哈尔滨工业大学计算机科学与技术学院

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：1160300226

姓名： 赵书光

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、克服过拟合的方法(如加惩罚项、增加样本)

二、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：Pycharm

三、设计思想（本程序中的用到的主要算法及数据结构）

1．算法原理

* 解析解法(最小二乘法)：通过对loss函数直接求一阶导数，令一阶导数为零，此时得到的解析解即是我们要的曲线最低点的解。
* （批量）梯度下降法：是一种迭代法，先给定一个W，然后向W下降最快的方向调整J（loss函数），在若干次迭代之后找到局部最小，这里当学习率（步长）选的好的话，局部最优解可以视为最优解。
* 共轭梯度法：在回归模型中，回归系数 β 正是线性方程组 Ax=b 的解，其中 A=X′X，b=X′y。而共轭梯度法（Conjugate gradient, CG），就是想像上面这个式子一样，把解 X表达成共轭向量基的线性组合：只要依次算出所有的共轭向量 Pi 和对应的系数 αi，就可以得出 X。而在实际情况中，有可能用更少数目的 Pi就能得到对 X 的良好近似，于是在这个意义上 CG 就是一种迭代法了。
* 比较：梯度下降即给定一个值 x0，计算当前的梯度，然后沿着该梯度方向移动到下一个更新值 x1，再计算梯度，如此反复循环。而共轭梯度法，则是说我们并不是沿着梯度走，而是沿着所谓的“共轭梯度”，即 pi，进行移动，每次移动的方向是共轭的（即关于 A 是正交的），因此不会有重复的劳动。

1. 算法的实现

1.生成数据(n 维)：

x:定义域[0,1]

t：sin(2Πx)+高斯噪声(均值为0，方差为1)

2.拟合曲线：

利用多项式（泰勒展开）去逼近数据函数：

拟合函数(m+1 阶)：y（xn,w）= w0+w1\*x+w2\*x...+wm\*x

误差函数：E(W) = 1/2n (y(xn,w)-tn) 

利用矩阵转化：E(W) = 1/2n (XW-T)’(XW-T)

①解析解：

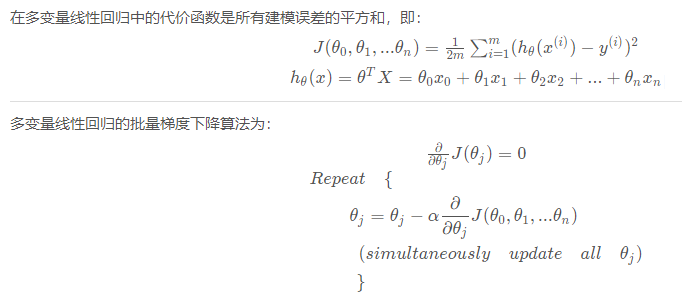
求得 = 1/n \* (X’XW-X’T),令其等于0，得到W=(X’X)X’T

将得到的W向量（代表系数）带入误差函数，画出拟合曲线。

加入正则项（惩罚项）:E(W) = 1/2n ((XW-T)’(XW-T) + λW’W ),

W=(X’X+λI)X’T.

②梯度下降：



这里做一下处理，其实我们这个问题处理的是单变量非线性回归问题，但是那样太复杂了，我们转化为多变量线性回归问题，就像降了维一样，即：

1=x0,x=x1,x=x2......x=xn

然后不用一个一个去求W的每一项，我们直接得到W向量的值：

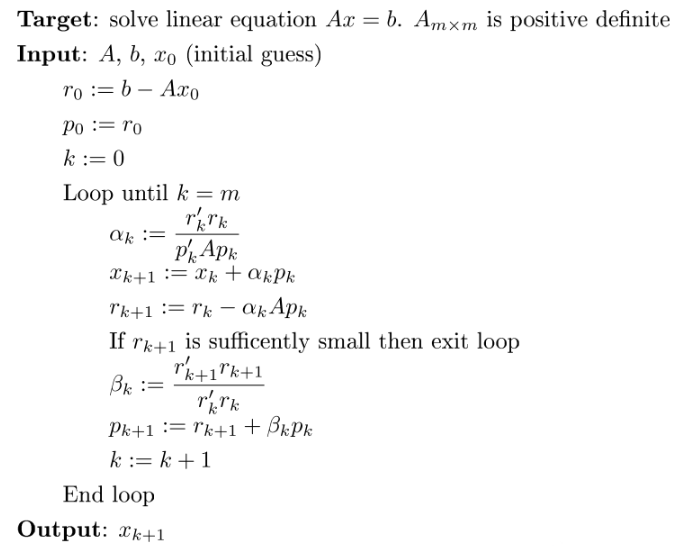
W = W - ɑ\*

其中ɑ是学习率，也就是每一次沿着梯度最快的方向走的步长。

加入惩罚项： W = W - ɑ\*式子不变，但是变为

1/n \* (X’XW-X’T+λW).

③共轭梯度：



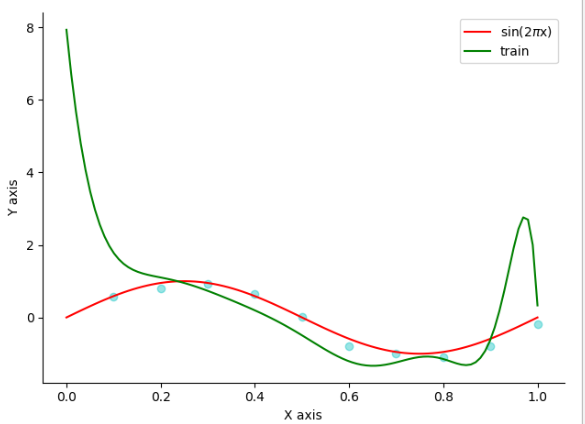
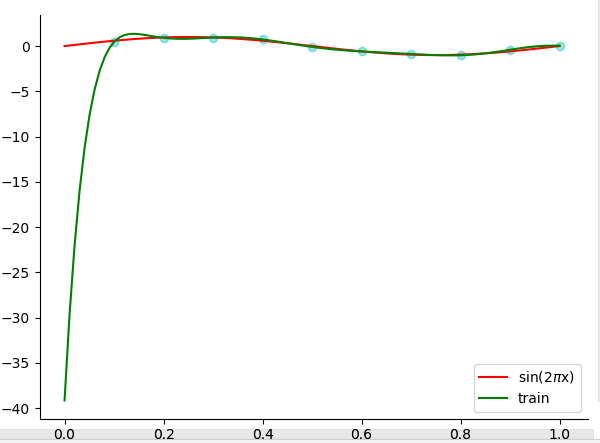
此问题中：A = X’X , b = X’T ,得到的x解即是此问题的W的解

加入惩罚项：A = X’X+λI.

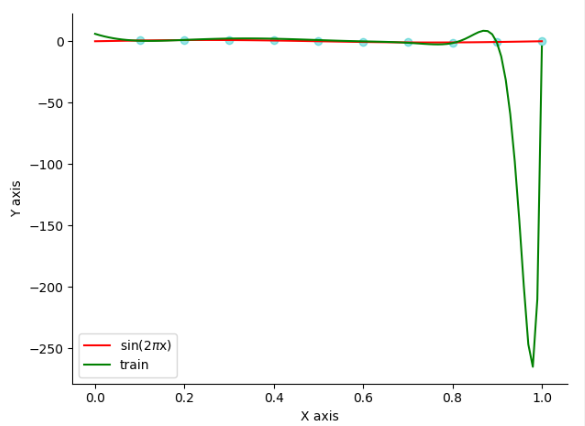
1. 实验结果与分析

样本数据量：10

解析解：



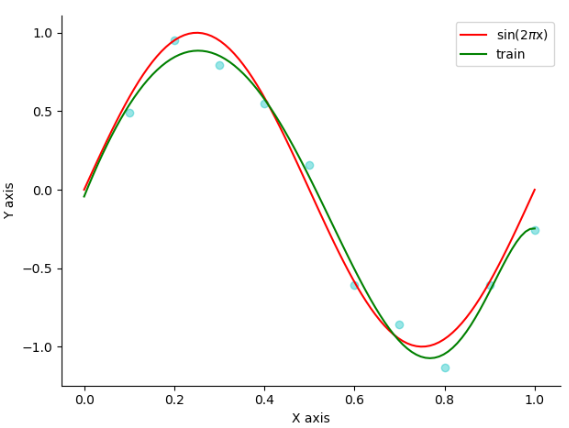
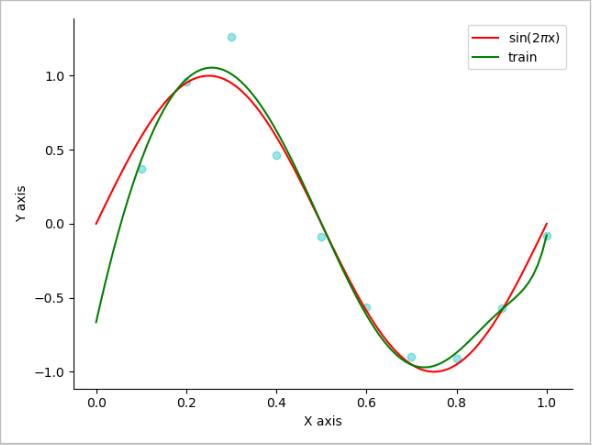
n=10,m=10 n=10,m=20



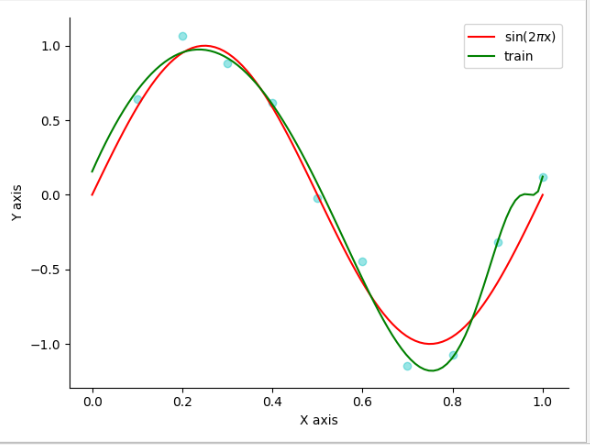
n=10,m=30

三个图片依次为m阶数等于10，20，30的情况，可以看出当数据量小的时候，拟合的情况不是很理性，而且能看出，随着m阶数的增大会出现过拟合的情况，比如图三的过拟合较为明显。

加入惩罚项：



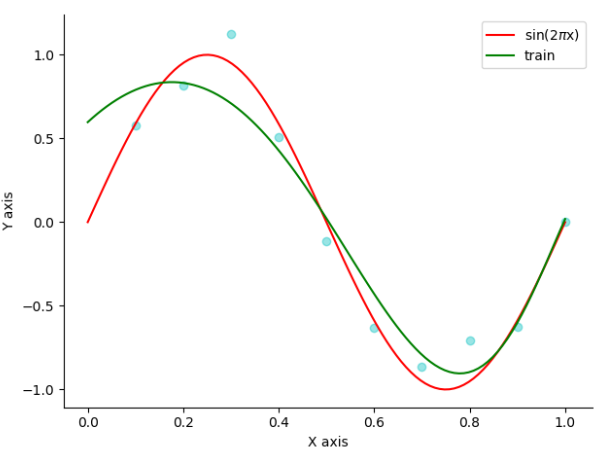
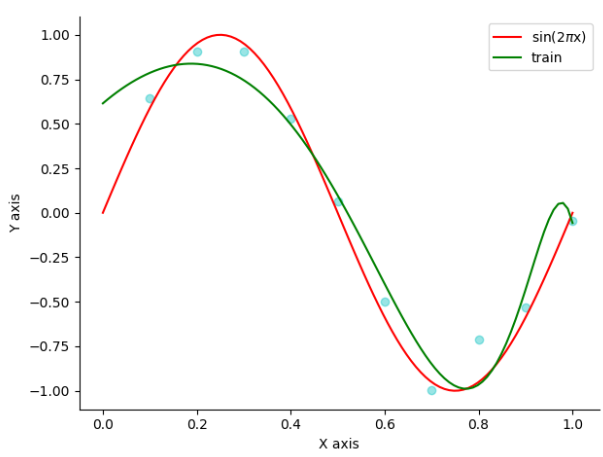
n=10,m=10 n=10,m=20



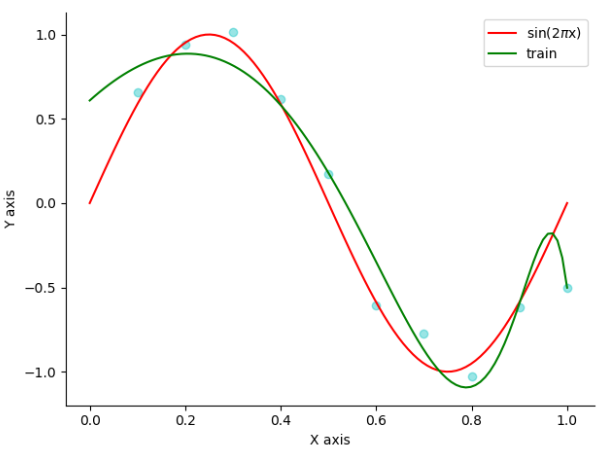
n=10,m=30

加入惩罚项后，能看出来拟合曲线已经很逼近正弦函数了，加入惩罚项只能说是可以减弱过拟合的，但是绝对不是消除，所以当m阶数很大的时候能看出来图三的过拟合还是很明显的，几乎经过了所有的十个点。

梯度下降：



n=10,m=10 n=10,m=20

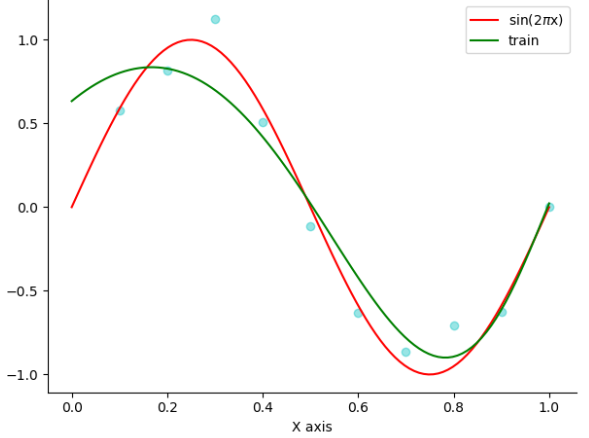


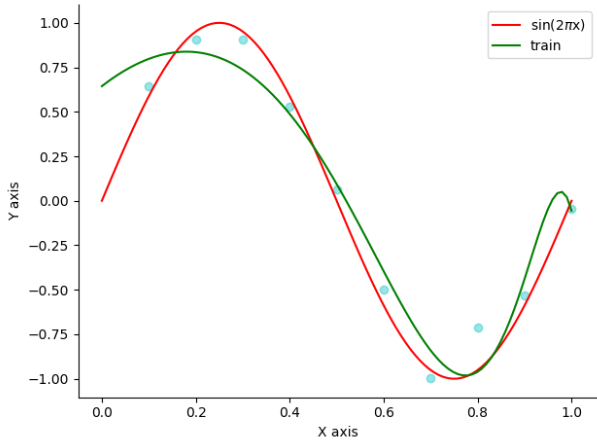
n=10,m=30

随着m阶数的增大，拟合曲线的表达能力在增强，但是会出现过拟合的情况。m=30时的过拟合还是很明显的。

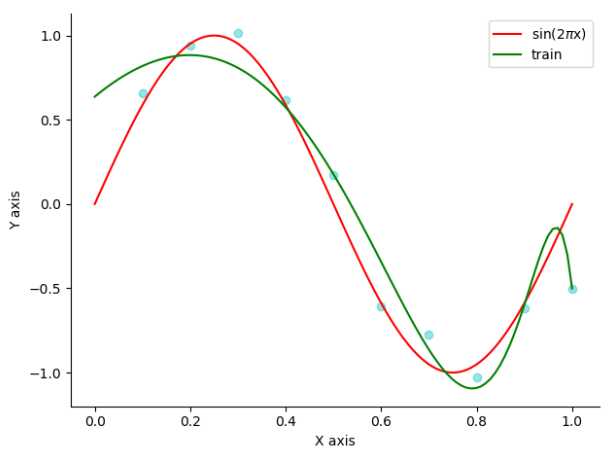
另一方面，能看出来梯度下降的速度是计算的最慢的，因为梯度下降要不断的迭代，以寻找到局部最优解，要在学习率和步长之间调试出一个比较理想的组合。但是这只是数据量不太大的情况，当数据量非常非常大的时候，其实梯度下降是要比解析解法快的，因为解析解要消耗很大的空间来进行各种复杂的矩阵运算。

（我选的步长是0.05，迭代次数是1万次）

加入惩罚项：



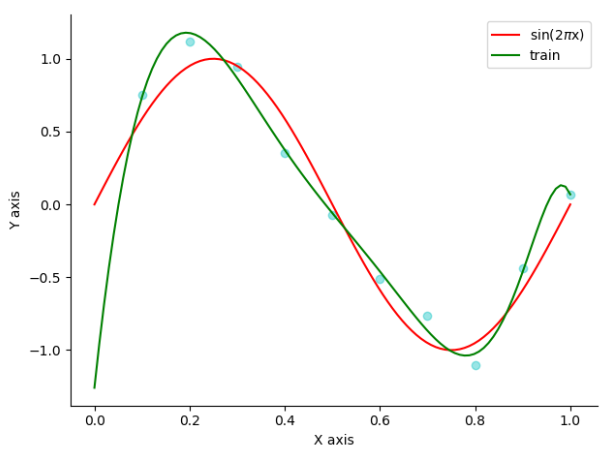
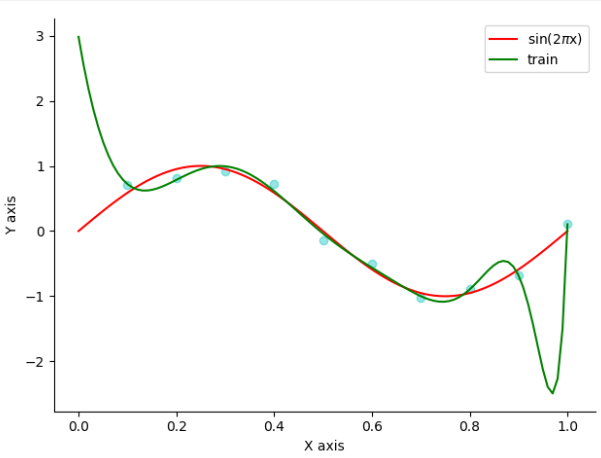
n=10,m=10 n=10,m=20



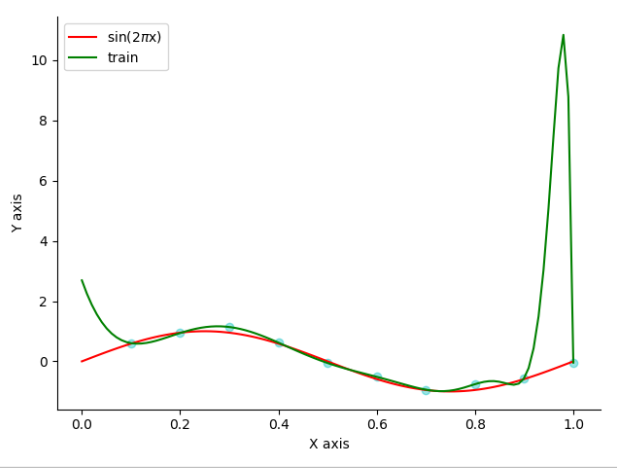
n=10,m=30

梯度下降里加入惩罚项，效果不明显，几乎没有什么变化。

共轭梯度：



n=10,m=10 n=10,m=20

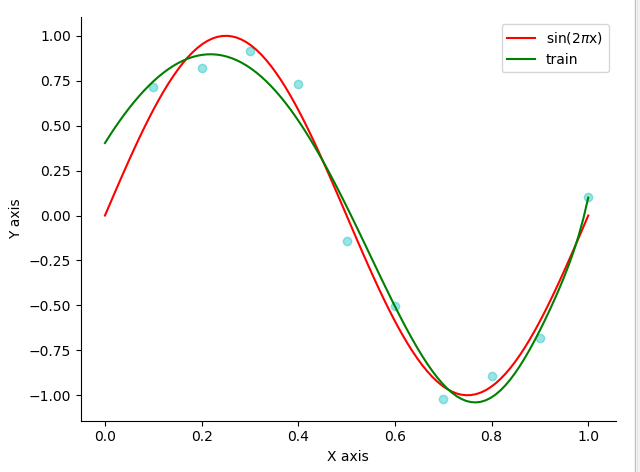
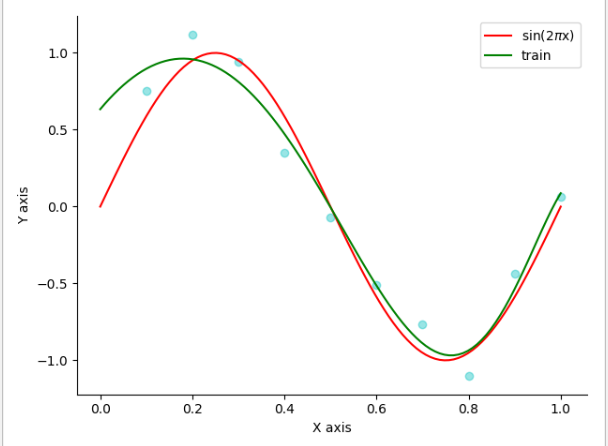


n=10,m=30

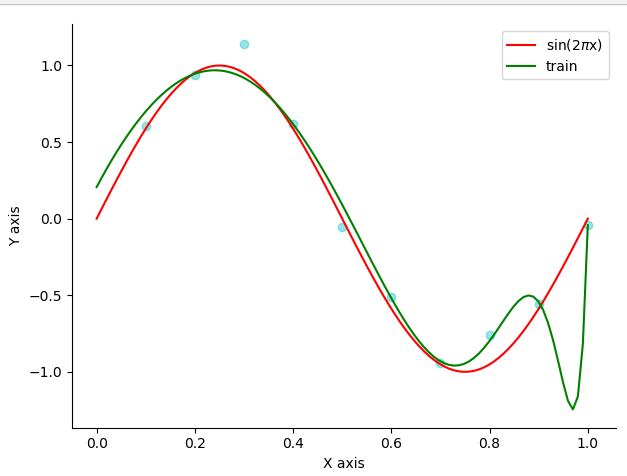
当不加入惩罚项的时候，共轭梯度的过拟合是看的最明显的，这三个图都很好的呈现了过拟合的状态。

另一方面，能看比较出在三个方法中，共轭梯度速度是最快的，前面说过，因为我们并不是沿着梯度走，而是沿着所谓的“共轭梯度”，即 pi，进行移动，每次移动的方向是共轭的（即关于 A是正交的），因此不会有重复的劳动。

加入惩罚项：



n=10,m=10 n=10,m=20



n=10,m=30

共轭梯度也是三个方法里，加入惩罚项后克服过拟合最明显的一个.对共轭梯度顿生好感:) . 前两个图能看出来很好的克服了过拟合，但是当m阶数到40的时候，能看出来即使加入了惩罚项也出现了过拟合的情况，这也正验证了加入惩罚项只能减弱过拟合而不能消除过拟合的理论。

样本数据量：100个

克服过拟合的方法有两种：加入惩罚项（正则项）或者加大样本数据量。

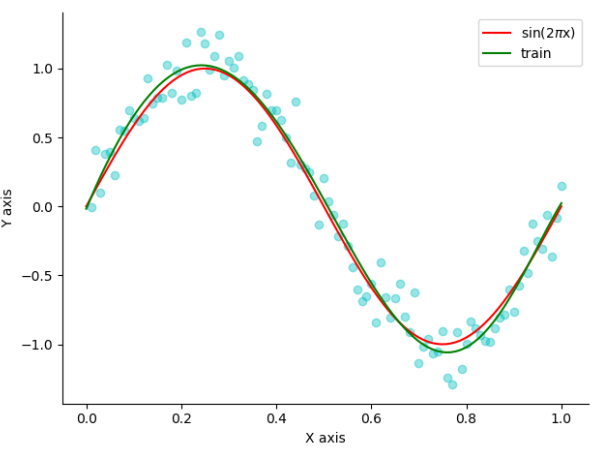
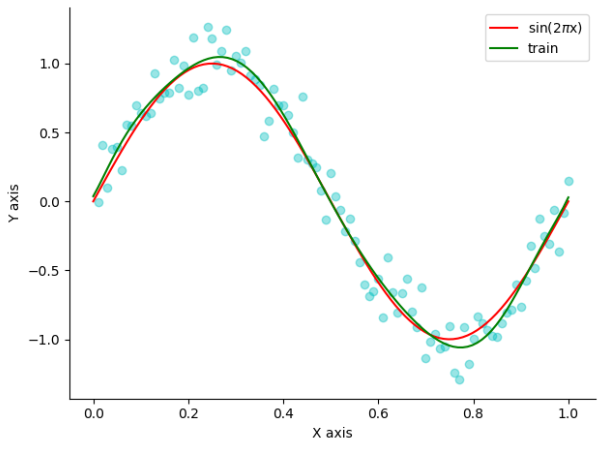
而样本数据量的加大克服过拟合的更加明显，也就是说不加入惩罚项的大数据量拟合曲线也能拟合的很好。

因为在100个数据量的情况下，三个方法拟合的都很好.

具体的话，共轭梯度法拟合的最好，几乎都和正弦函数重合了；梯度下降拟合的最差，因为前面讲过，梯度下降最后求的是局部最优解并不是最优解。

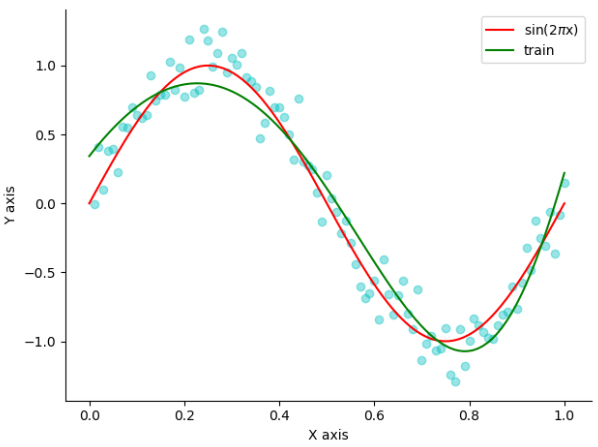
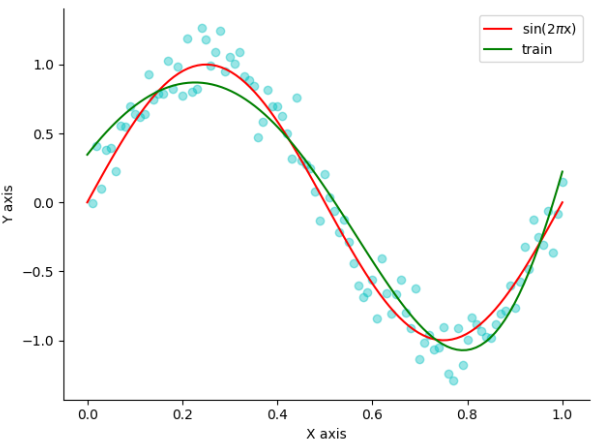
下面就不做具体分析了，只把拟合图像展示一下。

解析解:



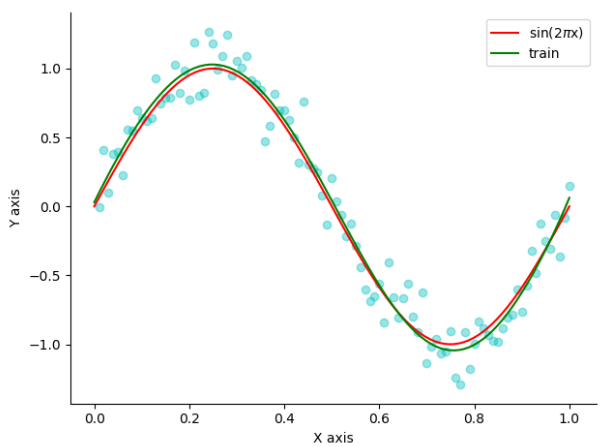
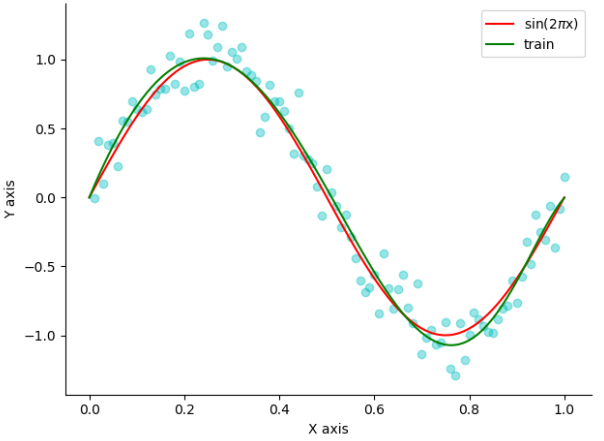
n=100 ,m=10 n=100 ,m=10(加入惩罚项)

梯度下降：



n=100 ,m=10 n=100 ,m=10(加入惩罚项)

共轭梯度：



n=100 ,m=10 n=100 ,m=10(加入惩罚项)

1. 结论

**Ⅰ**解析解法:是直接对J求导找出全局最小，是非迭代法。在数据量比较小的时侯会产生过拟合的情况，这个时候加入惩罚项可以减弱一些；但是随着m阶数的增大加入惩罚项也无法克服过拟合的情况。数据量很大的时候有，过拟合的情况明显得到改善，加不加入惩罚项几乎没有影响，而且拟合曲线已经很毕竟理想函数曲线（正弦曲线函数）。

要注意的是，最小二乘法要求矩阵可逆。并且，解析解法是用了很大的空间来存储矩阵并进行矩阵运算的，数据量如果特别大比如千级或者万级，那么解析解法速度会变得很慢，甚至系统瘫痪。

**Ⅱ**梯度下降法:是一种迭代法，先给定一个X，然后向X下降最快的方向调整J，在若干次迭代之后找到局部最小。梯度下降法的缺点是到最小点的时候收敛速度变慢，并且需要调试学习率ɑ和迭代次数以达到最佳状态。

梯度下降是可以进行优化的，减少学习率和迭代次数不知如何选择带来的缺点，比如用特征缩放等方法。

要注意的是，梯度下降求出来的并不是全局最优解而是局部最优解，当学习率和迭代次数适当的时候可以认为两者是相等的。还有，数据量不大的时候（比如100以内）梯度下降法是最慢的，因为迭代次数决定了梯度下降不会很快，但是当数据量很大的时候，梯度下降比解析解是有优势的，不需要耗费太多的空间。

**Ⅲ**共轭梯度法:并不是沿着梯度走，而是沿着所谓的“共轭梯度”进行移动，每次移动的方向是共轭的。

在所有的实验测试中，共轭梯度法是这里边拟合的最好的方法，由于共轭梯度方法沿着共轭方向移动的优势，速度也是这里边最快的。

至于过拟合的情况，前面都做了比较详细的介绍和比较了，不做叙述了。

1. 参考文献

# matplotlib绘图：<https://blog.csdn.net/u014453898/article/details/73395522>

numpy矩阵使用：<https://blog.csdn.net/taxueguilai1992/article/details/46581861>

多变量梯度下降：<https://www.kancloud.cn/baozou/stanfordml/562649>

共轭梯度：<https://cosx.org/2016/11/conjugate-gradient-for-regression/>

1. 附录：源代码（带注释）

Github地址：

<https://github.com/KobeHV/machine-learning-linear-regression>

1. import numpy as np
2. import matplotlib.pylab as plt
3. import random
4. import math

7. #第一部分
9. #取观测值x.y,绘制散点图
10. a = []
11. b = []
12. x=0
13. xnum = 100#数据集x数目，可以更改
14. def func(x):
15. mu=0#均值
16. sigma=0.15#方差
17. epsilon = random.gauss(mu,sigma) #高斯分布随机数
18. **return** np.sin(2\*np.pi\*x)+epsilon
19. **for** i in range(0,xnum):
20. x=x+1.0/xnum
21. a.append(x)
22. b.append(func(x))

25. #创建X,T,W矩阵
26. m = 10#M阶数，可以更改
27. n = len(a)
28. Xa = np.zeros(shape=(n,m+1))
29. **for** i in range(0,n):
30. **for** j in range(0,m+1):
31. Xa[i,j] = a[i]\*\*j
32. X = np.mat(Xa)
33. Ta = np.zeros(shape=(n,1))
34. **for** i in range(0,n):
35. Ta[i] = b[i]
36. T = np.mat(Ta)
37. W = W = np.mat(np.ones(shape=(m + 1, 1)))

40. #第二部分
42. #解析解
43. def Mode1\_1():
44. W = (X.T \* X).I \* X.T \* T  # 不加惩罚项
45. **return** W
46. def Mode1\_2():
47. lamda = math.exp(-10)
48. I = np.eye(m + 1)  # 加惩罚项
49. Ie = lamda\*I
50. W = (X.T \* X + Ie).I \* X.T \* T
51. **return** W
53. #梯度下降
54. def Mode2\_1():
55. alpha = 0.05  # 学习率
56. iterNum = 100000  # 迭代次数
57. W = np.mat(np.ones(shape=(m + 1, 1)))
58. **for** i in range(0, iterNum):
59. # E(W)对矩阵W的一阶导数,结果是一个矩阵
60. Differential = 1.0 / xnum \* (X.T \* X \* W - X.T \* T)# 不加惩罚项
61. W = W - alpha \* Differential
62. **return** W
63. def Mode2\_2():
64. alpha = 0.05  # 学习率
65. iterNum = 100000  # 迭代次数
66. lamda = math.exp(-8)
67. W = np.mat(np.ones(shape=(m + 1, 1)))
68. **for** i in range(0, iterNum):
69. # E(W)对矩阵W的一阶导数,结果是一个矩阵
70. Differential = 1.0 / xnum \* (X.T \* X \* W - X.T \* T + lamda \* W)# 加惩罚项
71. W = W - alpha \* Differential
72. **return** W
74. #共轭梯度
75. def Mode3\_1():
76. A = X.T \* X
77. B = X.T \* T
78. W = np.mat(np.ones(shape=(m + 1, 1)))
79. R = W
80. R = B - A \* W
81. P = R
82. beta = 0
83. k = 0
84. alfa = 0
85. **while** (k != m):
86. R1 = R
87. alfa = R.T \* R \* 1.0 / (P.T \* A \* P)
88. W = W + alfa[0, 0] \* P
89. R = R - alfa[0, 0] \* A \* P
90. beta = R.T \* R \* 1.0 / (R1.T \* R1)
91. P = R + beta[0, 0] \* P
92. k = k + 1
93. **return** W
94. def Mode3\_2():
95. lamda = math.exp(-8)
96. I = np.eye(m + 1)  # 加惩罚项
97. Ie = lamda \* I
98. A = X.T\*X + Ie
99. B = X.T \* T
100. W = np.mat(np.ones(shape=(m + 1, 1)))
101. R = W
102. R = B - A \* W
103. P = R
104. beta = 0
105. k = 0
106. alfa = 0
107. **while** (k != m):
108. R1 = R
109. alfa = R.T \* R \* 1.0 / (P.T \* A \* P)
110. W = W + alfa[0, 0] \* P
111. R = R - alfa[0, 0] \* A \* P
112. beta = R.T \* R \* 1.0 / (R1.T \* R1)
113. P = R + beta[0, 0] \* P
114. k = k + 1
115. **return** W
117. #第三部分
119. #拟合多项式函数
120. def match(x):
121. sum = W[0,0]
122. **for** i in range(1,m+1):
123. sum += W[i,0]\*(x\*\*i)
124. **return** sum
126. #绘制sin(x)标准曲线和拟合曲线
127. def draw():
128. plt.figure(1, dpi=100)
129. plt.figure()
130. plt.xlabel("X axis")
131. plt.ylabel("Y axis")
132. plt.scatter(a, b, c='c', alpha=0.4)
133. x = np.linspace(0, 1, 100)  # 中间间隔100个元素
134. y1 = np.sin(2 \* np.pi \* x)
135. y2 = match(x)
136. # 显示LaTex符号,用$\--$表示
137. plt.plot(x, y1, color="r", label='sin(2$\pi$x)')
138. plt.plot(x, y2, color="g", label='train')
139. plt.legend()
140. # 去掉右边框和上边框
141. ax = plt.gca()
142. ax.spines['right'].set\_color('none')
143. ax.spines['top'].set\_color('none')
144. # 显示所画的图
145. plt.show()
146. **return**
148. #选择模式
149. choice = 0
150. **while**(1):
151. choice = input("please input mode:")
152. **if** choice == '1':
153. W = Mode1\_1()
154. draw()
155. elif choice == '2':
156. W = Mode1\_2()
157. draw()
158. elif choice == '3':
159. W = Mode2\_1()
160. draw()
161. elif choice == '4':
162. W = Mode2\_2()
163. draw()
164. elif choice == '5':
165. W = Mode3\_1()
166. draw()
167. elif choice == '6':
168. W = Mode3\_2()
169. draw()
170. elif choice == '-1':
171. exit(0)
172. **else**:
173. print("Input Error,Please input a number 1-6, -1 is exit")