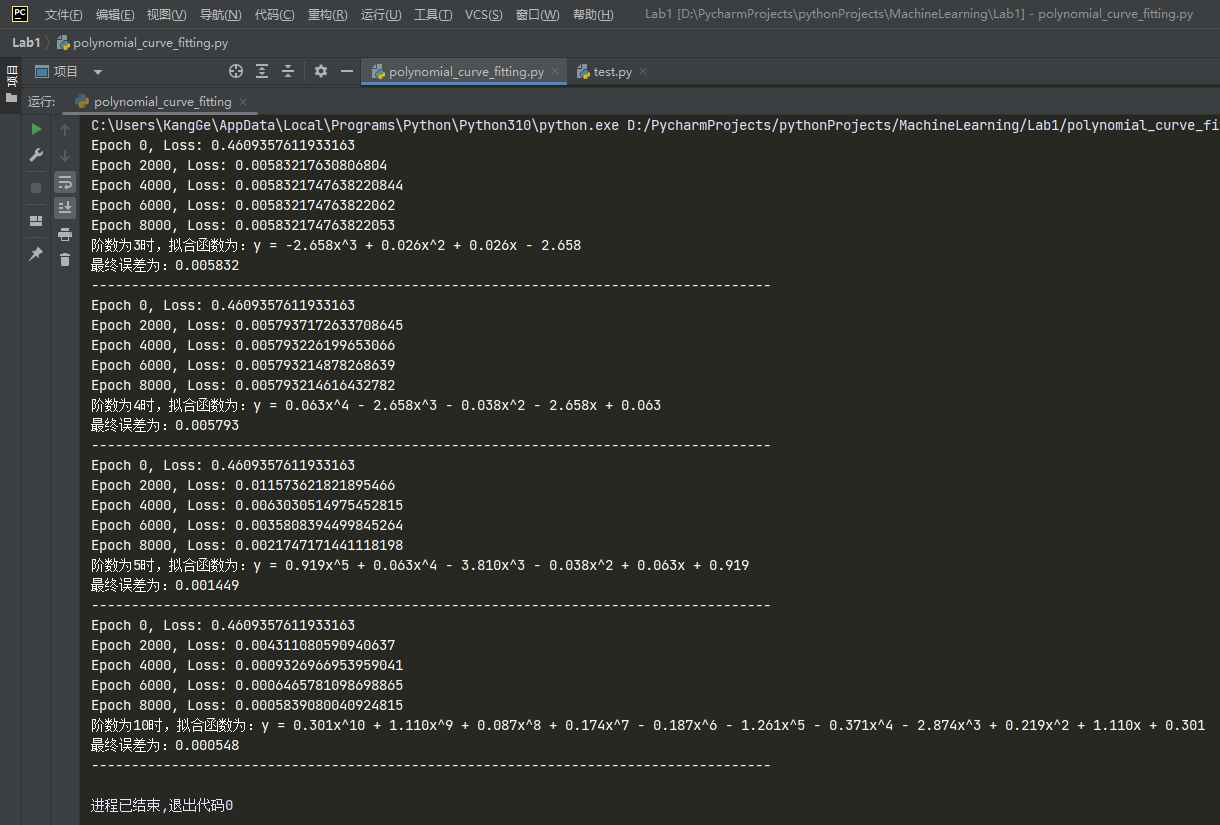
利用训练集向两侧扩充1.2倍后的样本作为测试集，使用Matplotlib库的pyplot模块绘制图像，将拟合函数与原始函数和加噪声后的样本点进行对比，分析拟合效果，判断是否发生过拟合现象。

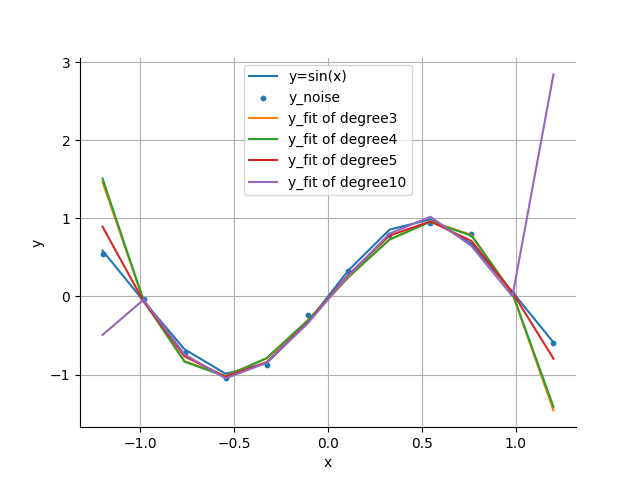


**4.3 实验结果及分析**

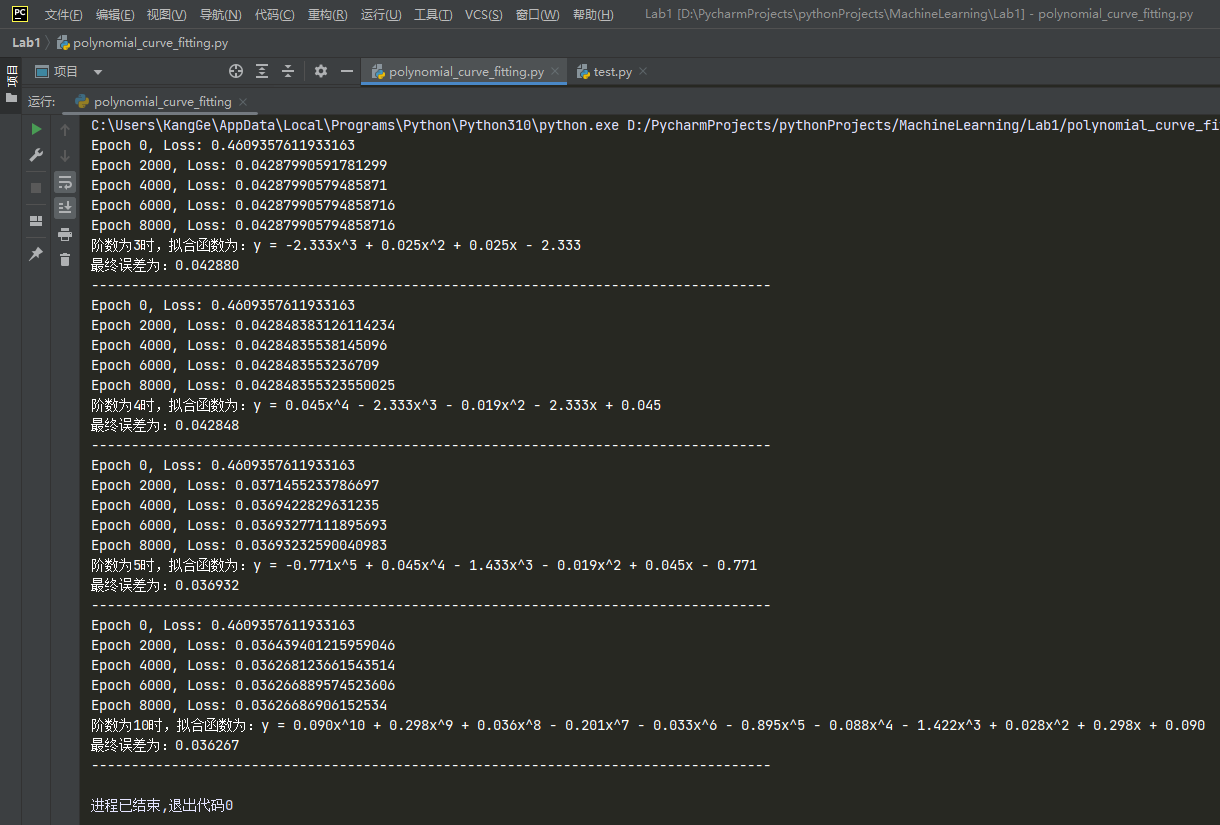
**4.3.1样本量为10**

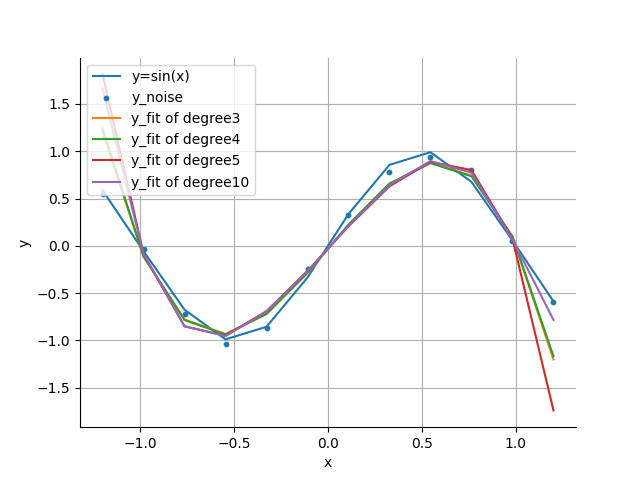
·无惩罚项





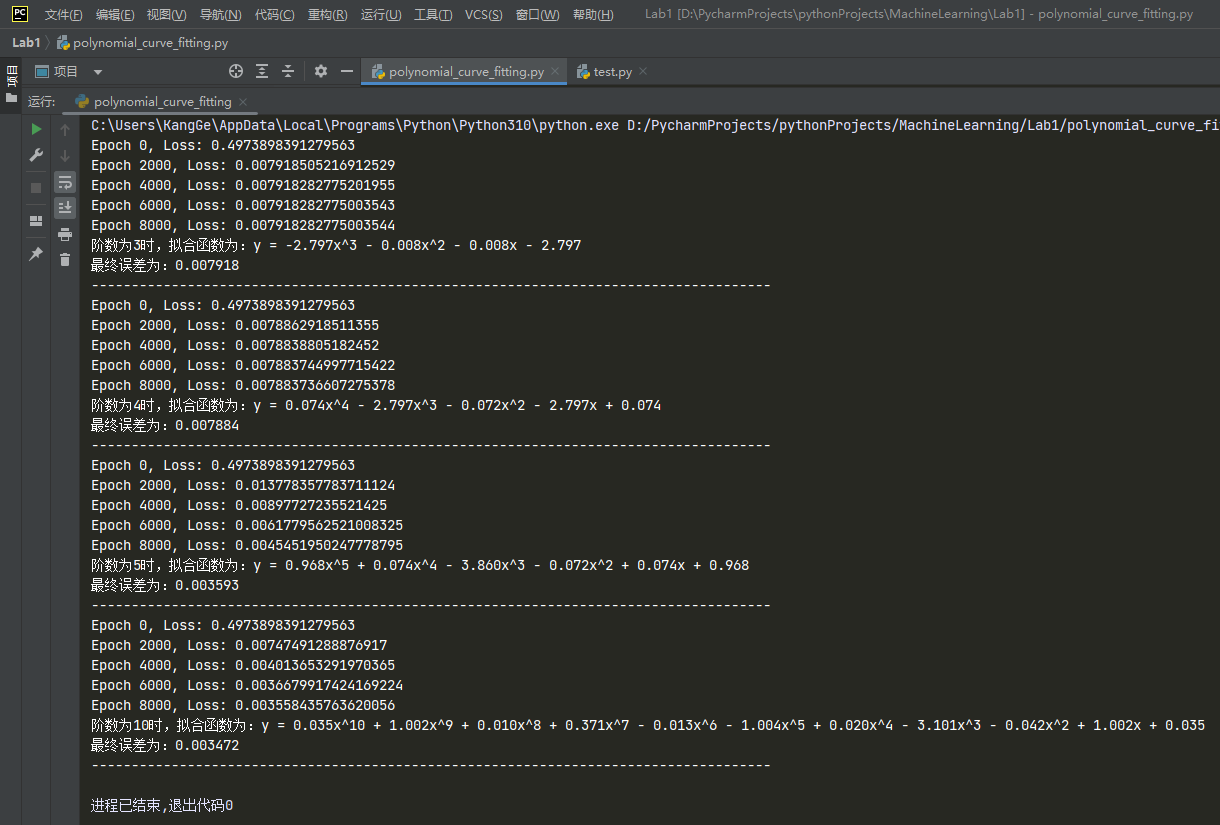
·有惩罚项

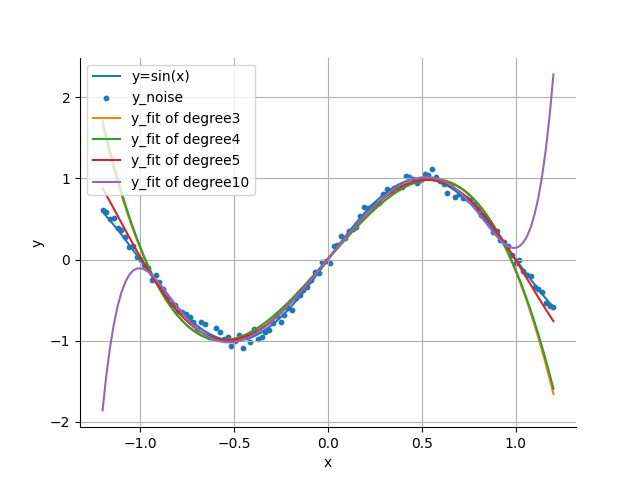




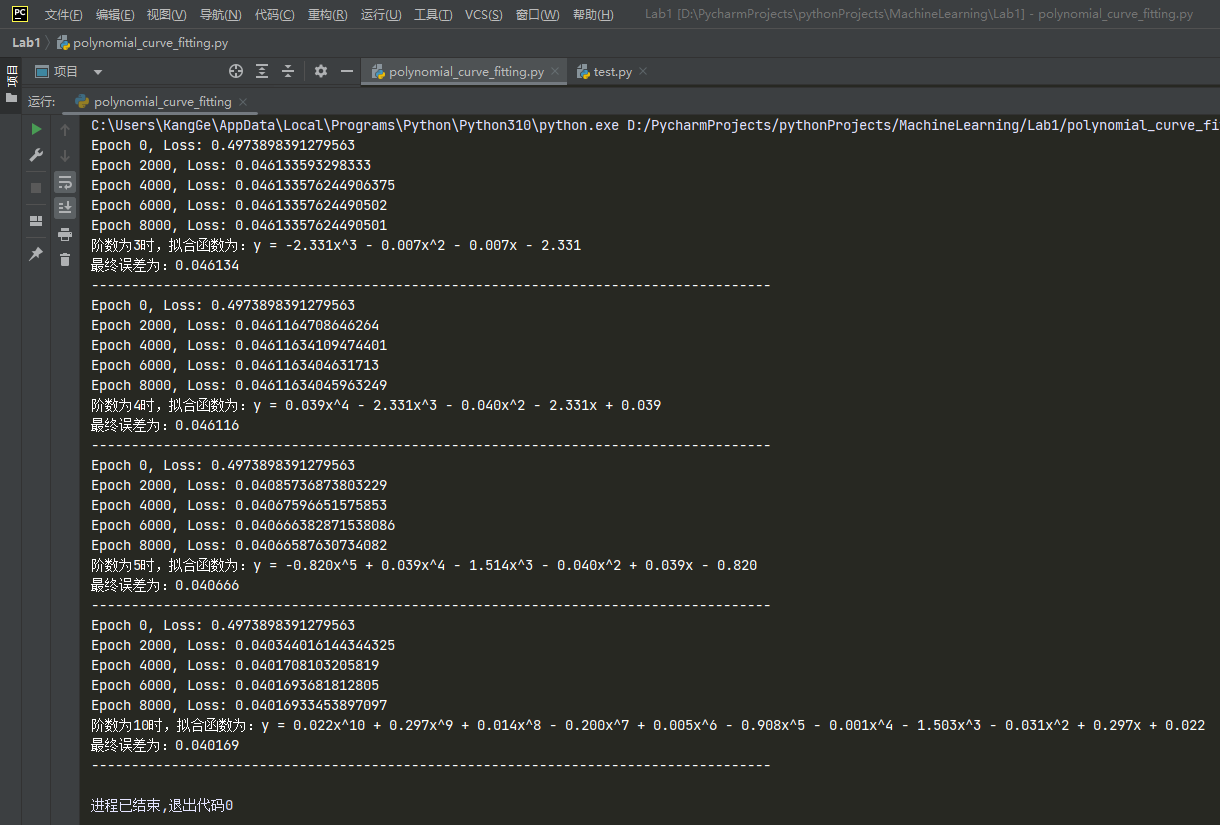
**4.3.2样本量为100**

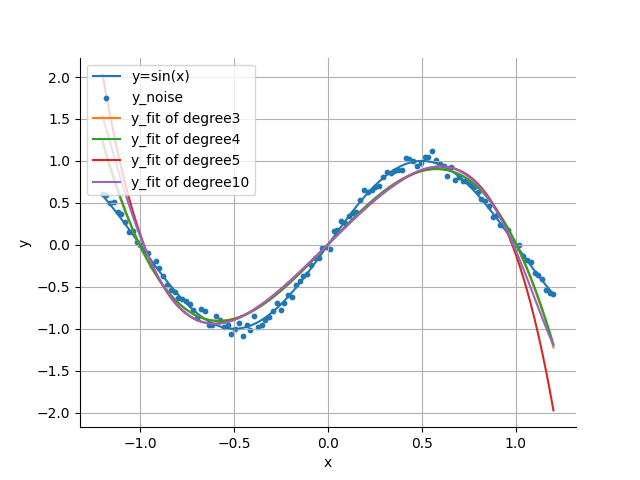
·无惩罚项





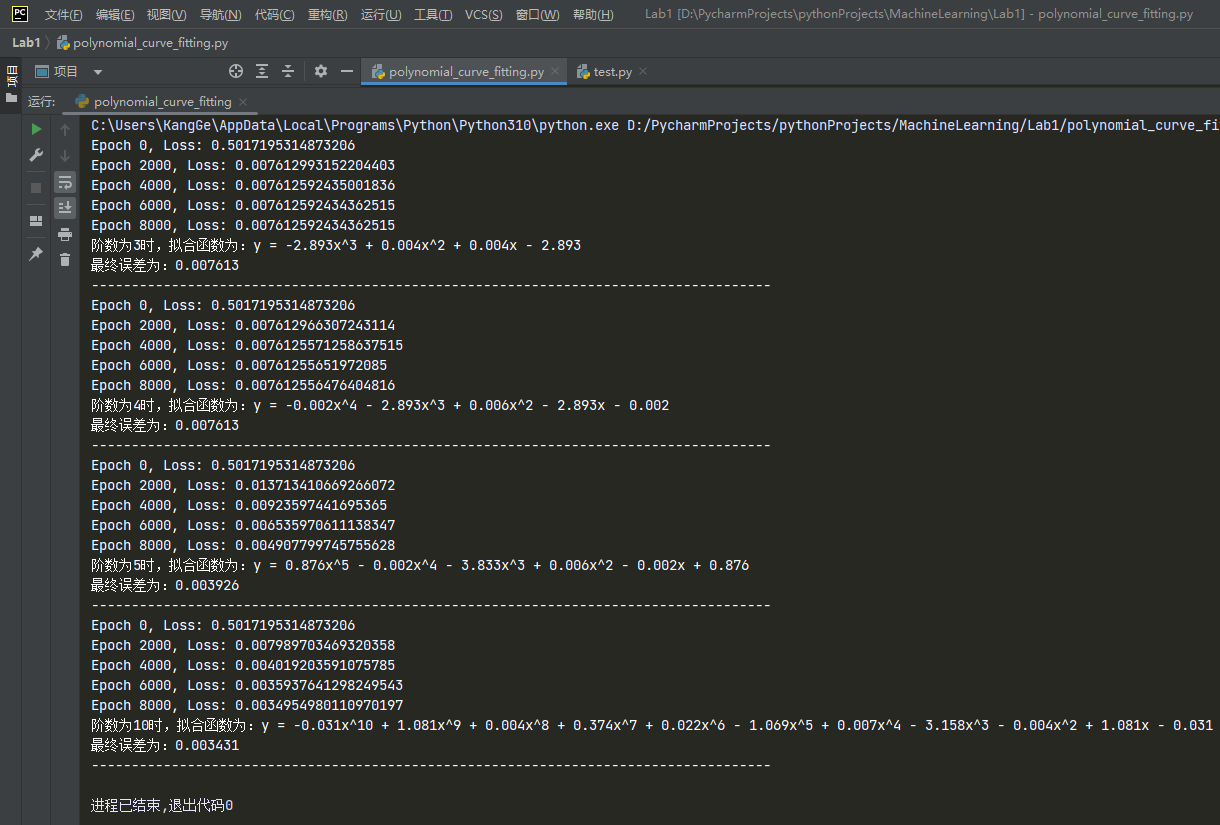
·有惩罚项

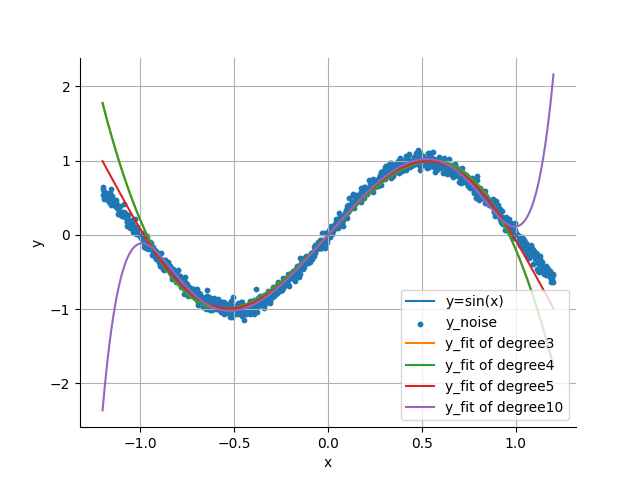




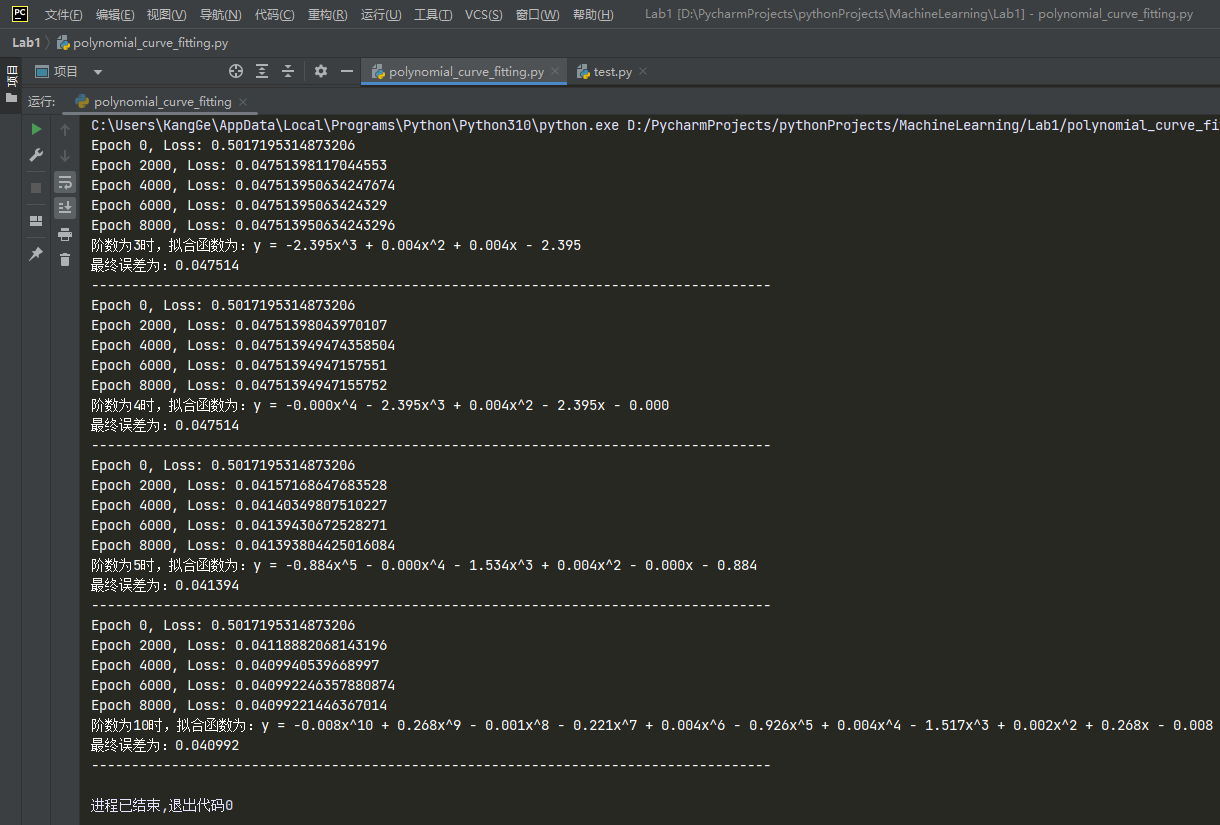
**4.3.3样本量为1000**

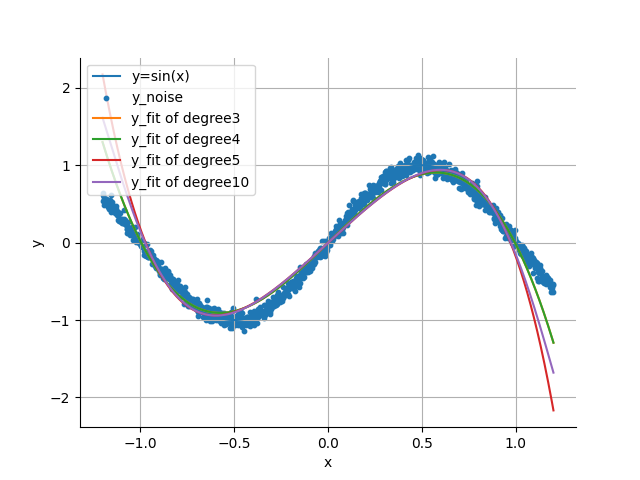
·无惩罚项





·有惩罚项





**五、实验总体结论**

在进行基于梯度下降法优化的多项式函数拟合局部正弦函数过程中，通过调整合适的数据量、学习率以及惩罚项系数等超参数，实现了较好的拟合效果。

过拟合指模型复杂度高于实际问题，模型在训练集上表现很好，但在测试集上却表现很差，即无法将从训练集学习到的特征泛化至测试集。根据实验结果，当使用高阶多项式进行拟合时（例如10阶），会出现明显过拟合现象，这时可以通过加入惩罚项或增加数据量来抑制和解决。实验中惩罚项使用L2范数是合理的，因为生成的噪声符合高斯分布，若高斯分布取到最大似然，同时L2 损失函数也会最小。

然而，拟合结果显示加入惩罚项虽然抑制了过拟合问题，但也会使得拟合效果变差，需要合理调整惩罚项系数和多项式阶数，使拟合效果和防过拟合能力达到较好的平衡。

**六、完整实验代码**

1. import numpy as np
2. import matplotlib.pyplot as plt
3. np.random.seed(16)  *# 设置随机数种子，便于结果复现*
4. *# 多项式函数，degree为最高阶数*
5. def polynomial(x, coef, degree):
6. y = 0
7. for i in range(degree + 1):
8. y += coef[i] \* x \*\* i
9. return y
10. *# 损失函数MSE，flag表示是否加入正则化项*
11. def loss(y\_true, y\_pred, flag=False):
12. if flag:
13. return np.mean((y\_true - y\_pred) \*\* 2) + np.sum(lamda \* coef \*\* 2)
14. return np.mean((y\_true - y\_pred) \*\* 2)
15. *# 计算梯度，flag表示是否加入正则化项*
16. def gradient(x, y\_true, y\_pred, degree, flag=False):
17. grad = []
18. if flag:
19. for i in range(degree + 1):
20. grad.append(-2 \* np.sum((y\_true - y\_pred) \* x \*\* i) / len(x) + 2 \* lamda \* coef[i])
21. else:
22. for i in range(degree + 1):
23. grad.append(-2 \* np.sum((y\_true - y\_pred) \* x \*\* i) / len(x))
24. return np.array(grad)
25. if \_\_name\_\_ == '\_\_main\_\_':
26. *# 样本量，阶数*
27. sample\_num = 10
28. degrees = [3, 4, 5, 10]
29. *# 正弦函数*
30. x = np.linspace(-1, 1, sample\_num)
31. y = np.sin(x \* np.pi)
32. *# 高斯噪声*
33. sigma = 0.05
34. y\_noise = y + np.random.normal(0, sigma, y.shape)
35. *# 测试集（训练集两侧扩充）*
36. x\_test = np.linspace(-1.2, 1.2, int(sample\_num \* 1.2))
37. y\_test = np.sin(x\_test \* np.pi)
38. y\_test\_noise = y\_test + np.random.normal(0, sigma, y\_test.shape)
39. y\_pred\_all = []     *# 不同阶数的拟合函数*
40. *# 梯度下降法求解系数*
41. epochs = 10000  *# 迭代次数*
42. lr = 0.1        *# 学习率*
43. flag = False    *# 是否加入惩罚项*
44. lamda = 0.003   *# 惩罚项系数*
45. y\_fit = np.zeros(sample\_num)
46. for degree in degrees:  *# 不同阶数的拟合*
47. coef = np.zeros(degree + 1)
48. for epoch in range(epochs):
49. y\_fit = polynomial(x, coef, degree)         *# 多项式拟合函数*
50. loss\_value = loss(y\_noise, y\_fit, flag)     *# 计算loss值*
51. coef -= lr \* gradient(x, y\_noise, y\_fit, degree, flag)  *# 梯度下降*
52. if epoch % 2000 == 0:
53. print(f"Epoch {epoch}, Loss: {loss\_value}")
54. y\_pred = polynomial(x\_test, coef, degree)
55. y\_pred\_all.append(y\_pred)
56. print(f"阶数为{degree}时，拟合函数为：y = {coef[degree]:.3f}x^{degree}", end=' ')
57. for i in range(degree-1, 1, -1):
58. if coef[i] < 0:
59. print(f"- {-coef[i]:.3f}x^{i}", end=' ')
60. else:
61. print(f"+ {coef[i]:.3f}x^{i}", end=' ')
62. if coef[-2] < 0:
63. print(f"- {-coef[-2]:.3f}x", end=' ')
64. else:
65. print(f"+ {coef[-2]:.3f}x", end=' ')
66. if coef[-1] < 0:
67. print(f"- {-coef[-1]:.3f}")
68. else:
69. print(f"+ {coef[-1]:.3f}")
70. print(f"最终误差为：{loss(y\_noise, y\_fit, flag):.6f}")
71. print("-------------------------------------------------------------------------------------")
72. *# 绘图*
73. fig, ax = plt.subplots()
74. ax.spines['right'].set\_visible(False)
75. ax.spines['top'].set\_visible(False)
76. plt.plot(x\_test, y\_test, label='y=sin(x)')              *# 原始函数*
77. plt.scatter(x\_test, y\_test\_noise, s=10, label='y\_noise')*# 加高斯噪声的样本点*
78. for degree, y\_pred in zip(degrees, y\_pred\_all):         *# 不同阶数的拟合函数*
79. plt.plot(x\_test, y\_pred, label='y\_fit of degree' + str(degree))
80. plt.legend()
81. plt.xlabel('x')
82. plt.ylabel('y')
83. plt.grid()
84. plt.show()

**七、参考文献**

[1] 刘远超. 深度学习基础: 高等教育出版社, 2023.