计算机科学与技术学院2018机器学习—实验报告

- 课程名称： 机器学习
- 课程类型： 选修
- 实验题目： 多项式拟合正弦函数
- 学号：1160100626
- 姓名：单心茹

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)

实验要求及实验环境

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。

实验环境

Win10 + 8GB RAM + python 3.6

设计思想与实现

对函数$y = sin(2\pi x)$离散成数据点，得到样本，对y的值加入$\mu = 0, \sigma = 0.1$**的高斯噪声**，采用最小二乘法，批量梯度下降法，共轭梯度下降法实现多项式拟合曲线。主要过程为：

- 将多项式回归转化为线性回归，初始化样本数据，形成矩阵$A\theta = Y$
- 采用不同的方法求解
- 计算损失函数
- 比较不同数据量，不同参数，不同多项式阶数的结果

由于是多项式回归，由函数生成样本点之后，要根据阶数初始化为矩阵

$X = x_1, x_2, ..., x_k$初始化为：

$$\begin{bmatrix} 1 & x_1 & x_1^2 ..... x_1^m \\ 1 & x_2 & x_2^2 ..... x_2^m \\ 1 & x_3 & x_3^2 ..... x_3^m \\ ..... & & \\ 1 & x_k & x_k^2 ..... x_k^m \end{bmatrix}$$

最小二乘法拟合曲线

由最小二乘法公式：

$$\theta = (X^T X)^{-1} X^T Y$$

构造$X_{N \times feture}, Y_{feture \times 1}$矩阵，其中$N$**为样本组数**，在线性回归中$feture$为特征数，类比到多项式回归中，$fetrue - 1$**为多项式阶数。**

则可以直接求得θ

首先将样本点按照**训练集：测试集为7:3**的比例进行划分

```
def initData(x, num):
    #训练集与测试集的比例为7:3
    x_train = np.mat(random.sample(x, round(num*0.7)))
    y_train = np.sin(2* math.pi *x_train)
    x_test = []
    for item in x:
        if item not in x_train:
            x_test.append(item)
    x_test = np.mat(x_test)
    y_test = np.sin(2*math.pi*x_test)
    return x_train, y_train, x_test, y_test
```

再将样本点初始化为相应矩阵：

```
def initMatrix(x, N):
    m = []
    for i in range(0, N):
        a = x**(i)
        m.append(a)
    x_new = np.mat(np.array(m)).T
    return x_new
```

调用**非正则化的最小二乘法**进行求解：

```
def normalResult(x_train, y_train, x_test,y_test, N):
    A = initMatrix(np.array(x_train), N)
    B = y_train.T
    theta_normal = LSE_normal(A, B, N)
    predict_normal = dot(A, theta_normal)
    loss_normal = LOSSNormal(predict_normal, B, N)
    lossTrain_normal.append(loss_normal)

    # 计算测试集的损失函数
    A_test = initMatrix(np.array(x_test), N)
    predict_normal = dot(A_test, theta_normal)
    loss_normal = LOSSNormal(predict_normal, y_test.T, N)
    lossTest_normal.append(loss_normal)
```

同理可以调用**正则化的最小二乘法**进行求解：

```
def regularResult(x_train, y_train, x_test, y_test, N):
    #由训练集求解
    A = initMatrix(np.array(x_train), N)
    B = y_train.T
    theta_regular = LSE_regular(A, B, N, lamda)
    predict_regular = dot(A, theta_regular)
    loss_regular = LOSSRegular(predict_regular, B, theta_regular, N)
    lossTrain_regular.append(loss_regular)

    #计算测试集的损失函数
    A_test = initMatrix(np.array(x_test), N)
    predict_regular = dot(A_test, theta_regular)
    loss_regular = LOSSRegular(predict_regular, y_test.T, theta_regular, N)
    lossTest_regular.append(loss_regular)
```

在实验结果与分析的部分会对正则化与非正则化的结果进行比较与分析

梯度下降法拟合曲线

梯度下降法的目的是找到一组θ，让损失函数达到极小，算法主要思想为：**调整合适的步长，沿着梯度下降的方向搜索，直到满足收敛条件**

1. 初始化θ, 设置学习速率即步长α

2. 由公式

$$\theta j := \theta j + \frac{1}{2m}\sum_{i=1}^{N}(h_\theta(x^{(i)}) - y^{(i)})^2 x_j^i$$

循环更新θ，直到满足收敛条件，**本实验中设置的收敛条件为前一次损失函数的值与后一次的差的绝对值小于一个人工设定的极小值。**

本实验中将θ**初始化为随机向量**

除了初始化矩阵外，在**梯度下降法中对数据进行了特征缩放的处理，即归一化，以提高模型的精度和收敛速度**：

由 Min—Max Scaling得：

$$x = \frac{x_i - \mu}{max(x_i) - min(x_i)}$$

归一化实现部分：

```
def scaling(feature, n):
    sum = feature.sum(axis=0)
    mean = sum/n
    s = feature[n-1] - feature[0]
    feature = (feature-mean)/s
    return feature
```

循环更新θ部分：

```
def gradient_descent(x,theta,y,x_scaling,alpha):
    error = 0
    count = 0
    while count < loop_max:
        count += 1
        h = x * theta
        for j in range(0, N):
            diff = 0
            diff = dot((h - y).T, x_scaling[:, j])       # 求偏导数
            theta[j] = theta[j] - alpha * (diff / num)  # 更新theta
        # 计算损失函数
        loss_function = 0
        h = x * theta
        for i in range(1, num):
            loss_function = loss_function + (y[i] - h[i]) ** 2
        loss_function = loss_function / (2 * num)
        # 判断收敛条件
        if abs(loss_function - error) < epsilon:
            break
        else:
            error = loss_function
    print(count)
    return theta
```

共轭梯度法拟合曲线

给定一个正定矩阵 A，如果两个向量 u 和 v 满足 $u'Av = 0$，就说 u 和 v 是关于 A 共轭的。一个 $m \times m$ 的正定矩阵，最多可以有 m 组相互共轭的向量，而它们就组成了 m 维向量空间里的一组基$p1, p2, \ldots, pm$。给定了一组基后，向量空间里的任何一个元素就可以写成这组基的线性组合，即：

$$x = \sum_{i=1}^{m} \alpha_i p_i$$

在线性回归模型中，回归系数β是线性方程组 $Ax = b$ 的解，其中 $A = X'X, b = X'y$,共轭梯度中，我们沿着p_i的方向下降。

算法如下：

Target: solve linear equation $Ax = b$. $A_{m \times m}$ is positive definite

Input: A, b, x_0 (initial guess)

$r_0 := b - Ax_0$

$p_0 := r_0$

$k := 0$

Loop until $k = m$

$\alpha_k := \dfrac{r_k' r_k}{p_k' A p_k}$

$x_{k+1} := x_k + \alpha_k p_k$

$r_{k+1} := r_k - \alpha_k A p_k$

If r_{k+1} is sufficently small then exit loop

$\beta_k := \dfrac{r_{k+1}' r_{k+1}}{r_k' r_k}$

$p_{k+1} := r_{k+1} + \beta_k p_k$

$k := k + 1$

End loop

Output: x_{k+1}

在共轭梯度的实现中θ**初始化为**0**向量，**$r_0 = b - Ax_0, p_0 = r_0$

算法实现部分如下：

```
    while k <= N:
        r_before = r
        a = dot(r.T, r)/dot(p.T, dot(A, p))
        theta = theta + dot(p, a)
        r = r - A*p*a
        # 判断收敛条件
        if np.linalg.norm(r) < epsilon:
            break
        else:
            beta = dot(r.T, r)/dot(r_before.T, r_before)
            p = r + p*beta
            k = k + 1

```

实验结果与分析

最小二乘法（正则化与非正则化）

多次实验观察，正则化与非正则化的loss的最优解在5~6阶时，既不会产生过拟合现象，损失函数值也较小。

非正则化

实验中为了拟合出光滑曲线，首先由最小二乘法求得θ，然后再利用区间内100个点由θ求得函数值，拟合出光滑曲线

如figure-1所示，为一次程序中，**0~8阶的曲线拟合情况：**

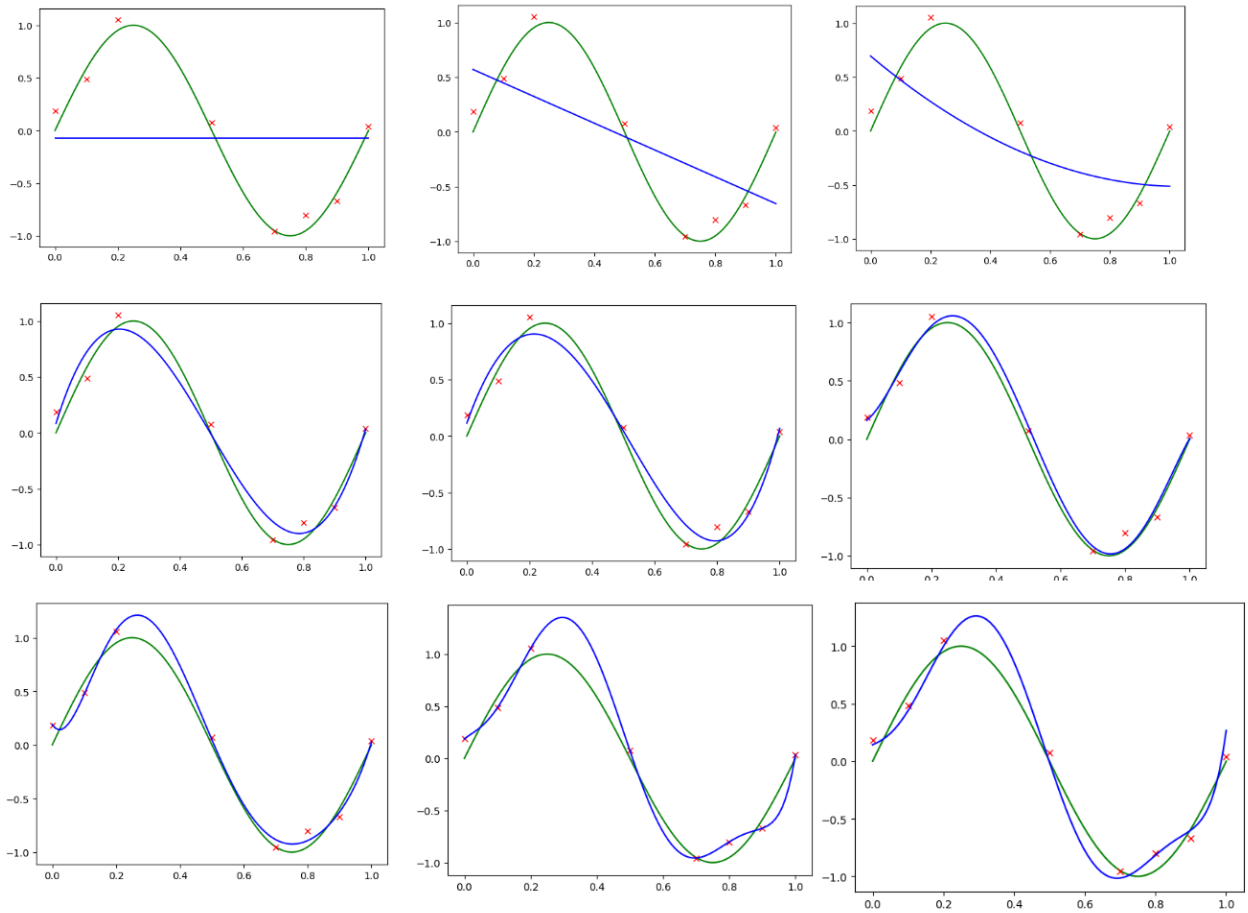

figure-1

记录不同阶数的E_{RMS}（$E_{RMS} = \frac{(2*loss)}{N}$）,以测试集和训练集绘制曲线，可以发现，在**过拟合时**，测试集的E_{RMS}有明显的上升，可以判断此时的过拟合现象造成了**训练的模型对"新数据"的预测有很大偏差**的现象。

如图figure-2：

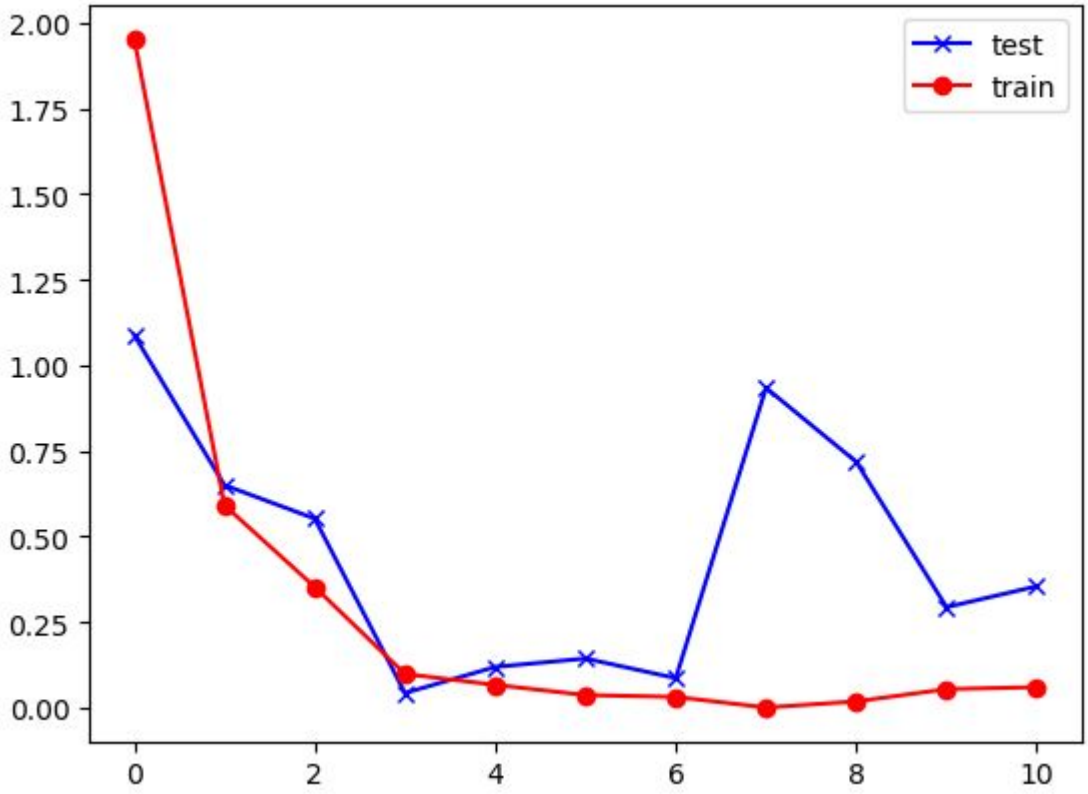

figure-2

正则化

正则化后，首先可以观察到函数的**过拟合现象消除**，如图figure-3为0~8阶加入正则项后，超参数$\lambda = 0.002$时的情况：

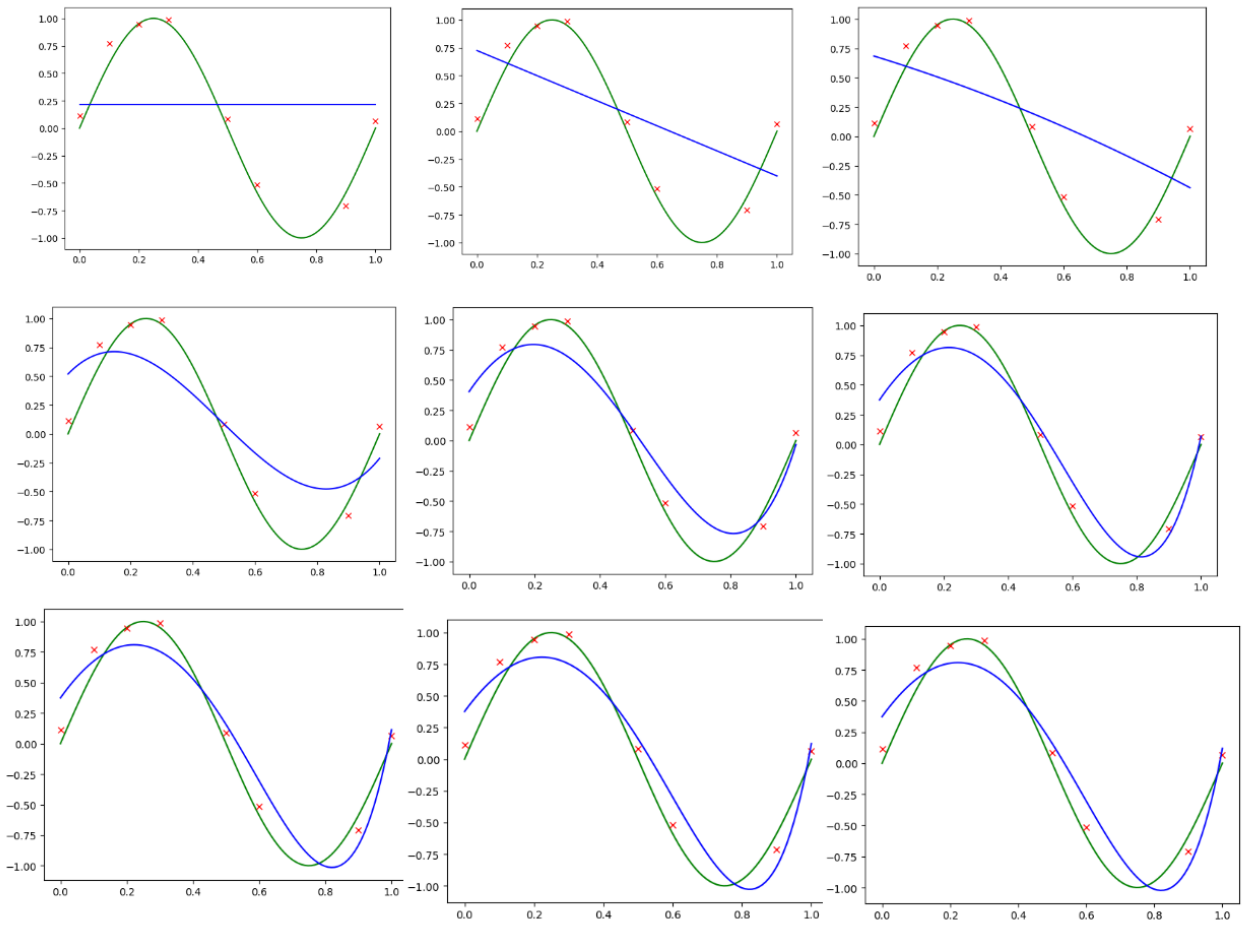

figure-3

再观察E_{RMS}的变化，可以观察到之前在**非正则化时的过拟合情况消失，高阶时，训练集与测试集的loss都较小**：

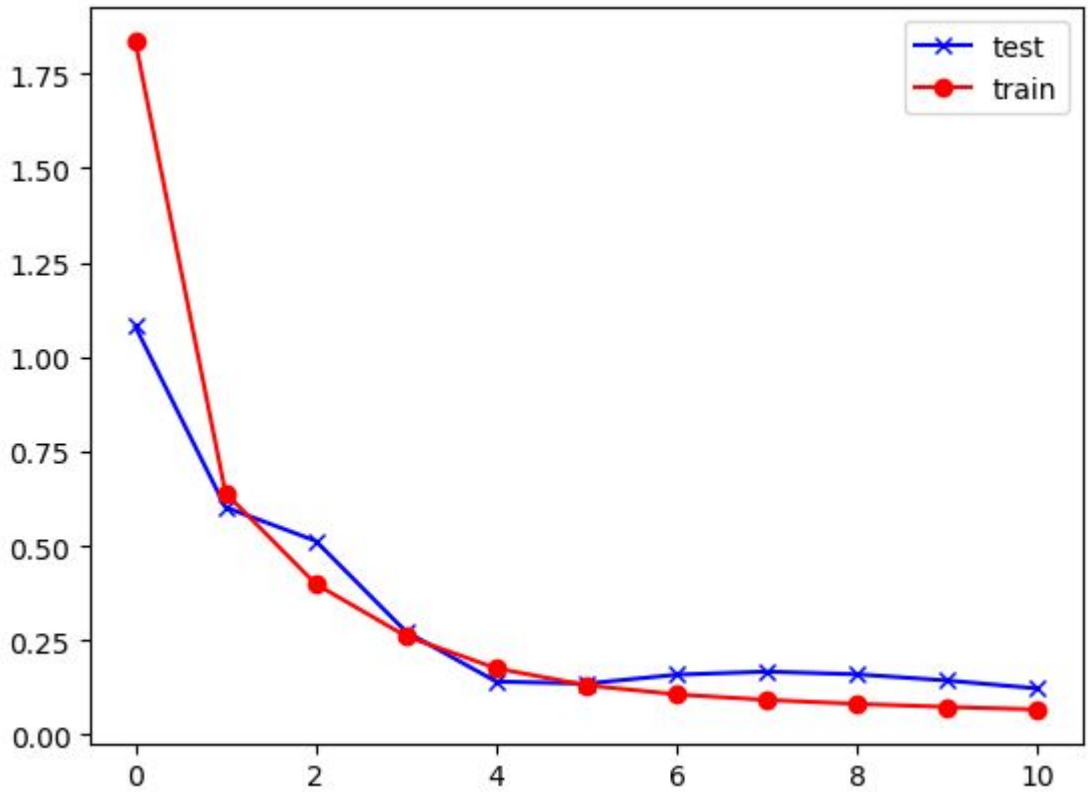

figure-4

不同超参数(λ)的对比

加入正则项后，对比不同$\ln\lambda$在[-40, 0]内的loss的变化，在9阶时分为测试集与训练集，可以观察得到$\ln\lambda$的最优取值范围为[-20,-10]，如图figure-5

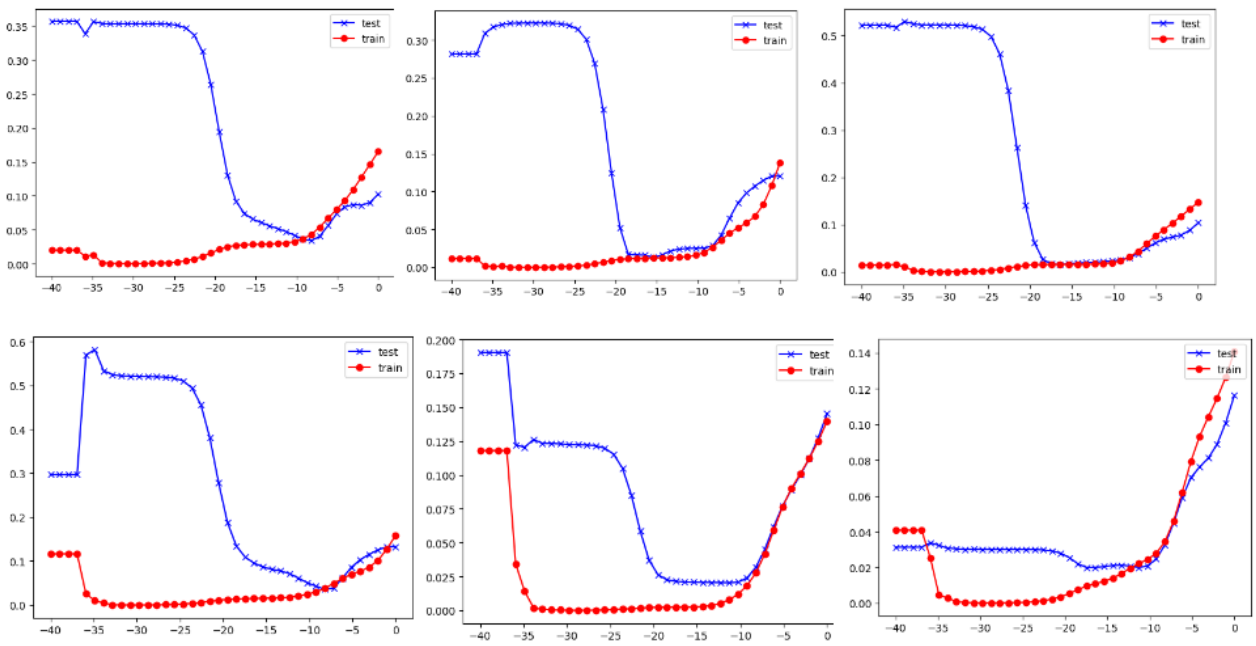

figure-5

梯度下降法

当设置学习速率为$\alpha = 0.5, \epsilon = 10^{-7}$, 加入正则项，$\lambda = 0.00002$, 样本数为20，2~10阶拟合曲线:

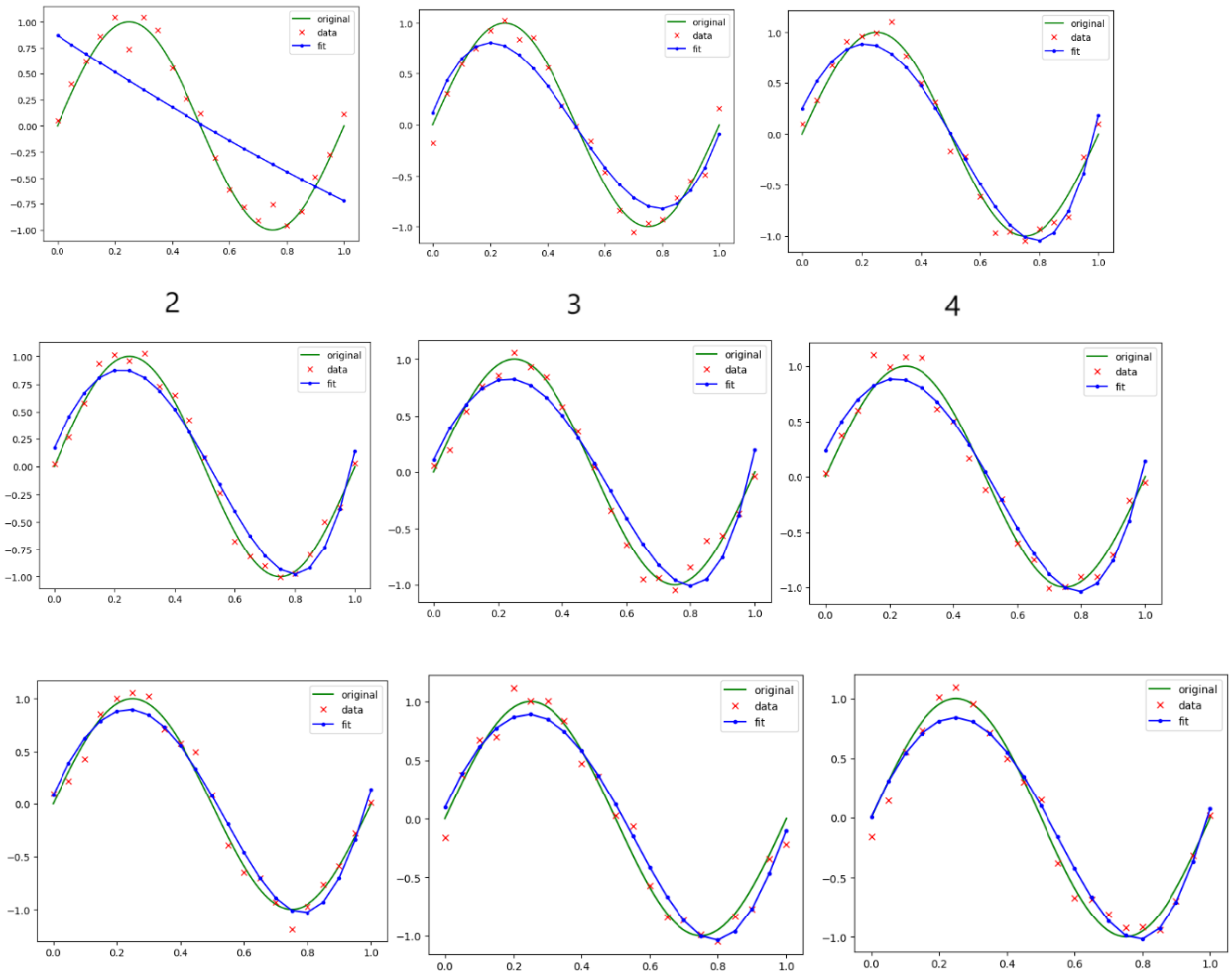

figure-6

当设置学习速率为$\alpha = 0.5, \epsilon = 10^{-7}$, 加入正则项，$\lambda = 0.002$, 样本数为100，2~7阶拟合曲线:

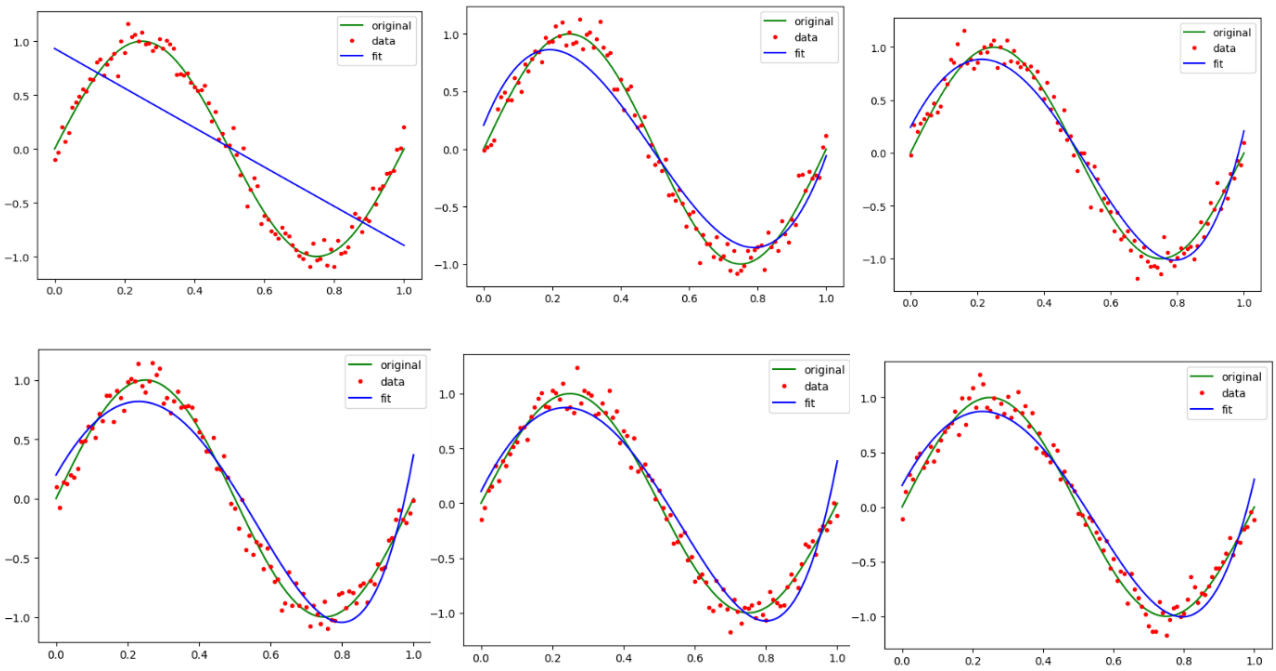

figure-7

对比可以得到，λ可以增大比重，则简化了模型复杂度，由增大样本量来消除过拟合的情况达到了同样的拟合效果，则可以得出在实践中权衡超参数与样本容量，可以消除过拟合现象。

共轭梯度下降法

共轭梯度法迭代次数远小于批量梯度下降法，且运行更快，如图figure-8为**10个样本点，5阶拟合时的情况：**

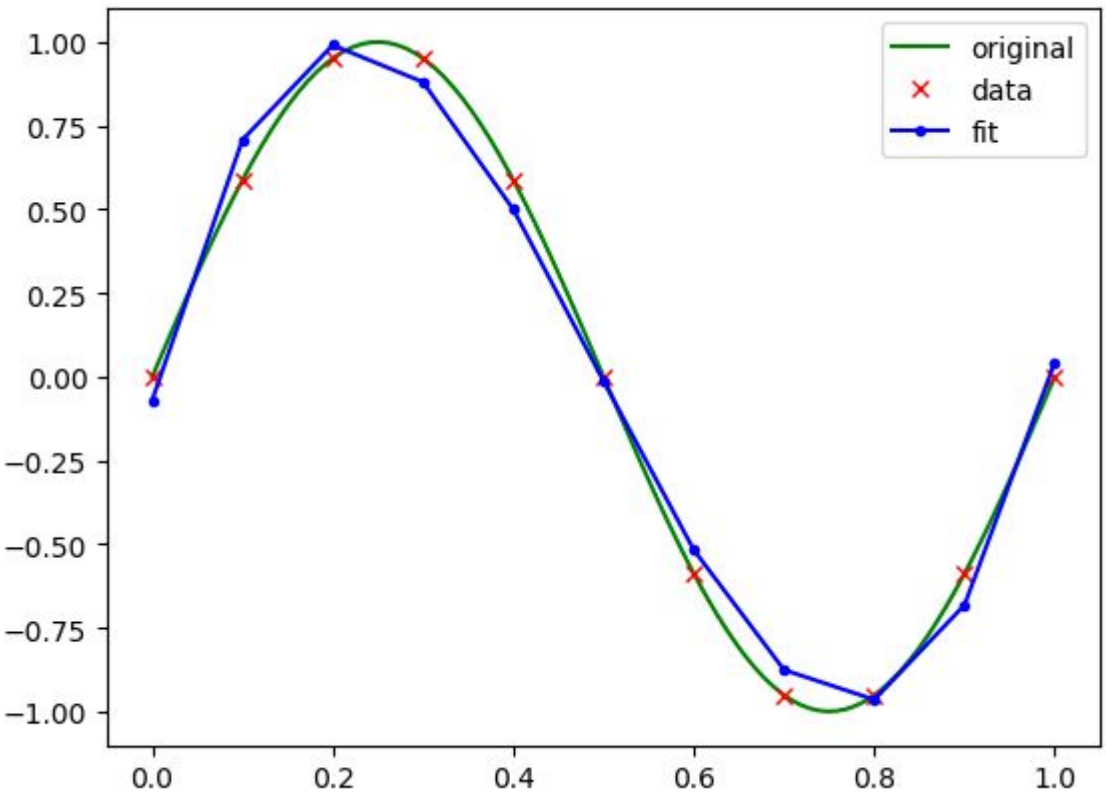

figure-8

结论

- 正则化是一种保留所有特征，却调整不同特征的比重从而消除过拟合的方法。在最小二乘法求解的实验中可以**多次测试不同超参数的值，选取合适的λ, 我发现$\ln \lambda$ 在[-20, -10]范围内，损失函数的值都较小，为合理取值范围**
- **不加入正则项时**，将样本按照7:3的比例分为训练集与测试集，对不同阶数进行实验，**发现在高阶拟合（>7阶）时，出现过拟合现象，即测试集的损失函数值突然增大。**
- **加入正则项后，发现过拟合现象被消除**，测试集在高阶拟合时表现依旧良好
- **正则化和增大样本量都是消除过拟合的好方法**，合理的权衡二者能得到不错的效果
- **梯度下降法的收敛条件(极小值), 学习速率的设置非常重要**，否则会造成不收敛，梯度下降法也就无法很好地表现。
- **共轭梯度法计算简单，可以控制收敛精度，利于实现，迭代次数远小于批量梯度下降法**，在满足共轭梯度的条件时，是很好的方法。

参考文献

共轭梯度法计算回归——邱怡轩

Conjugate gradient method - Wikipedia

python_numpy最小二乘法的曲线拟合

《统计学习方法》——李航

Andrew Ng Coursera机器学习课程

附录：源代码

详见解压文件中的LSE.py, Gradient_Descent.py, Conjugate_gradient.py