哈尔滨工业大学计算机科学与技术学院

机器学习实验报告

实验题目	姓名	学号	联系方式
多项式拟合正弦函数	崔路源	1163300402	997898829@qq.com

1. 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，并掌握克服过拟合的方法(如加惩罚项、增加样本)

2. 实验要求及实验环境

2.1. 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。

2.2. 实验环境

- matlab-2017b

3. 设计思想

本次实验的目标是对正弦函数 $\sin 2\pi x$ 进行拟合，即根据随机生成的数据集 (x_i, y_i) 来找到一条曲线，使其能够根据给定的 x 很好的预测出相应的 y。使用机器学习的相关知识在之后也可以模拟更多的函数曲线。

3.1. 算法原理

3.1.1. 预处理

1. 因为此实验拟合的是正弦函数，可以利用泰勒展开式来拟合相应的参数，因此假设拟合的多项式为：

$$h(a,x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{1}$$

用矩阵的方式来表示为

$$x = \begin{bmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^n \end{bmatrix}_{(n+1)\times 1} \qquad a = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}_{(n+1)\times 1}$$

则多项式的矩阵形式：

$$h(a,x) = a^T x \tag{2}$$

2. 代价函数，对于 m 组数据，则令

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \dots & x_m^n \end{bmatrix}_{m\times(n+1)} \qquad Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_m \end{bmatrix}_{m\times 1}$$

那么代价函数则为

$$\begin{aligned} J(a) &= \frac{1}{2m}\sum_{i=1}^{m}(h(x_{(i)}) - y_{(i)})^2 \\ &= \frac{1}{2m}\sum_{i=1}^{m}(a^T x_{(i)} - y_{(i)})^2 \\ &= \frac{1}{2m}(Xa - Y)^T(Xa - Y) \end{aligned} \tag{3}$$

3.1.2. 最小二乘法

最小二乘法（又称最小平方法）是一种数学优化技术，它通过最小化误差的平方和寻找数据的最佳函数匹配。此方法虽然在拟合曲线时可以得到解析解，但是由于最小二乘法需要计算矩阵的逆，因此计算量很大，根据经验来看，当数据的特征数大于 10000 时，最小二乘法的性能就会变的很差。

最小二乘法的数学推导：

1. 先对代价函数进行处理

$$\begin{aligned} J(a) &= \frac{1}{2m}(Xa - Y)^T(Xa - Y) \\ &= \frac{1}{2m}(a^T x^T x a - 2a^T x^T Y + Y^T Y) \end{aligned}$$

2. 代价函数对 a 求导

$$\frac{\partial J(a)}{a} = 0$$
$$X^T X a - X^T Y = 0$$

3. 得出最后的解析解

$$a = (X^T X)^{-1} X^T Y \tag{4}$$

3.1.3. 加惩罚项的最小二乘法

根据奥卡姆剃刀原理，所有的模型中，能很好的解释已知的数据，并且十分简单的模型才是最好的，而且为了防止过拟合现象的发生，因此我们在优化目标上加上惩罚项。

加入惩罚项之后，代价函数变为：

$$J(a)=\frac{1}{2m}\sum_{i=1}^{m}(a^Tx_{(i)}-y_{(i)})^2+\frac{\lambda}{2m}||a||^2 \tag{5}$$

其中，λ为惩罚系数。

之后与最小二乘法的处理一样，对 $J(a)$ 求偏导，是偏导数为0，最后得出

$$a=(X^TX+\lambda I_{(n+1)\times(n+1)})^{-1}X^TY \tag{6}$$

3.1.4. 梯度下降法

梯度下降法（Gradient descent）是一个一阶最优化算法，通常也称为最速下降法。 要使用梯度下降法找到一个函数的局部极小值，必须向函数上当前点对应梯度（或者是近似梯度）的反方向的规定步长距离点进行迭代搜索。梯度下降法利用迭代实现每一步的局部最优，从而最终得到最优解。

1. 梯度下降法的数学推导：

由之前 公式(3) 可知代价函数，则之后利用梯度下降的思想来逐步的调整 a 的取值，并写成矩阵的形式则为

$$a=a-\frac{\alpha}{m}X^T(Xa-Y) \tag{7}$$

其中，α 是学习的速率，也叫做步长

2. 学习速率的选择：

学习速率的选择是一件非常麻烦的事情，如果学习速率设置过大，就会导致越过最小值的事情发生，最后使得函数无法收敛；如果学习速率设置过小，就会使得程序运行过慢，迭代次数过多。并且对待不同的数据量，学习速率的选择也不一样。

3.1.5. 加惩罚项的梯度下降法

与加入惩罚项的最小二乘法类似，加入惩罚项的目的都是为了防止过拟合现象的发生，加入惩罚项之后的梯度下降的公式为

$$a=a-\frac{\alpha}{m}X^T(Xa-Y)-\frac{\lambda}{m}a \tag{8}$$

其中，λ为惩罚系数。

通过加入惩罚系数可以有效的防止过拟合的发生

3.1.6. 共轭梯度法

共轭梯度法是一种介于梯度下降法和牛顿法之间的无约束优化算法，具有超线性收敛速度，而且结构简单。共轭梯度法只用到了目标函数及其梯度值，避免了二阶导数的计算，降低了计算量和存储量。

在各种优化算法中，共轭梯度法是非常重要的一种。其优点是所需存储量小，具有步收敛性，稳定性高，而且不需要任何外来参数。

共轭梯度法将方向限制在初始点的共轭方向空间内，而不是像梯度下降法那样随意，于是有更好的优化效果。

共轭梯度法的算法过程：

$$
\begin{aligned}
&B = X^T X \\
&a_{(0)} = 0 \\
&r_0 = X^T t - B a_{(0)} \\
&p_0 = r_0 \\
&k = 0 \\
&while true: \\
&\qquad \alpha = \frac{r_k^T r_k}{p_k^T B p_k} \\
&\qquad a_{k+1} = a_k + \alpha_k p_k \\
&\qquad r_{k+1} = r_k - \alpha_k B p_k \\
&\qquad \text{如果} r_{k+1} \text{足够小，那么就停止迭代} \\
&\qquad \beta_k = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \\
&\qquad p_{k+1} = r_{k+1} + \beta_k p_k \\
&\qquad k = k + 1
\end{aligned}
$$

输出结果 a_{k+1}

3.1.7. 加惩罚项的共轭梯度法

加入惩罚项的共轭梯度法的算法过程只有一开始的矩阵不同：

$$B = X^T X + \lambda I_{(n+1)\times(n+1)} \tag{9}$$

其他的迭代与之前的共轭梯度法相似

3.2. 算法的实现

我主要是使用 matlab 来实现算法的，在此列出代码的核心函数部分

3.2.1. 生成数据，加入噪声

```
%生成(0,2)随机的x
xx = cell(1,3);
for i=1:3
    xx{1,i} = sortrows(rand(n,1)*2);
    n = n/2;
end
%生成对应的y值,生成服从正态分布(0,sigma^2)的噪声
yy = cell(1,3);
```

```
n = 100;
for i=1:3
    noise = normrnd(0,sigma,n,1);
    yy{1,i} = sin(2*pi*xx{1,i}) + noise;
    n = n/2;
end
```

3.2.2. 最小二乘法

不带惩罚项

```
A{1,i} = pinv(XZ{1,i}'*XZ{1,i})*XZ{1,i}'*yy{1,1};
```

带惩罚项

```
A{1,i} = pinv(XZ{1,i}'*XZ{1,i}+lambda*eye(M(i)+1))*XZ{1,i}'*yy{1,1};
```

其中，因对于不同的多项式的次数，所以公式中含有 i，公式本身为：

```
不带惩罚项
a = pinv(xT * x) * xT * y;
带惩罚项
a = pinv(xT * x + λI) * xT * y;
```

3.2.3. 梯度下降法

不带惩罚项

```
function [a,c] = gradient(a,lambda,X,y)
c = 0;
while true
    a_o = a;
    a = a - lambda /100 * X'*(X*a - y);
    c = c + 1;
    if ( (max(a_o - a) )< 10^(-6))
        break;
    end
end
end
```

带惩罚项

```
function [a,c] = gradient(a,lambda,X,y,mu)
c = 0;
while true
    a_o = a;
```

```
    a = a - lambda / 100 * X'*(X*a - y)-mu / 100 *a;
    c = c + 1;
    if ( (max(a_o - a) )< 10^(-3))
        break;
    end
end
end
```

3.2.4. 共轭梯度法

不带惩罚项

```
function [a,c] = Con_Gradient(a,X,y)
B = X'* X;
r = X'* y - B * a;
p = r;
c = 0;

while true
    alpha = (r'* r)/(p'* B * p);
    a = a + alpha * p;
    r_o = r;
    r = r - alpha * B * p;
    if (r<10^(-6))
        break;
    end
    beta = (r'* r)/(r_o' * r_o);
    p = r + beta * p;
    c = c + 1;
end
end
```

带惩罚项

```
function [a,c] = Con_Gradient(a,X,y,lambda,num)
B = X'* X + lambda * eye(num);
r = X'* y - B * a;
p = r;
c = 0;

while true
    alpha = (r'* r)/(p'* B * p);
    a = a + alpha * p;
    r_o = r;
    r = r - alpha * B * p;
    if (r<10^(-6))
```

```
            break;
        end
        beta = (r'* r)/(r_o' * r_o);
        p = r + beta * p;
        c = c + 1;
    end
    end
```

4. 实验结果与分析

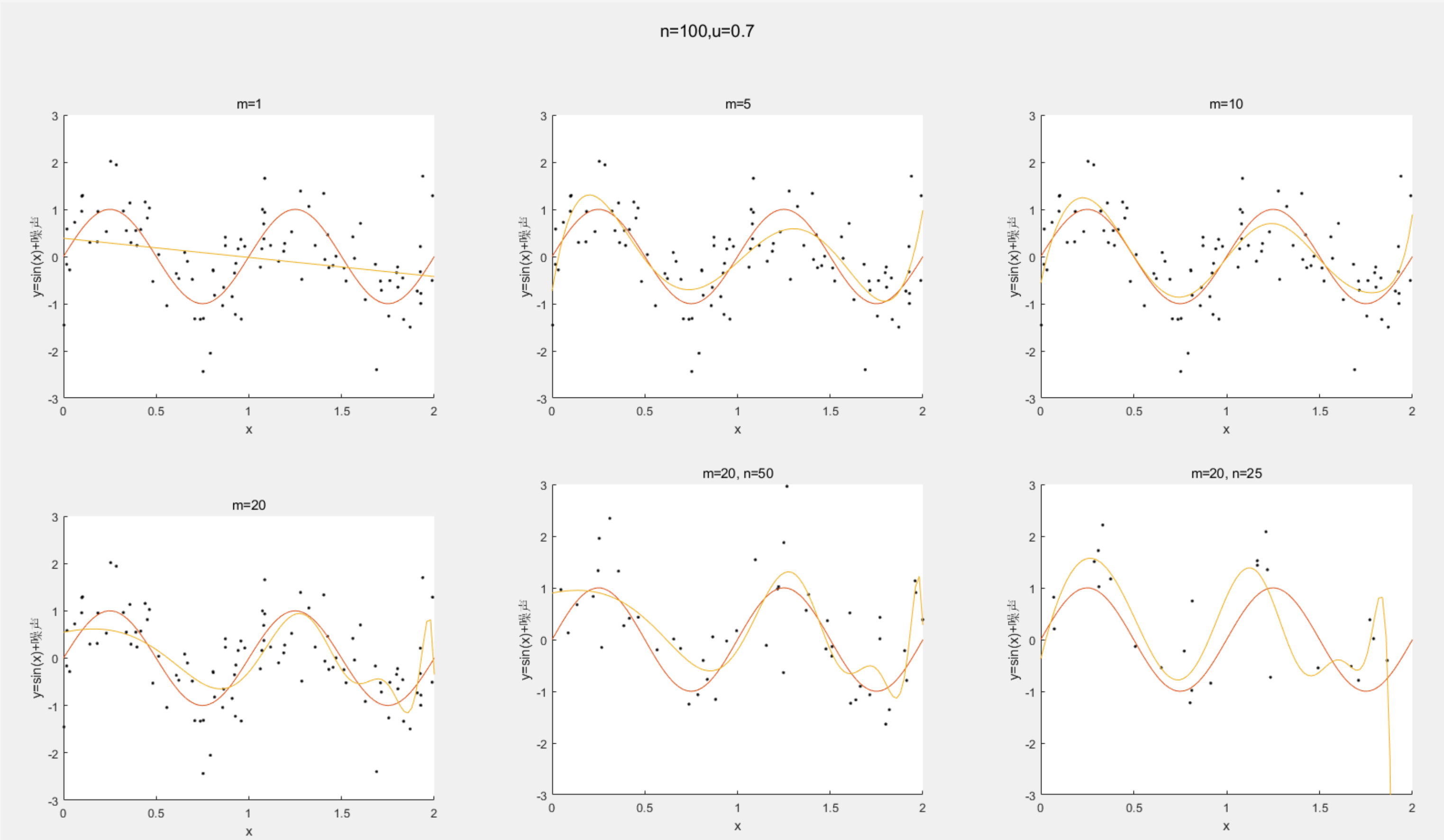
说明：

在生成的所有图像中

- 橙色曲线为正弦函数 $\sin 2\pi x$
- 黄色曲线为拟合的曲线

n为随机生成的数据量，m 为拟合函数的最高次数，u = 0.7 是加入的正态分布的噪声的标准差

4.1. 最小二乘法



分别模拟了6种情况

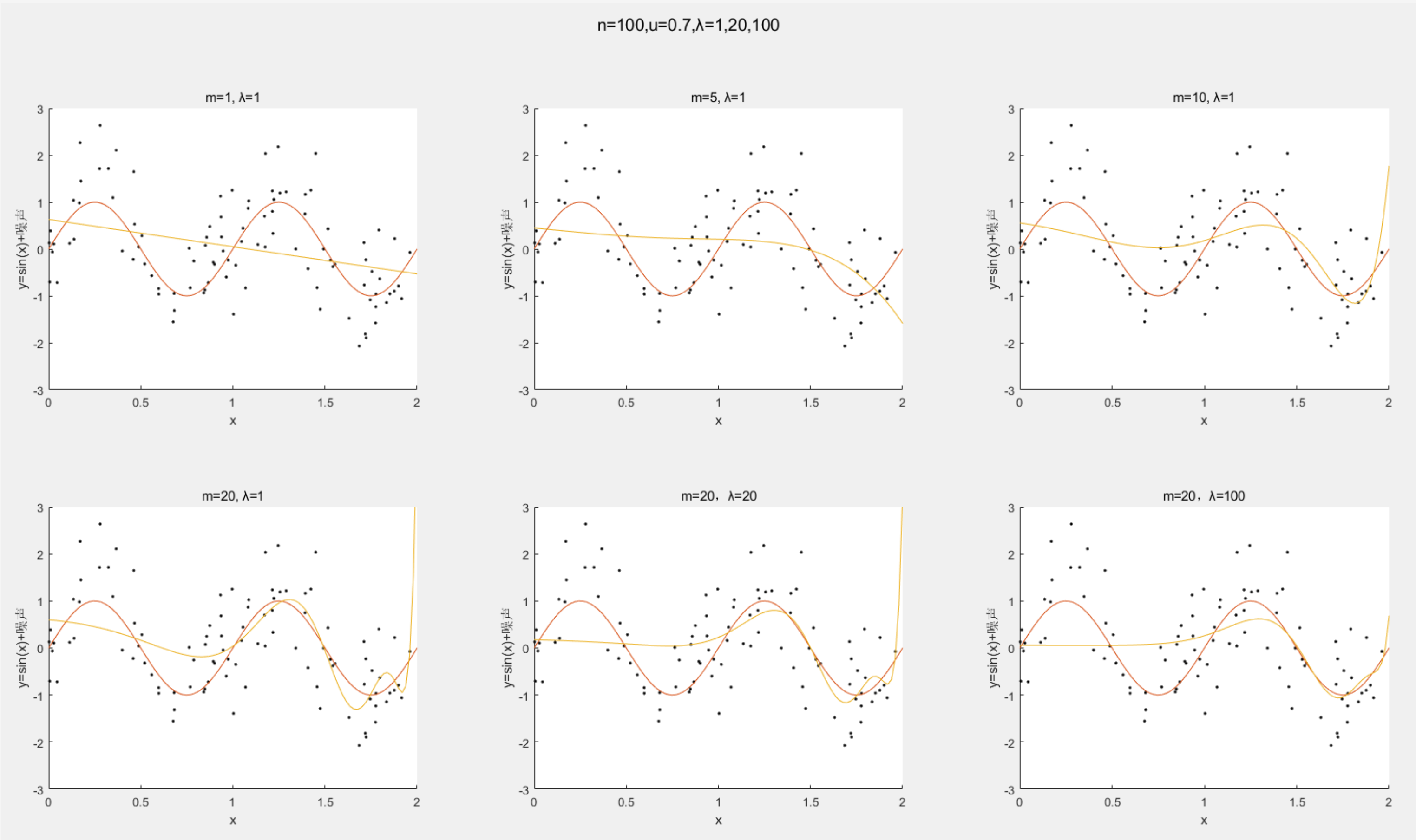
- n = 100（数据量不变，拟合函数的最高次数变）
  - m = 1
  - m = 5
  - m = 10
- m =20 （拟合函数的最高次数不变，数据量变）

- n = 100
- n = 50
- n = 25

发现

- 当数据量不变时
  - 现象：随着拟合函数的最高次数越来越大，拟合曲线越接近真实的正弦图像
  - 说明：拟合的函数次数越高，函数越复杂，曲线拟合的越好，落在上面的点越多
- 当拟合函数的最高次数不变时
  - 现象：随着数据量越来越多，拟合曲线越接近真实的正弦图像
  - 说明：拟合的数据量越多，说明给予的信息越充分，曲线拟合的越好，落在上面的点越多

4.2. 带惩罚项的最小二乘法



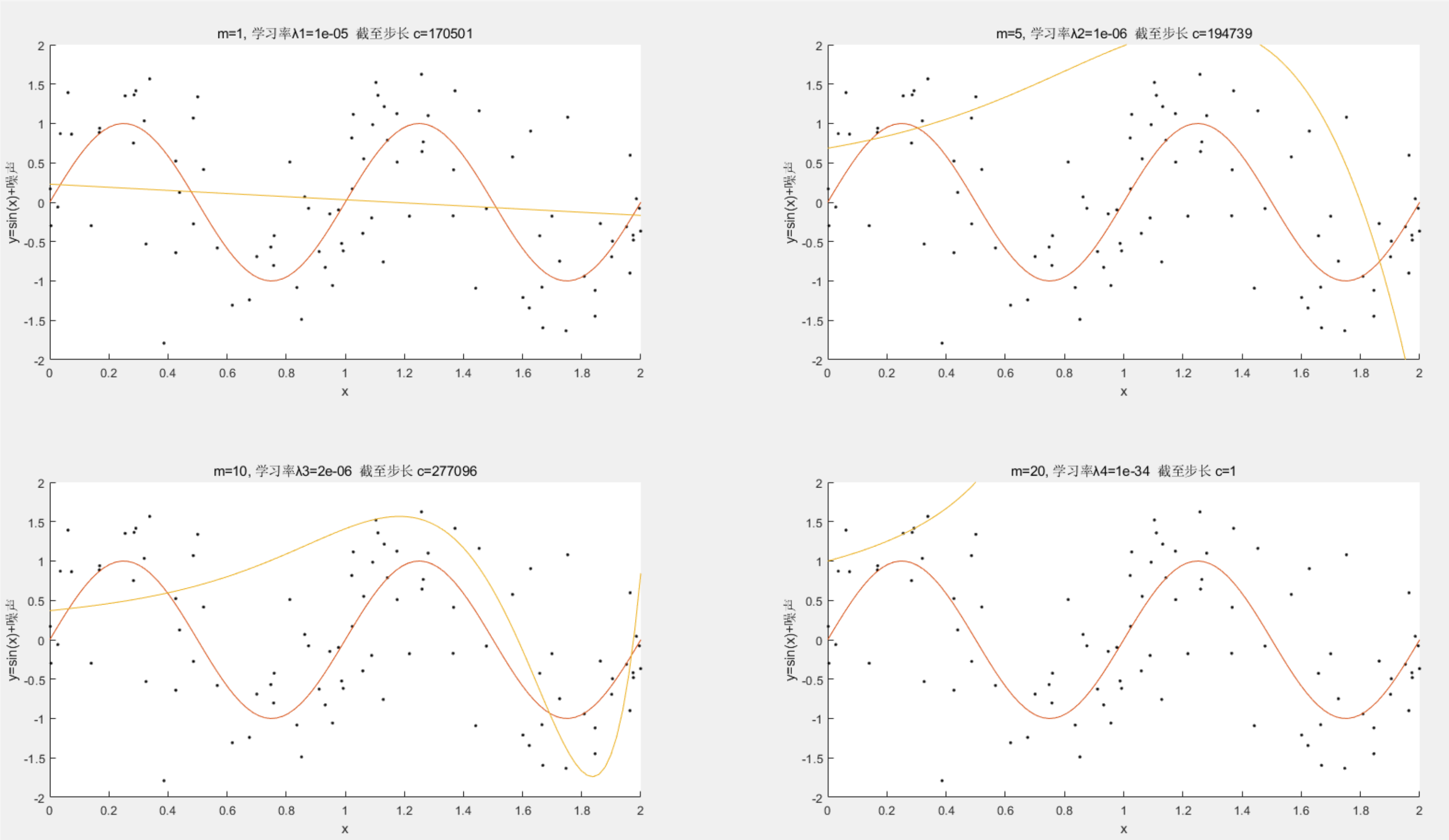
分别模拟了6种情况

- λ = 1（惩罚系数不变，拟合函数的最高次数变）
  - m = 1
  - m = 5
  - m = 10
- m = 20（拟合函数的最高次数不变，惩罚系数变）
  - λ = 1
  - λ = 20
  - λ = 100

发现

- 加入惩罚项之后，曲线拟合的比较离谱，拟合曲线的前一段都会比较平
- 惩罚系数越大，拟合曲线的泛化能力越好，但是相应的对训练集上的数据拟合就越不好

4.3. 梯度下降法



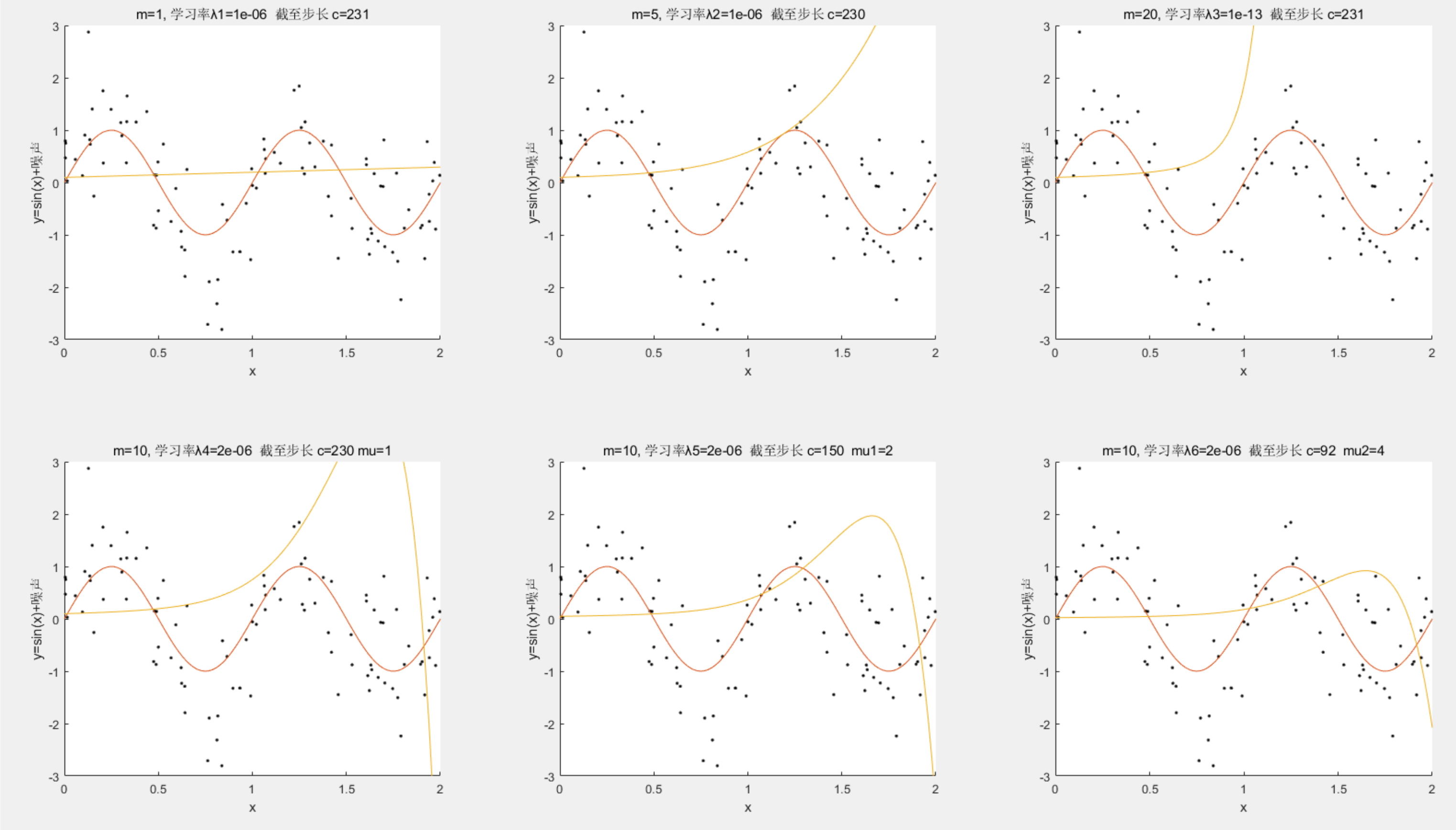
分别模拟了4种情况

- 拟合函数的最高次数变化
  - m =1，λ = 1 * 10^(-6)
  - m =5，λ = 1 * 10^(-6)
  - m =10，λ = 2 * 10^(-6)
  - m =20，λ = 1 * 10^(-34)

发现

调整学习速率真的很折磨人，而且梯度下降法拟合曲线比较奇怪，只有第3张图拟合的还稍微好一些，最后一张图比较奇怪，学习速率调的这么低，说明使用梯度下降法来拟合曲线，拟合函数的最高次数不能太高。

4.4. 带惩罚项的梯度下降法



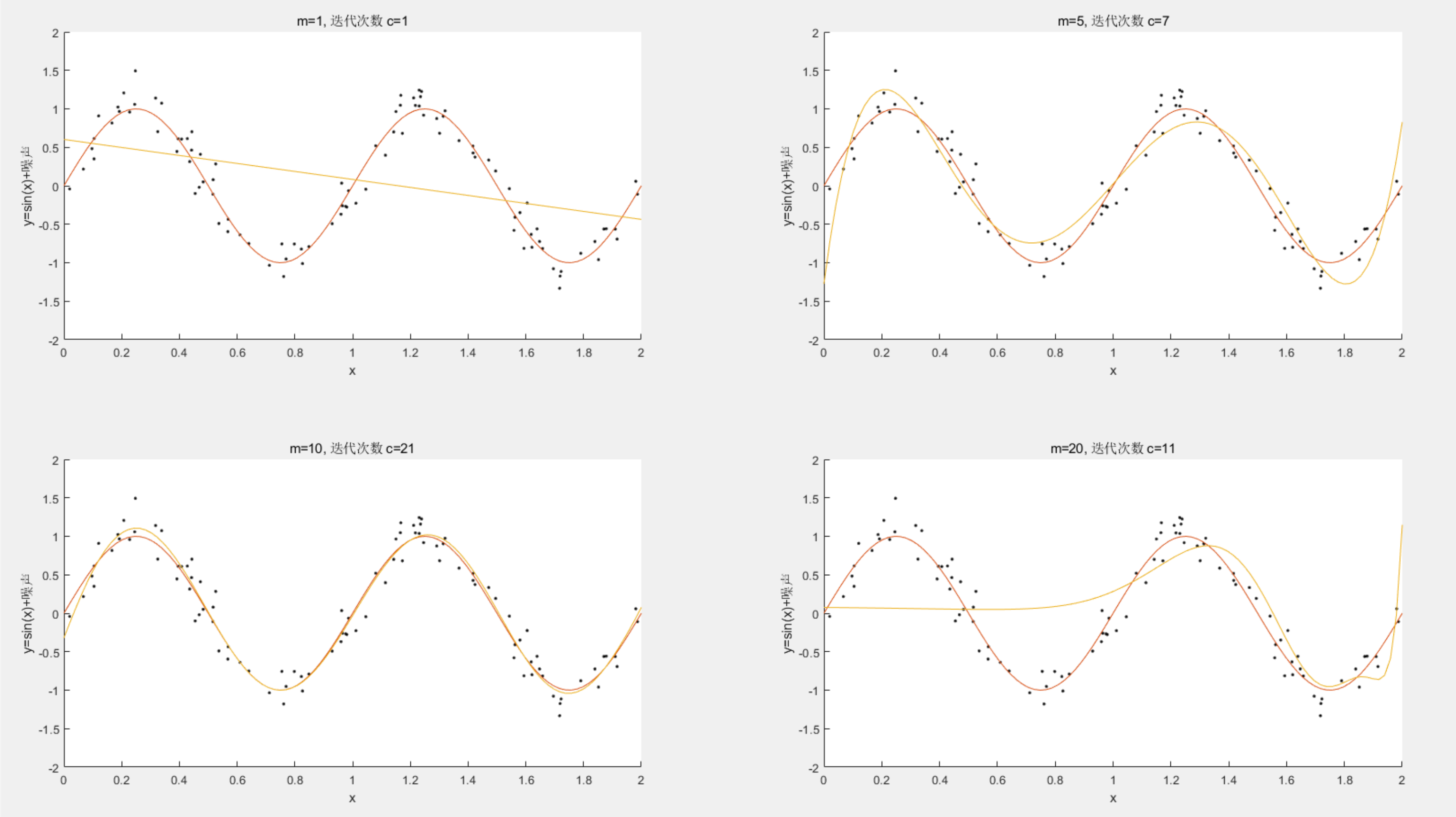
分别模拟了6种情况

- 拟合函数的最高次数变化
  - m =1，λ = 1 * 10^(-6)，mu = 1
  - m =5，λ = 1 * 10^(-6)，mu = 1
  - m =20，λ = 1 * 10^(-13)，mu = 1
- 惩罚系数变化
  - m =10，λ = 2 * 10^(-6)，mu = 1
  - m =10，λ = 2 * 10^(-6)，mu = 2
  - m =10，λ = 2 * 10^(-6)，mu = 4

发现

加入惩罚项的梯度下降法拟合的曲线比未加入惩罚项的好一些。

4.5. 共轭梯度法



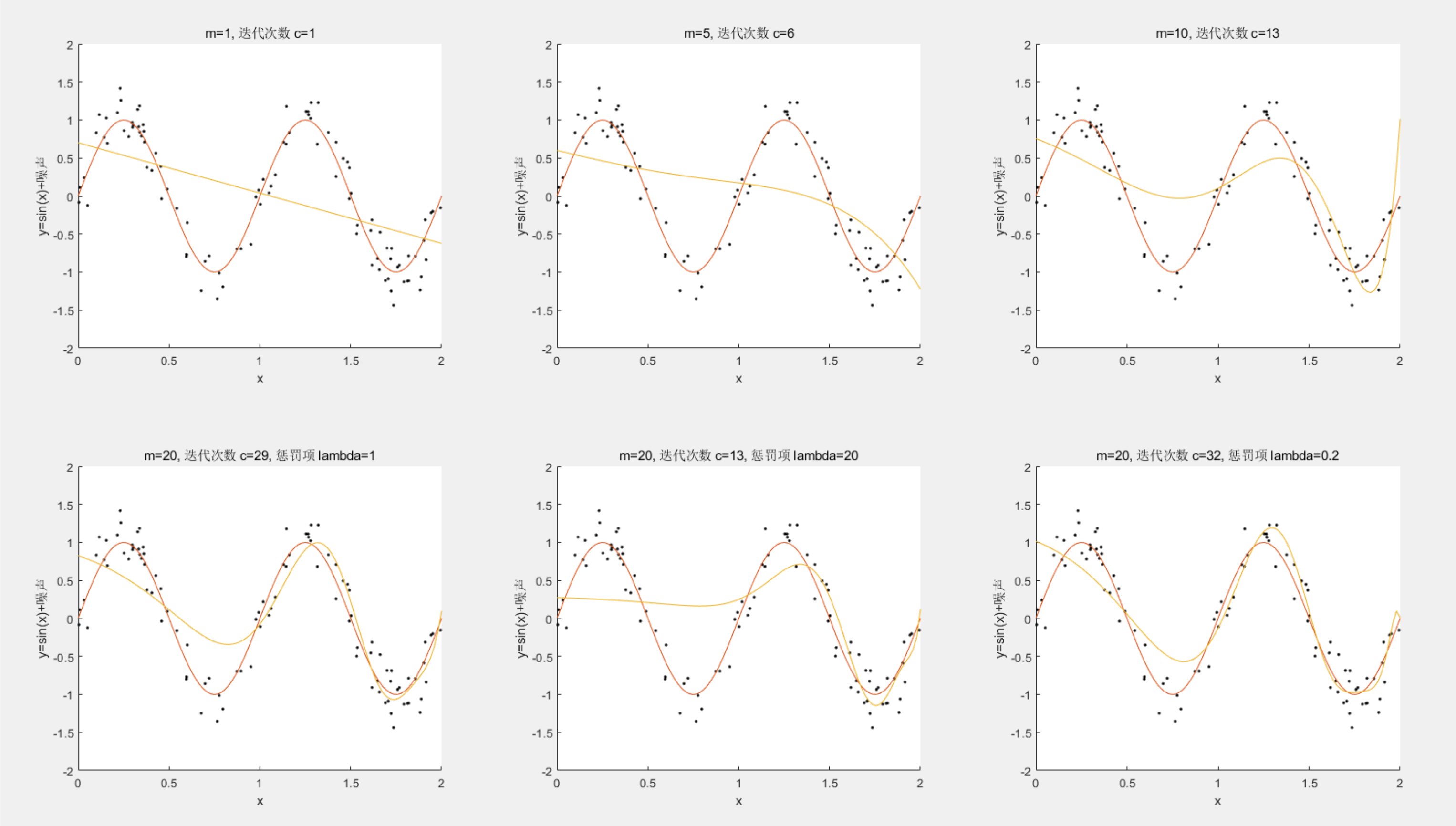
分别模拟了4种情况

- 拟合函数的最高次数变化
    - m =1
    - m =5
    - m =10
    - m =20

发现

通过比较发现，拟合函数的最高次数为 5 和 10 时，拟合的曲线十分好，拟合函数的最高次数过高或过低拟合的曲线都会出问题，另外迭代次数并没有十分明显的区分，但是和梯度下降法相比，迭代次数有了十分明显的下降

4.6. 带惩罚项的共轭梯度法



分别模拟了6种情况

- 拟合函数的最高次数变化
  - m =1
  - m =5
  - m =10
- 惩罚系数变化
  - m =20，λ = 1
  - m =20，λ = 20
  - m =20，λ = 0.2

发现

- 拟合函数的最高次数越高，拟合曲线与原曲线符合的越好
- 惩罚项越大，拟合曲线的泛化能力越强，但是在训练集的表现也越不好

5. 结论

5.1. 过拟合现象及克服过拟合的方法

过拟合现象

简单的理解就是，模型把训练集中的数据学习的太好，把一些不普适的特征也学习到了，导致在之后的预测中，错误率反而有提高的现象

在我的实验中，也有出现过拟合的情况，如使用共轭梯度法时 m=10 的情况，就是典型的过拟合现象

克服过拟合的方法

通过学习大概知道克服过拟合的方法有以下几种

- 减小拟合函数的最高次数
- 减少迭代次数，适时终止
- 增加数据集
- 加入惩罚项

5.2. 实验心得

1. 最小二乘法计算矩阵的逆的效率偏低，但是当矩阵的维度小于10000时，使用最小二乘法很方便，因为直接求的是解析解，也就是最优解。
2. 梯度下降法掌握的不好，在使用过程中，调整参数非常费时费力。学习速率小，运行速度慢；学习速率大，容易出现不收敛的情况。
3. 共轭梯度法是这次实验中表现较好的方法，但是其原理还未完全吃透，需要自己再推导公式，进行学习。

6. 参考文献

理论主要是参考的书籍是周志华老师的机器学习（西瓜书）和一些博客，以及英文的维基百科。

算法在具体的实现过程中参考了一些博客在使用matlab时的技巧

7. 附录

- 最小二乘法
  - Analytical_Penalty.m
  - Analytical_NoPenalty.m
- 梯度下降法
  - Gradient_Descent_Penalty.m
  - Gradient_Descent_NoPenalty.m
- 共轭梯度法
  - Conjugate_Gradient_Penalty.m
  - Conjugate_Gradient_NoPenalty.m
- 画图的函数
  - draw.m
- 生成数据的函数（实际并未调用）
  - product_data.m

代码文件在文件夹中