2020 年春季学期

计算学部《机器学习》课程

Lab 1 实验报告

姓名	王科龙
学号	1180801203
班号	1803104
电子邮件	1264405807@qq.com
手机号码	19917620613

目录

1 问题描述

1.1

2 数学原理

2.1 解析解

(1)解析解不带正则项

设带噪音的观测数据为$\boldsymbol{T}=[t_1,t_2,\dots,t_n]^T$,自变量为$\boldsymbol{X}=[x_1,x_2,\dots,x_n]^T$,$\boldsymbol{\omega}=[\omega_0,\omega_1,\dots,\omega_m]^{\boldsymbol{T}}$,则误差函数可表示为$E(\boldsymbol{\omega})=\frac{1}{2}(\boldsymbol{Y}-\boldsymbol{T})^T(\boldsymbol{Y}-\boldsymbol{T})$，其中$\boldsymbol{Y}=\overline{\boldsymbol{X}}\boldsymbol{w}$。

$\overline{\boldsymbol{X}}$为$\boldsymbol{X}$的范德蒙行列式形式的矩阵，$\overline{\boldsymbol{X}}$=$\begin{pmatrix} x_1^0 & \cdots & x_1^m \\ x_2^0 & \cdots & x_2^m \\ \vdots & \ddots & \vdots \\ x_n^0 & \cdots & x_n^m \end{pmatrix}$

求解$\frac{\partial E(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}}=0$,得

$$\boldsymbol{\omega}=(\overline{\boldsymbol{X}}^{\boldsymbol{T}}\overline{\boldsymbol{X}})^{-\boldsymbol{1}}\overline{\boldsymbol{X}}^{\boldsymbol{T}}\boldsymbol{T}$$

(2)解析解带正则项

携带正则项后，$E(\boldsymbol{\omega})=\frac{1}{2}(\boldsymbol{Y}-\boldsymbol{T})^T(\boldsymbol{Y}-\boldsymbol{T})+\frac{\lambda}{2}\boldsymbol{\omega}^{\boldsymbol{T}}\boldsymbol{\omega}$，求解$\frac{\partial E(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}}=0$得

$$\boldsymbol{\omega}=(\overline{\boldsymbol{X}}^{\boldsymbol{T}}\overline{\boldsymbol{X}}+\lambda\boldsymbol{E})^{-\boldsymbol{1}}\overline{\boldsymbol{X}}^{\boldsymbol{T}}\boldsymbol{T}$$

2.2 迭代法

(1)梯度下降法

设方程为$\boldsymbol{Ax}=\boldsymbol{b}$，其中$\boldsymbol{A}$为对称正定矩阵，该方程的解可以等价为一个二次函数$f(\boldsymbol{x})=\frac{1}{2}(\boldsymbol{Ax},\boldsymbol{x})-(\boldsymbol{b},\boldsymbol{x})$的最小值，因此建立求解$f(\boldsymbol{x})$的最小值的迭代法，即可求得$\boldsymbol{Ax}=\boldsymbol{b}$的解。

由于梯度是某一个点的最快上升方向，因此若想要到达二次函数的最小值点，沿梯度反方向下降是较好的一个选择，从而有迭代式：

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\alpha^{(k)}\boldsymbol{r}^{(k)}$$

其中$\boldsymbol{r}^{(k)}$为第 k 次迭代时$\boldsymbol{x}^{(k)}$点的负梯度，$\alpha^{(k)}$为步长。

在实际过程中，有可能$\alpha^{(k)}$选择的过大，使得$\boldsymbol{x}$错过了最小值点，此时需要将步长减半，减缓下降速度以逼近最小值点。

(2)共轭梯度下降法

共轭梯度下降法比梯度下降法快的多，在每次选择下降方向时，选择该点的共轭方向，经过有限步迭代即可找到最小值。

迭代过程为

$$\boldsymbol{x}^{(0)} \in \boldsymbol{R}^n, \boldsymbol{p}^{(\boldsymbol{0})} = \boldsymbol{r}^{(0)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(0)}$$

$$\begin{cases} \alpha^{(k)} = \dfrac{\boldsymbol{p}^{(k)^{T}}\boldsymbol{r}}{\boldsymbol{p}^{(k)^{T}}\boldsymbol{A}\boldsymbol{p}^{(k)}} \\ \boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha^{(k)}\boldsymbol{p}^{(k)} \\ \boldsymbol{r}^{(k+1)} = \boldsymbol{r}^{(\boldsymbol{k})} - \alpha^{(k)}\boldsymbol{A}\boldsymbol{p}^{(\boldsymbol{k})} \\ \beta^{(k)} = \dfrac{(\boldsymbol{r}^{(k+1)}, \boldsymbol{r}^{(k+1)})}{(\boldsymbol{r}^{(k)}, \boldsymbol{r}^{(k)})} \\ \boldsymbol{p}^{(k+1)} = \boldsymbol{r}^{(k+1)} + \beta^{(k)}\boldsymbol{p}^{(k)} \end{cases}$$

3 实验做法

3.1 生成数据

在解析解中，拟合了曲线$\sin(2x)$,使用 np.random.random()生成属于$[0,1]$的n 个数据，在乘以2π使区间达到$[0,2\pi]$

在优化解中，由于$[0,2\pi]$这个区间过大容易导致溢出，因此改拟合曲线$\sin(2\pi x)$，数据区间$[0,1]$

$y0 = \sin(2x_0)$ 或 $y_0 = \sin(2\pi x_0)$

加噪使用了 randn,加入高斯噪音 noise

$y = y_0 + noise$

3.2 解析解

```python
# 解析法求解 loss 的最优解，无正则项

# w=(x 转置*X)^(-1)*x 转置*t  x 为范德蒙行列式转置形式，t 为带有噪音的观测数据
def best_answer_without_correct(x0, t, m):
    x = vandermonde(x0, m)
    res = np.linalg.inv(x.T @ x) @ x.T @ t.reshape(t.shape[0], 1)
    return res
```

```python
# 解析法求解 loss 的最优解，有正则项

# w=(x 转置*x+lambda*E)^(-1)*x 转置*t
def best_answer_with_correct(x0, t, m, l_lambda):
    x = vandermonde(x0, m)
    return np.linalg.inv(x.T @ x + l_lambda * np.eye(m, m)) @ x.T @ t
```

其中 x = vandermonde(x0, m)是求解$\overline{X}$矩阵的方法

3.3 梯度下降

```python
# k 最多的迭代次数

def grad_down(w0, A, b, k, x, t):
    w = np.mat(w0)
    A = np.mat(A)
    b = np.mat(b)
    step_length = 0.001  # 步长
    r = np.mat(b - A * w)  # 梯度反方向
    value1 = get_Ew(x, w, t)  # 第二个
    value0 = value1
    precision = 1e-5
    for _ in range(k):
        w = w + step_length * r
        value0 = value1  # 前一个
        value1 = get_Ew(x, w, t)  # 当前的
        # 如果二次函数取值变大，则减小步长
        if value1 - value0 > 0: step_length = 0.5 * step_length
        # 跳出条件 当 loss 足够小时
        if value1 < precision:
            print(_)
            break
        r = np.mat(b - A * w)  # 梯度反方向
    return w
```

3.4 共轭梯度下降法

```python
def conjugate_down(w0, A, b):
```

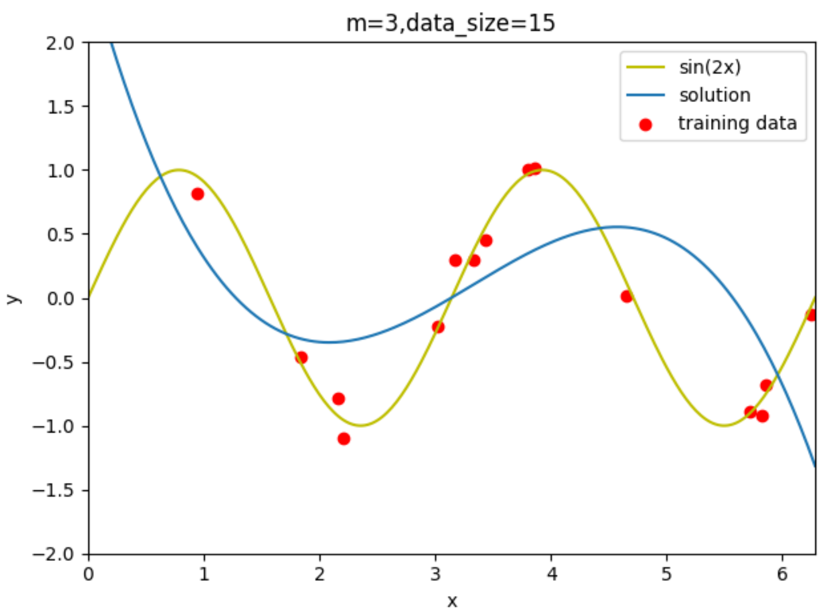
```
w = np.mat(w0)
A = np.mat(A)
b = np.mat(b)
r = np.mat(b - A * w)
p = r
precision = 1e-5
for i in range(A.shape[1] + 1):
    alpha = float((p.T @ r) / (p.T @ A @ p))  # alpha(k)
    w = w + alpha * p  # w(k+1)= alpha(k)*p(k)
    r_prev = r  # 记录r(k-1)
    r = r - alpha * A * p  # r(k+1)=r(k)-alpha(k)*p(k)
    if r.T @ r < precision: break  # 精度要求
    beta = float((r.T @ r) / (r_prev.T @ r_prev))
     # beta(k) =[r(k+1).T @ r(k+1)]/[r(k).T @ r(k)]
    p = r + beta * p  # p(k+1)=r(k+1)+beta*p(k)
return w
```

4 实验结果分析

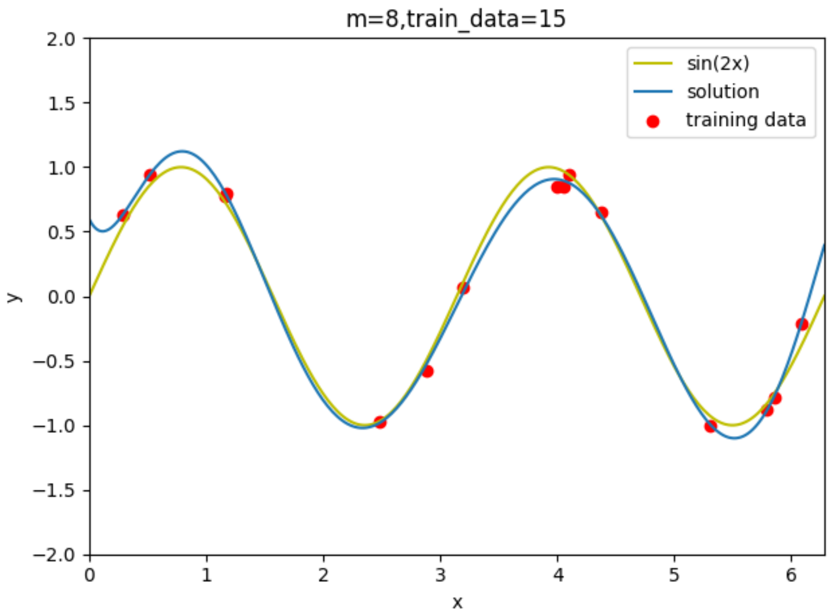
4.1 解析解拟合效果

(1)不带正则项

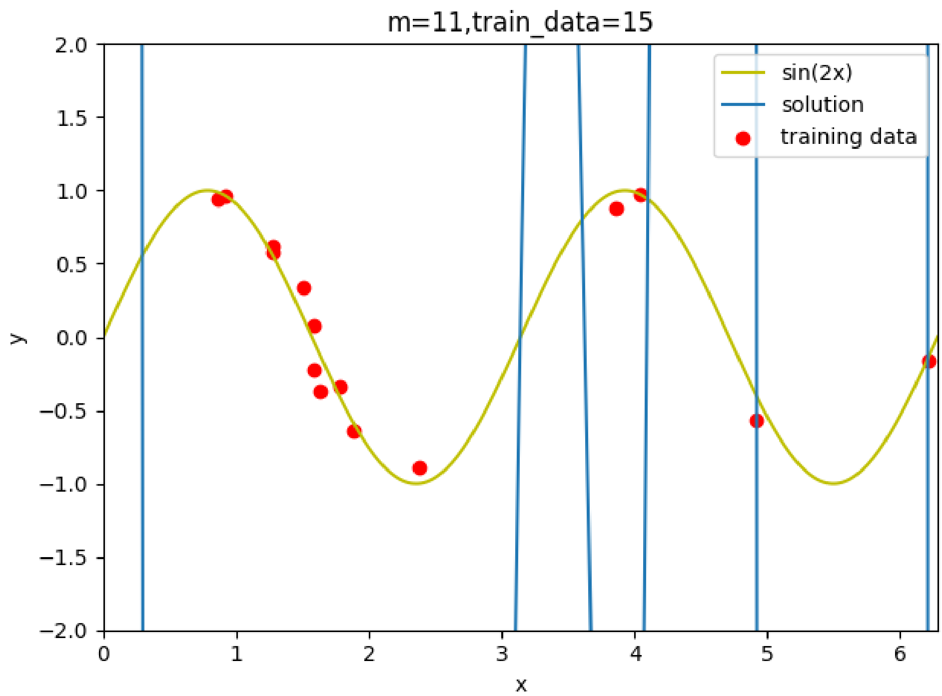
m=3，从图中可以看出阶数太小，出现了欠拟合现象



m=8，拟合效果刚好

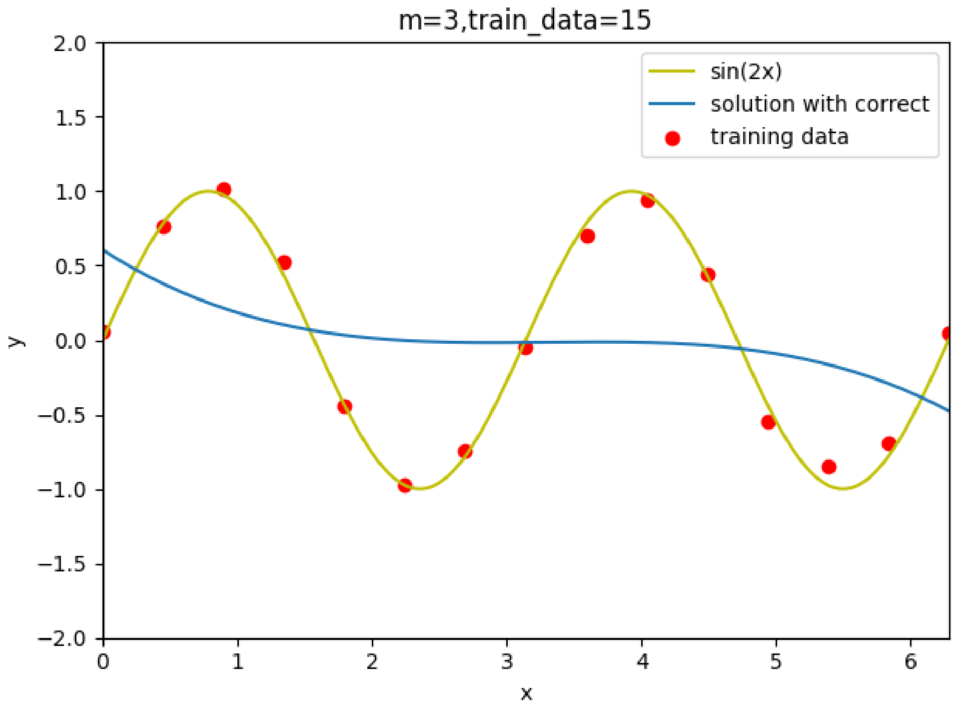


m=11,出现了过拟合现象

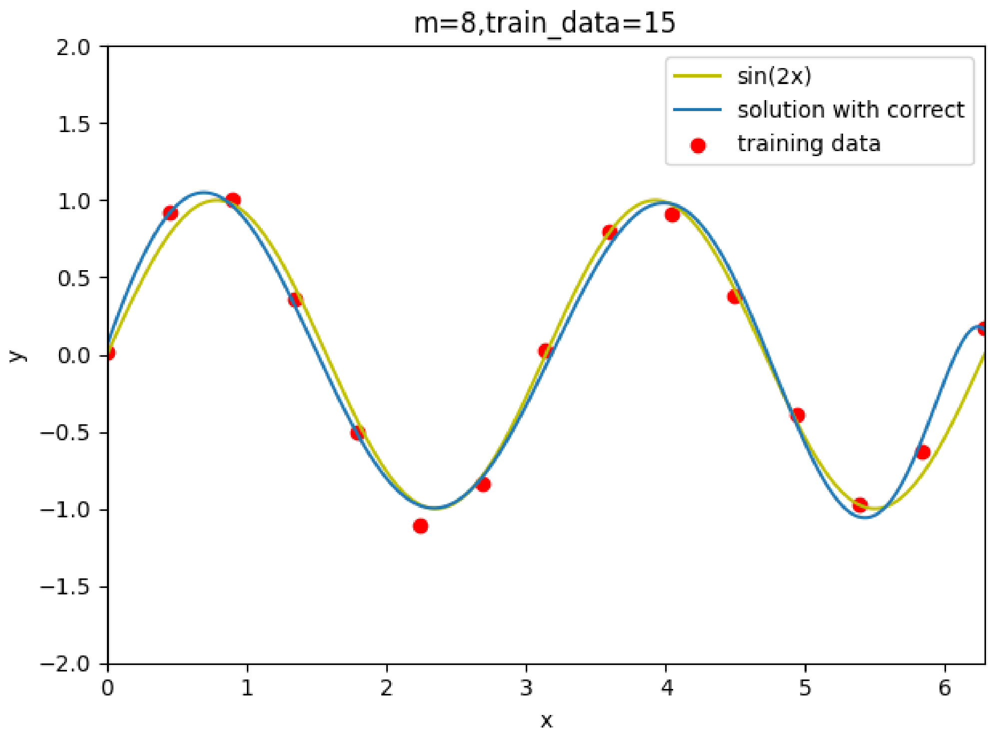


(2)带修正项

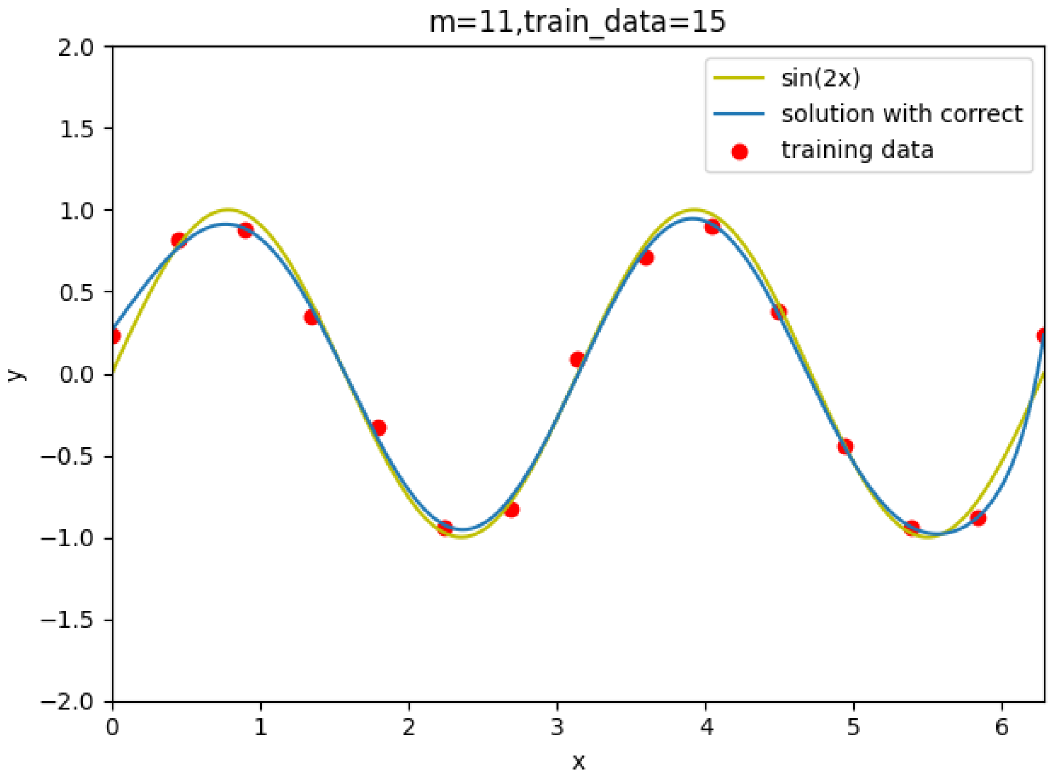
m=3，欠拟合



m=8，拟合效果较好



m=11,过拟合现象得到缓解

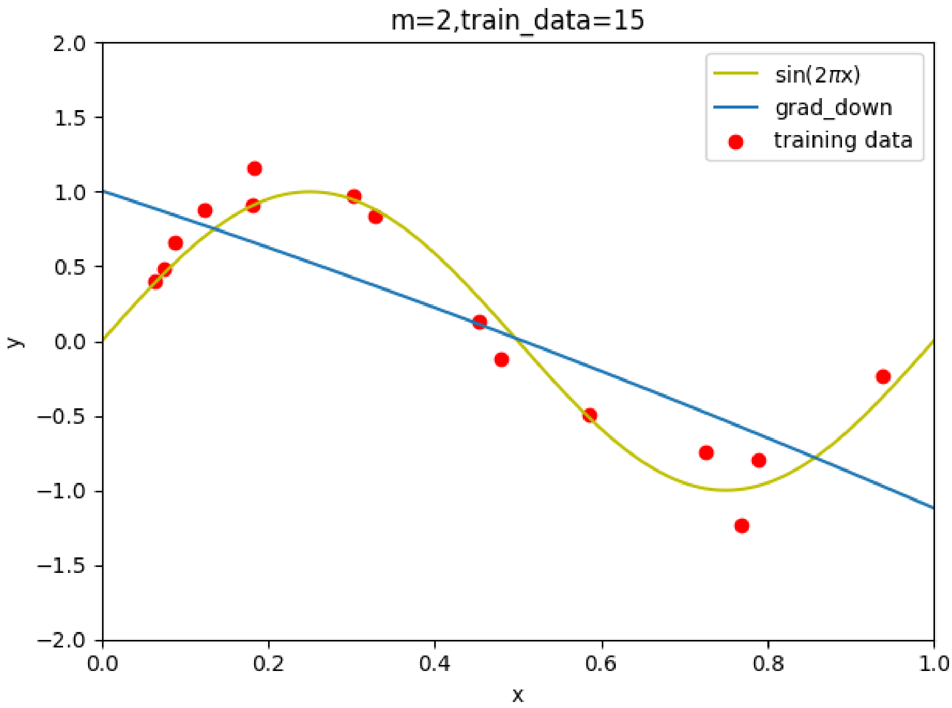


4.2 梯度下降法拟合效果

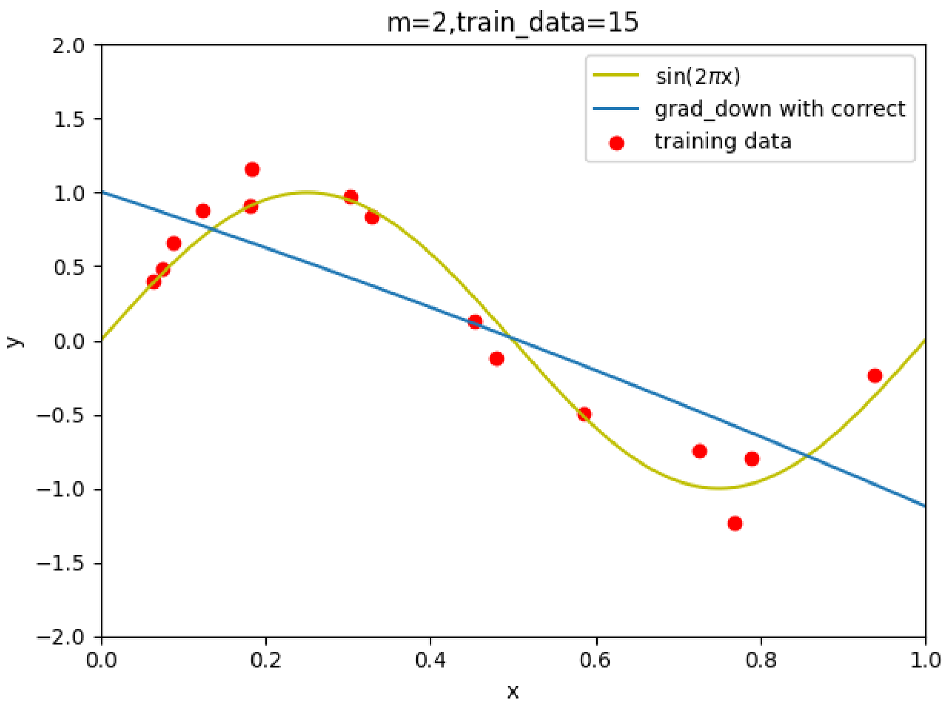
λ选择e^{-9}

(1) m=2, train_size=15, test_size=40

不带正则项



带正则项

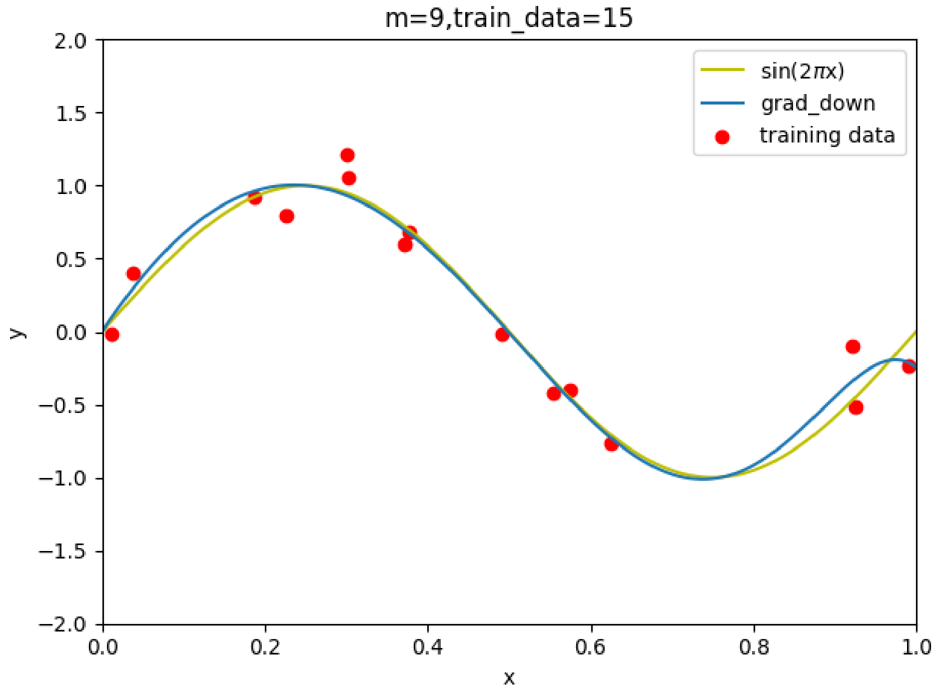


测试集上的E_{RMS}

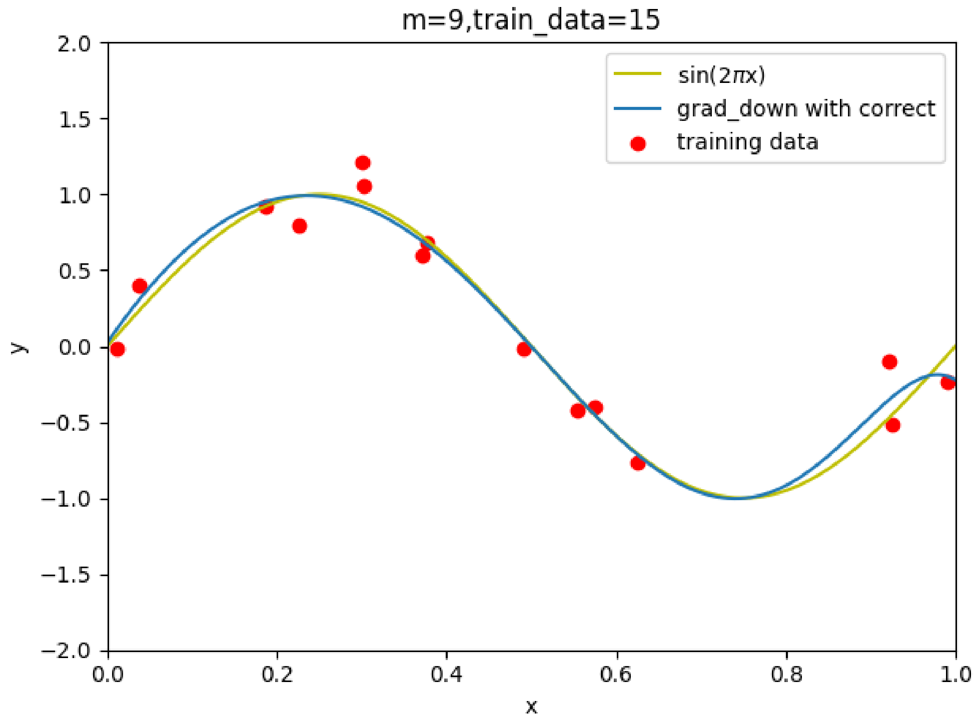
```
without correct in test data:e_rms= [[0.51503461]]
with correct in test data:e_rms= [[0.51501398]]
```

(2) m=9, train_size=15, test_size=40

不带正则项



带正则项

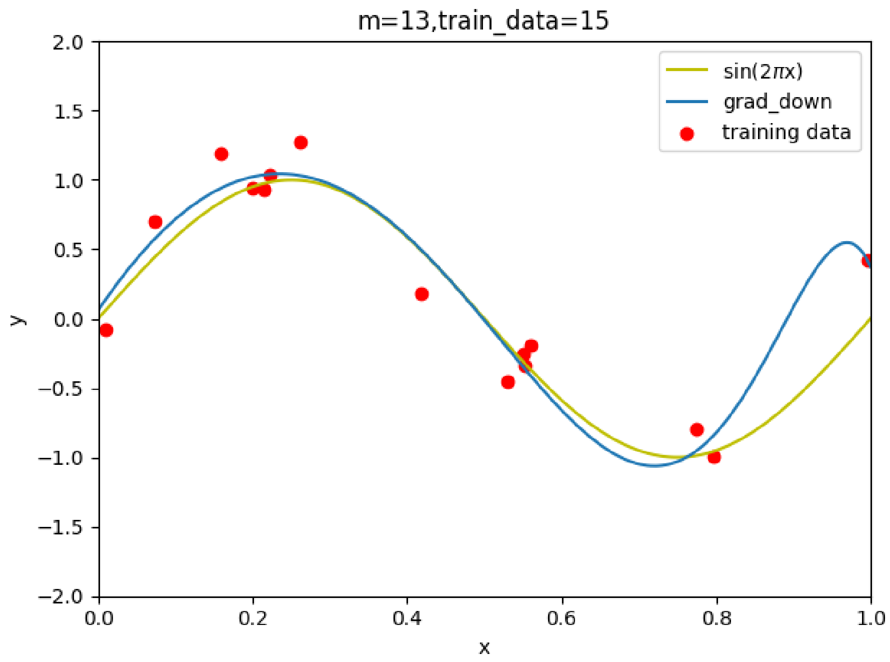


测试集上的E_{RMS}

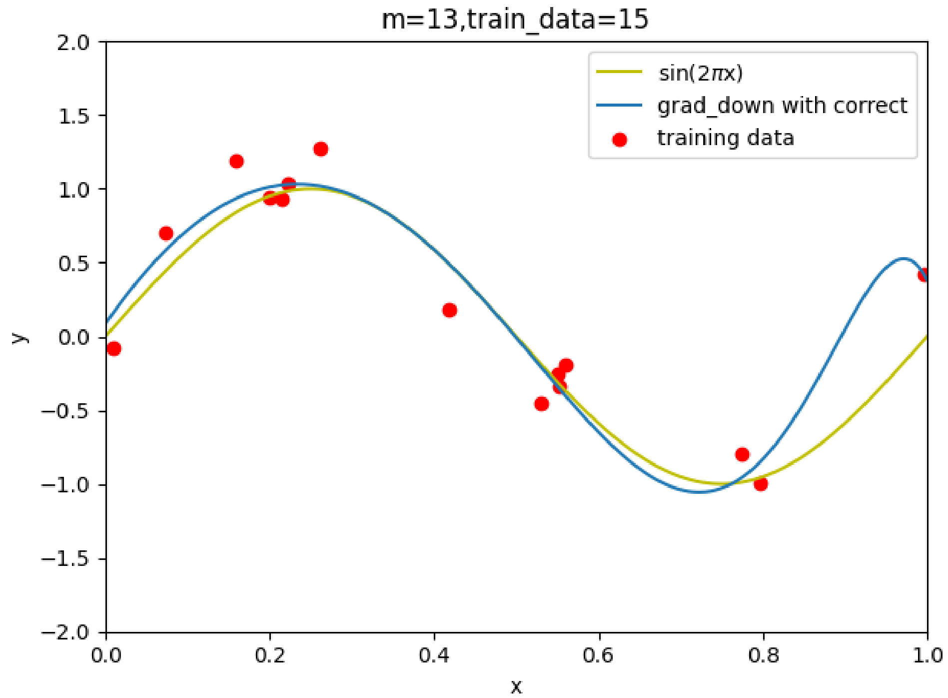
```
without correct in test data:e_rms= [[0.25535464]]
with correct in test data:e_rms= [[0.25601732]]
```

(3) m=13, train_size=15, test_size=40

不带正则项



带正则项



测试集上的E_{RMS}

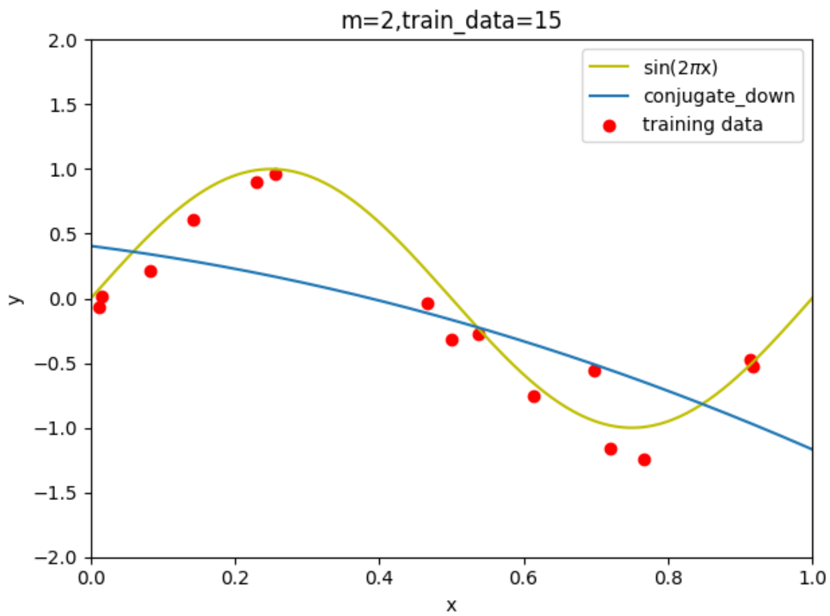
```
>>> runfile('E:/桌面文件/文档/机器学习/机器学习实验/lab1/la
without correct in test data:e_rms= [[0.35736713]]
with correct in test data:e_rms= [[0.34645851]]
```

4.3 共轭梯度法拟合效果

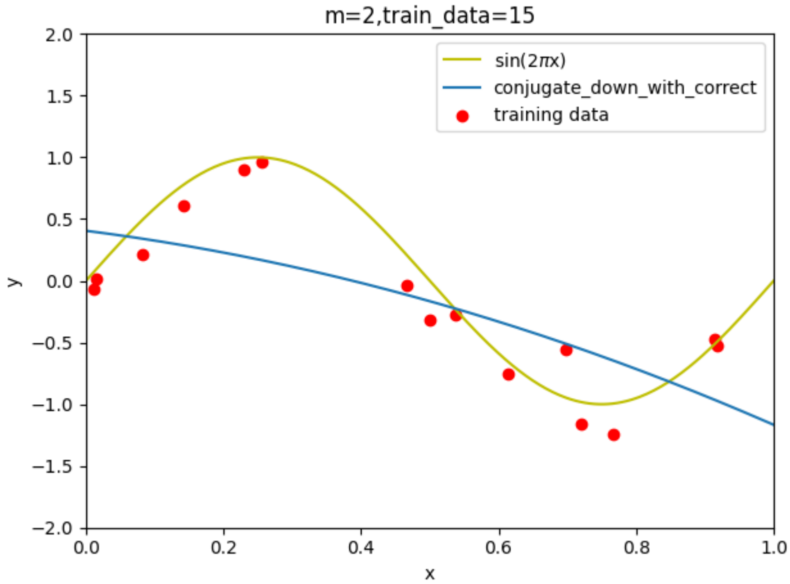
λ选择e^{-9}

(1) m=2, train_size=15, test_size=40

不带正则项



带正则项

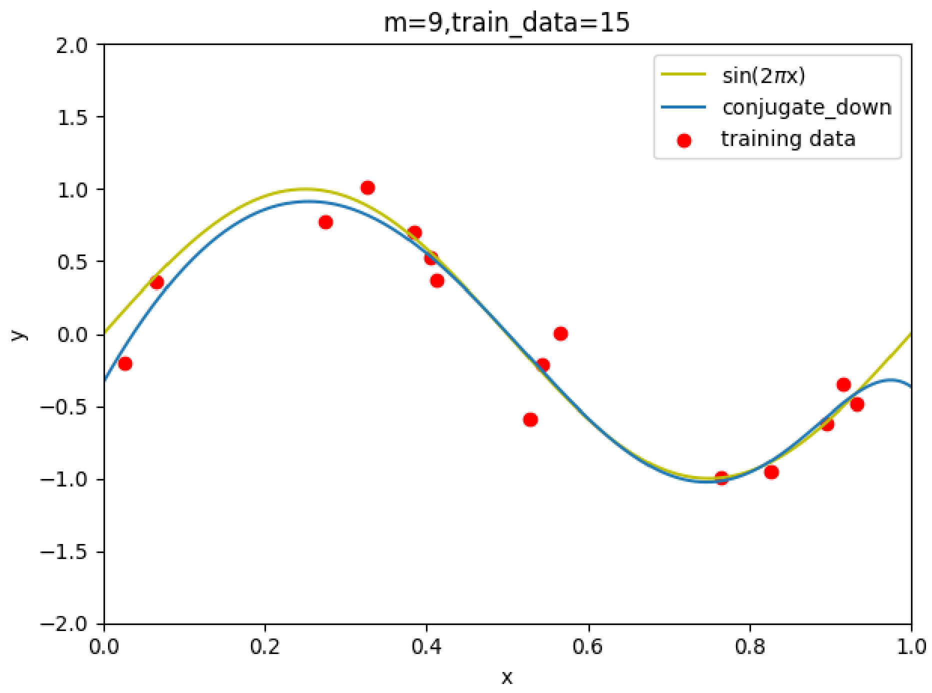


测试集上的E_{RMS}

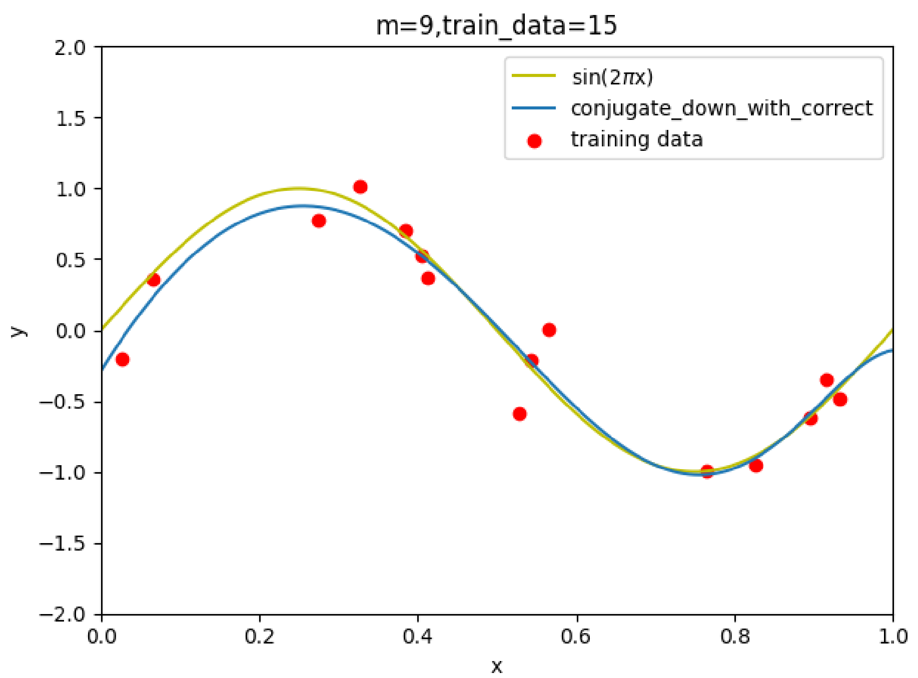
```
>>> runfile('E:/桌面文件/文档/机器学习/机器学习实验/lab1/
without correct in test data:e_rms= [[0.6359433]]
with correct in test data:e_rms= [[0.63594092]]
```

(2) m=9, train_size=15, test_size=40

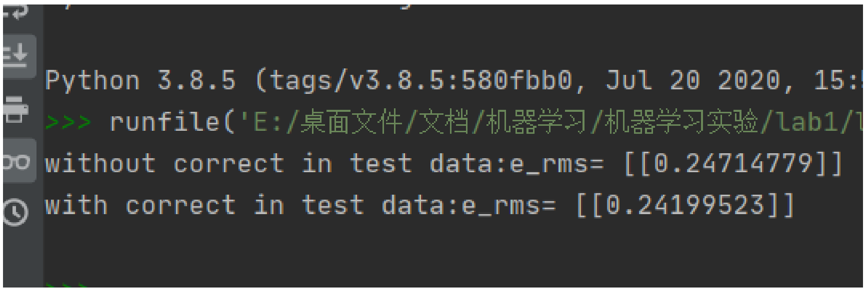
不带正则项



带正则项

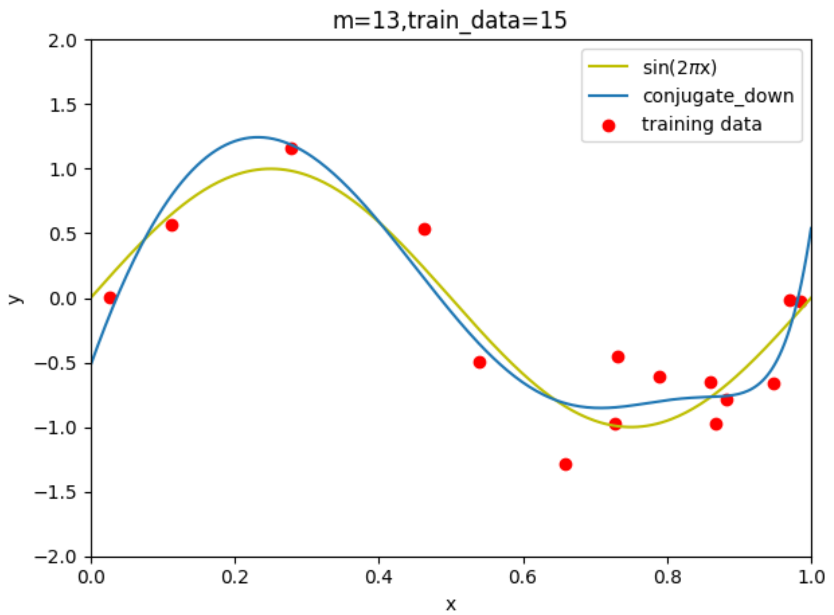


测试集上的E_{RMS}

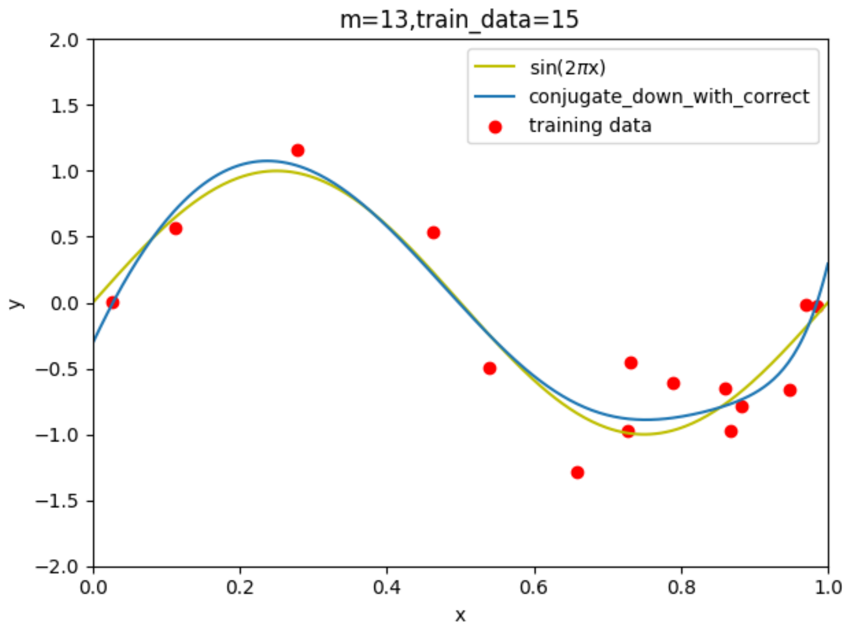


(3) m=13, train_size=15, test_size=40

不带正则项



带正则项



测试集上的E_{RMS}

```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:5
>>> runfile('E:/桌面文件/文档/机器学习/机器学习实验/lab1/l
without correct in test data:e_rms= [[0.26776481]]
with correct in test data:e_rms= [[0.22819272]]
```

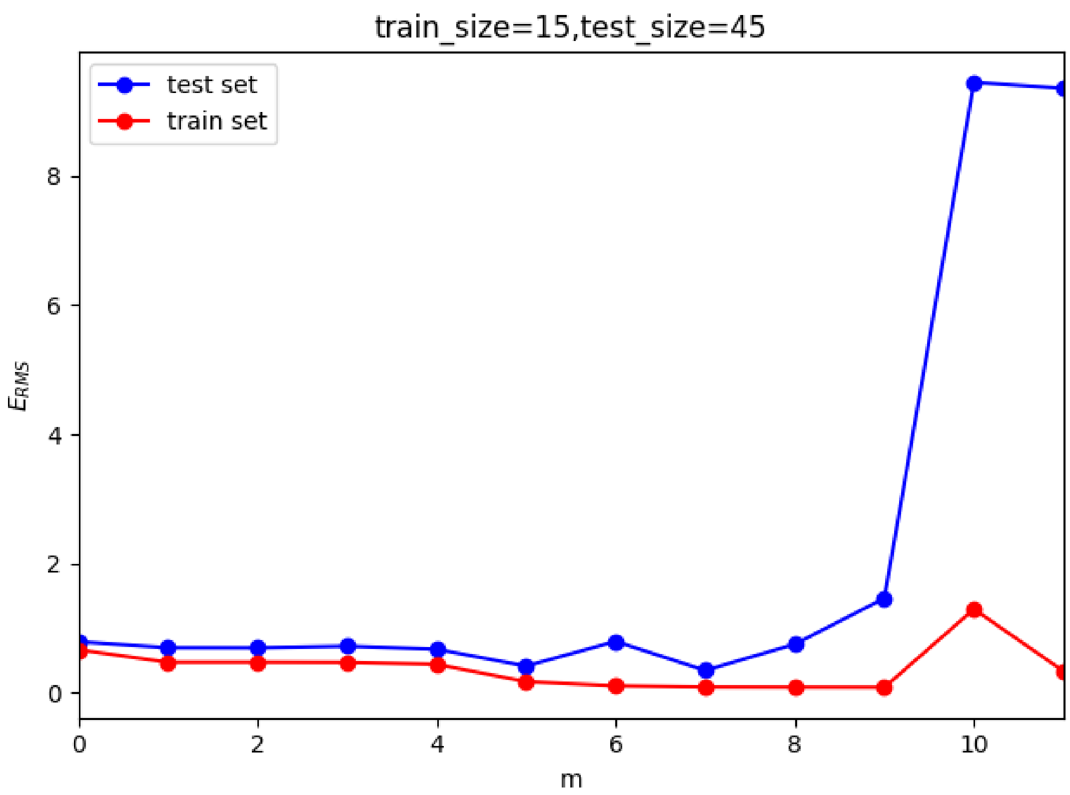
4.4 观察过拟合现象

过拟合现象是由于过于学习训练数据的特征造成的，模型不仅学习了训练数据的一般特征，同时将数据的特殊特征也学习到了，从而变成了“死记硬背”,泛化能力也大打折扣。

经常用来控制过拟合现象的一个方法是增加一个正则项，使得系数不会达到一个很大的值。

用E_{RMS}表示过拟合和欠拟合现象

在 train_size=15,test_size=45 的情况下，使用不同的阶数拟合曲线得到的结果如下:



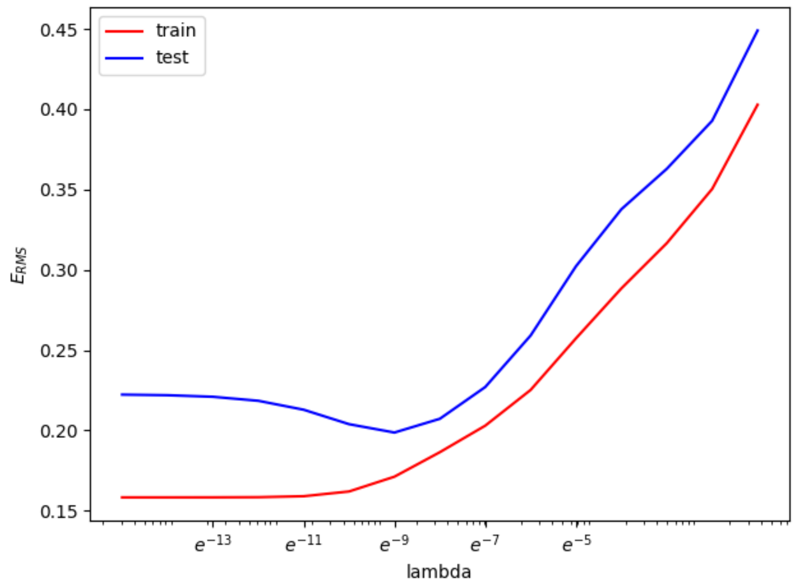
从图中可以看出，当 m 取值在 0 到 5 之间时，对应的E_{RMS}值较大，是欠拟合的表现，当 m 取值在 5 到 8 时，拟合效果刚刚好，但是当 m 到于 9 之后，

E_{RMS}值在 test data 上极具增大，表示模型已经失去了泛化能力。

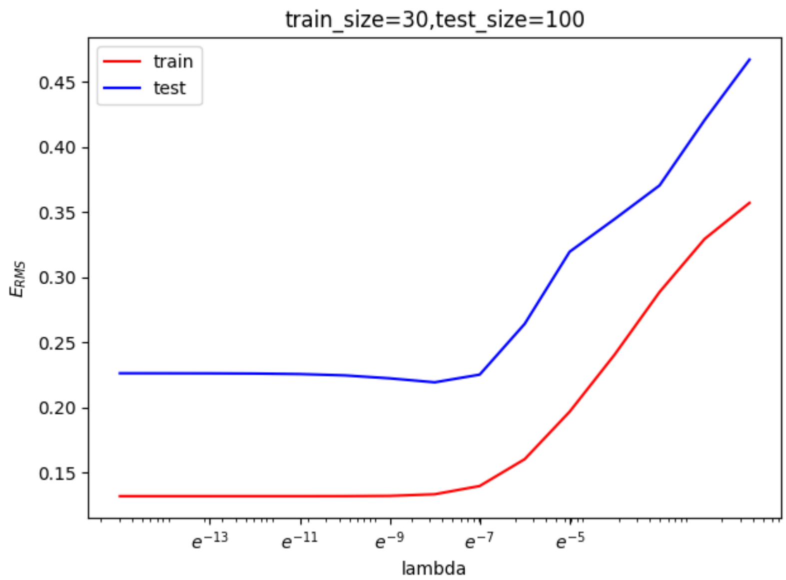
4.5 正则项中λ的选取

为了取得较好的惩罚效果，需要选择一个合适的λ，
观察不同的数据量下不同λ下E_{RMS}的值

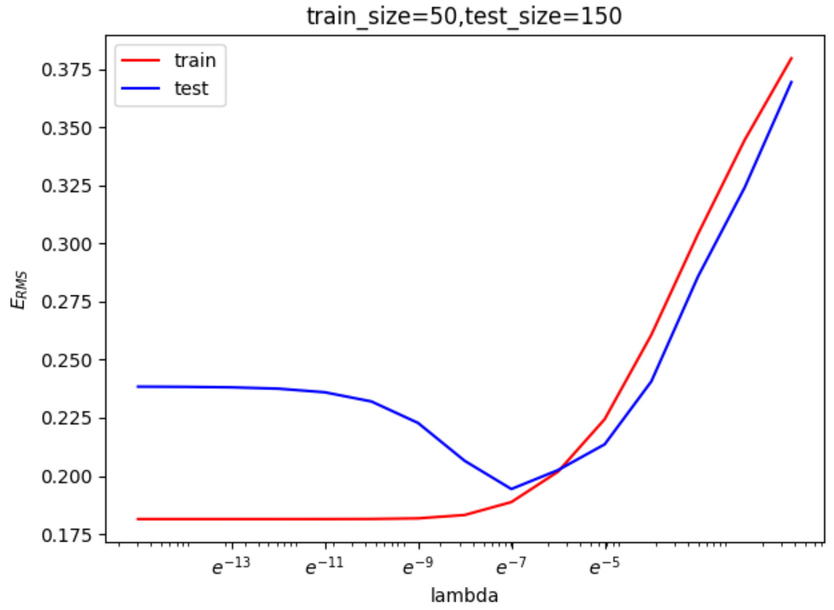
(1). m=12, train_size=15 test_size=40



(2) m=27, train_size=30, test_size=100



(3) m=47, train_size=50, test_size=150

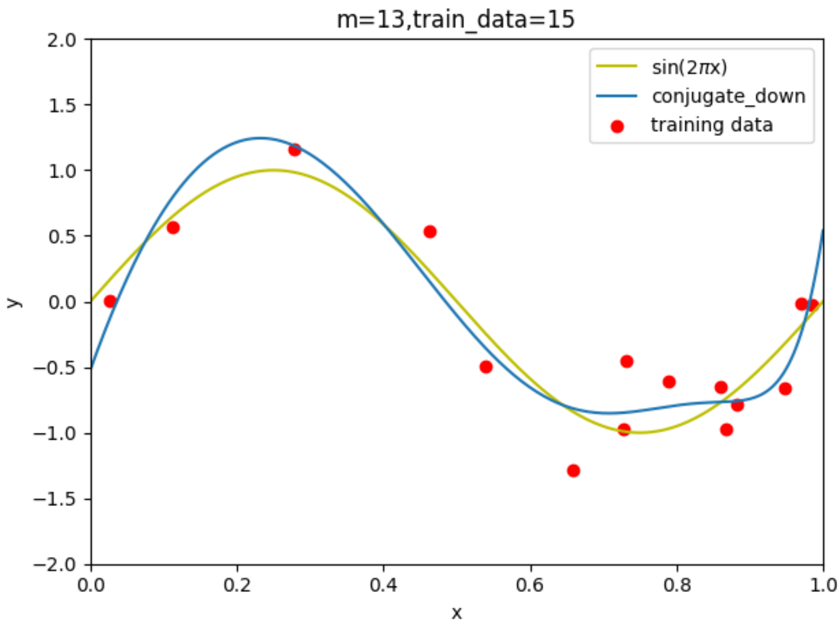


从三幅图中可以看出，选择$\lambda = e^{-9}, e^{-8}, e^{-7}$整体的$E_{RMS}$较小，比较合适。

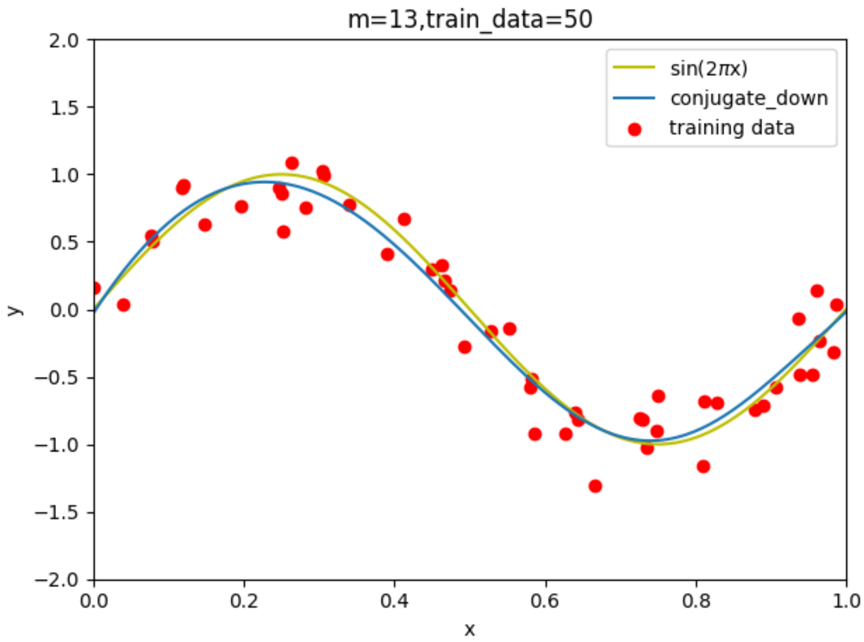
4.6 数据量比较

确定 m=13，在不同的数据量下比较拟合结果

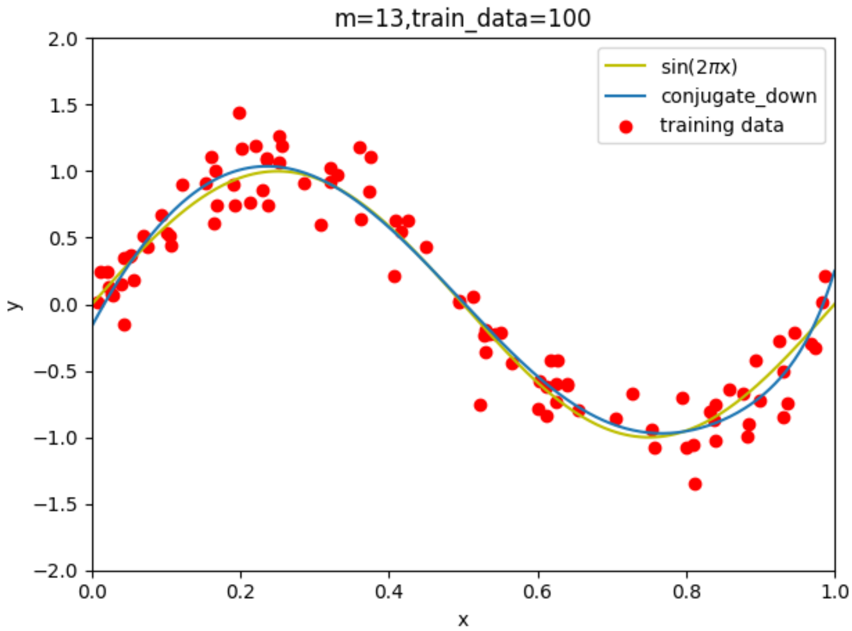
当 train_size=15 时



当 train_size=50 时



当 train_size=100 时



从图中可以看出对于相同的阶数，不同的数据规模的精度也是不相同的，增大数据规模也可以缓解过拟合现象

5 结论

(1)当阶数较小时，会出现欠拟合现象

(2)当阶数过大时，会出现过拟合现象

(3)通过增加正则项或者增大数据规模可以缓解过拟合现象

(4)不同的λ对模型的修正能力不同，缓解过拟合的能力也不同

(5)梯度优化是一种可行的方法，但是当数据量过多时，计算的时间不可接受

(6)共轭梯度下降法时一种较快的迭代法，可以在有限步内找到优化解